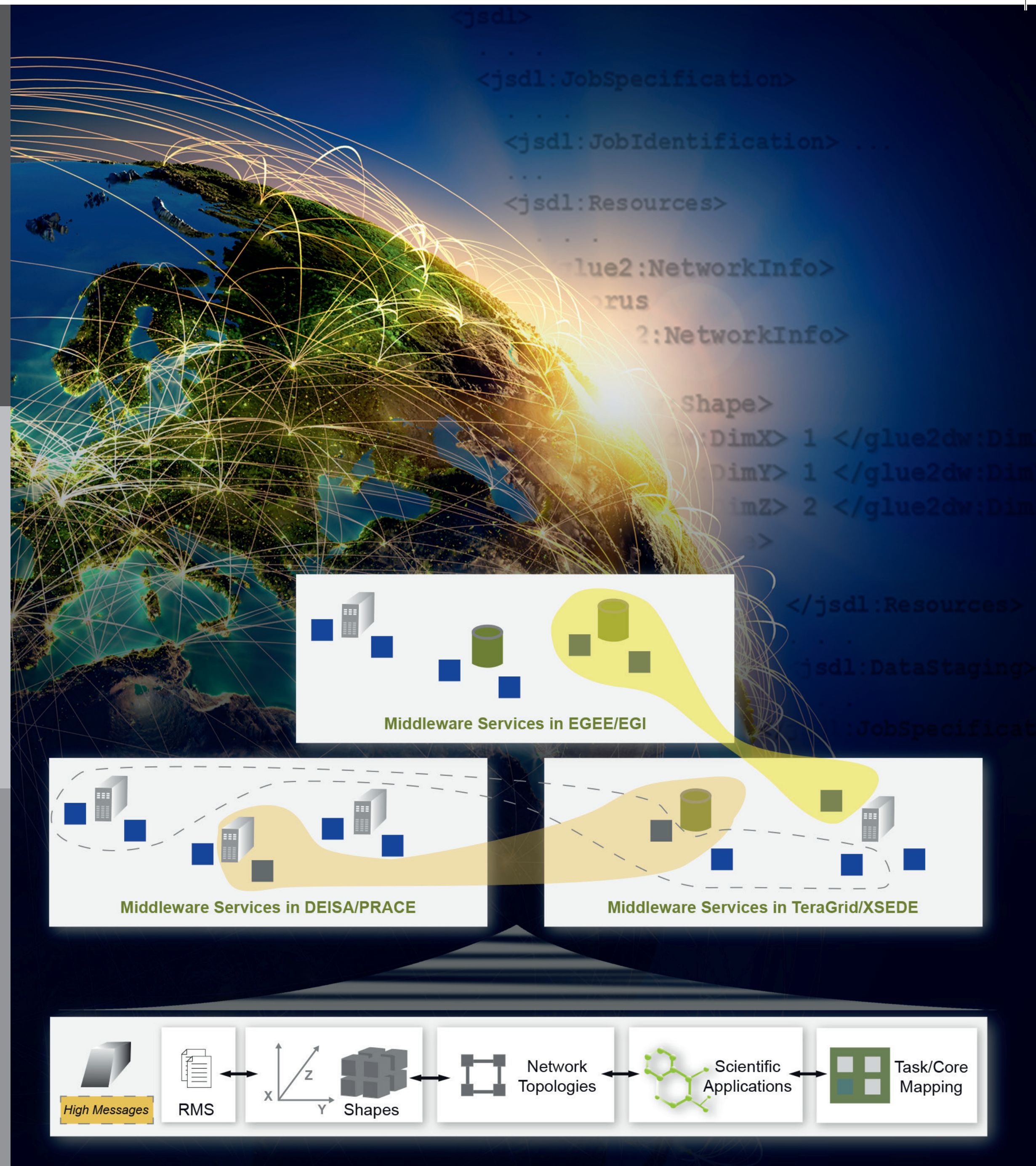
Computational simulations and thus scientific computing is the third pillar alongside theory and experiment in todays science. The term e-Science evolved as a new research field that focuses on collaboration in key areas of science using next generation data and computing infrastructures (i.e. e-Science infrastructures) to extend the potential of scientific computing. During the past decade, significant international and broader interdisciplinary research is increasingly carried out by global collaborations that often share resources within a single production e-Science infrastructure. More recently, increasing complexity of e-Science applications that embrace multiple physical models (i.e. multi-physics) and consider a larger range of scales (i.e. multi-scale) is creating a steadily growing demand for world-wide interoperable infrastructures that allow for new innovative types of e-Science by jointly using different kinds of e-Science infrastructures. But interoperable e-Science infrastructures are still not seamlessly provided today and this thesis argues that this is due to the absence of a production-oriented e-Science infrastructure reference model. The goal of this thesis is thus to present an infrastructure interoperability reference model (IIRM) design tailored to production needs and that represents a trimmed down version of the Open Grid Service Architecture (OGSA) in terms of functionality and complexity, while on the other hand being more specifically useful for production and thus easier to implement. This reference model is underpinned with lessons learned and numerous experiences gained from production e-Science application needs through accompanying academic case studies of the bio-informatics, e-Health, and fusion domain that all seek to achieve research advances by using interoperable e-Science infrastructures on a daily basis. Complementary to this model, a seven segment-based process towards sustained infrastructure interoperability addresses important related issues like harmonized operations, cooperation, standardization as well as common policies and joint development roadmaps.

This publication was edited at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.

IAS Series
Volume 16
ISBN 978-3-89336-861-7

JÜLICH
FORSCHUNGSZENTRUM

JÜLICH FORSCHUNGSZENTRUM
e-Science Infrastructure Design and Applications
Morris Riedel
IAS Series
16
Mitglied der Helmholtz-Gemeinschaft



Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures

Morris Riedel

JÜLICH
FORSCHUNGSZENTRUM

Schriften des Forschungszentrums Jülich

IAS Series Volume 16

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Jülich Supercomputing Centre (JSC)

Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures

Morris Riedel

Schriften des Forschungszentrums Jülich

IAS Series Volume 16

ISSN 1868-8489 ISBN 978-3-89336-861-7

Bibliographic information published by the Deutsche Nationalbibliothek.
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Publisher and Distributor:	Forschungszentrum Jülich GmbH Zentralbibliothek 52425 Jülich Phone +49 (0) 24 61 61-53 68 · Fax +49 (0) 24 61 61-61 03 e-mail: zb-publikation@fz-juelich.de Internet: http://www.fz-juelich.de/zb
Cover Design:	Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH Weltkugel auf dem Cover: © Anton Balazh - Fotolia.com
Printer:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2013

Schriften des Forschungszentrums Jülich
IAS Series Volume 16

Diss. (Karlsruher Institut für Technologie (KIT), 2012)

ISSN 1868-8489
ISBN 978-3-89336-861-7

Persistent Identifier: urn:nbn:de:0001-2013031903
Resolving URL: http://www.persistent-identifier.de/?link=610

Abstract

Computational simulations and thus scientific computing is the third pillar alongside theory and experiment in todays science. The term e-Science evolved as a new research field that focuses on collaboration in key areas of science using next generation data and computing infrastructures (i.e. e-Science infrastructures) to extend the potential of scientific computing. During the past decade, significant international and broader interdisciplinary research is increasingly carried out by global collaborations that often share resources within a single production e-Science infrastructure. More recently, increasing complexity of e-Science applications that embrace multiple physical models (i.e. multi-physics) and consider a larger range of scales (i.e. multi-scale) is creating a steadily growing demand for world-wide interoperable infrastructures that allow for new innovative types of e-Science by jointly using different kinds of e-Science infrastructures. But interoperable e-Science infrastructures are still not seamlessly provided today and this thesis argues that this is due to the absence of a production-oriented e-Science infrastructure reference model. The goal of this thesis is thus to present an infrastructure interoperability reference model (IIRM) design tailored to production needs and that represents a trimmed down version of the Open Grid Service Architecture (OGSA) in terms of functionality and complexity, while on the other hand being more specifically useful for production and thus easier to implement. This reference model is underpinned with lessons learned and numerous experiences gained from production e-Science application needs through accompanying academic case studies of the bio-informatics, e-Health, and fusion domain that all seek to achieve research advances by using interoperable e-Science infrastructures on a daily basis. Complementary to this model, a seven segment-based process towards sustained infrastructure interoperability addresses important related issues like harmonized operations, cooperation, standardization as well as common policies and joint development roadmaps.

Zusammenfassung

Computersimulationen sowie wissenschaftliches Rechnen sind heutzutage der dritte Pfeiler neben Theorie und Experiment in der Wissenschaft. Das Thema "e-Science" hat sich als ein neues Forschungsfeld etabliert und fokussiert sich auf die Kollaboration in Schlüsselbereichen der Wissenschaft mit Hilfe von Daten- und Computer-Infrastrukturen (sogenannte e-Science Infrastrukturen) um das Potential des wissenschaftlichen Rechnens zu erweitern. Vielversprechende internationale und eine steigende Anzahl von inter-disziplinarer Forschung wird mehr und mehr von globalen Kollaborationen, welche oft Ressourcen innerhalb einer e-Science Infrastruktur teilen, durchgeführt. In den letzten Jahren hat die ansteigende Komplexität von e-Science Anwendungen, die verschiedene physikalische Modelle nutzen und hoch skalierend sind, den Bedarf an weltweit interoperablen Infrastrukturen noch gesteigert um neue innovative e-Science Anwendungen zu erstellen die den gleichzeitigen Zugriff auf unterschiedliche e-Science Infrastrukturen ausnutzen. Allerdings sind interoperable e-Science Infrastrukturen heutzutage kaum vorhanden oder nicht einfach zugänglich und diese Doktorarbeit begründet diese Situation durch das Fehlen eines auf wissenschaftliche Produktion ausgerichtetes Referenzmodell für e-Science Infrastrukturen. Das Ziel dieser Doktorarbeit ist daher die Definition eines sogenannten "Infrastructure Interoperability Reference Models (IIRM)", welches auf die wissenschaftliche Produktion zugeschnitten ist und durch weniger Komplexität sowie geringerem Umfang als die "Open Grid Services Architecture (OGSA)" auch einfacher zu implementieren ist. Das vorgestellte Referenzmodell basiert auf vielzähligen Erfahrungen, welche mit e-Science Anwendungen aus den wissenschaftlichen Bereichen der Bio-Informatik, "e-Health" und der Fusionsenergie gewonnen wurden wobei alle diese Forschungsbereiche einen Zugang zu interoperablen e-Science Infrastrukturen ausnutzen können um ihre Forschung schneller voranzubringen. Ergänzend zu dem Referenzmodell wird ein sieben Segment-basierter Prozess vorgestellt der wichtige langfristige Probleme angeht in den Bereichen der Nachhaltigkeit von Interoperabilität, harmonisierten Infrastruktur-Operationen, Kooperation, Standardisierung, sowie gemeinsamen Betriebskonzepten und Entwicklungsplänen.

Acknowledgements

Over the past few years, the doctoral studies leading to this PhD thesis have been primarily performed at the Jülich Supercomputing Centre (JSC) of Forschungszentrum Jülich in Germany. I would like to express my gratitude to the many people who have helped me directly or indirectly with this thesis and my doctoral studies in Computer Science at the Karlsruhe Institute of Technology and the JSC respectively.

First and foremost, I would like to thank Prof. Dr. Achim Streit and Prof. Dr. Dieter Kranzlmüller. This thesis would not have been possible without their invaluable advice and continuous support over many years. I also want to express my gratitude to Prof. Dr. Dr. Thomas Lippert, head of the JSC, for giving me the opportunity to pursue my doctoral studies in an extraordinary and internationally renowned institute that laid the foundations for this work. I would like to acknowledge the efforts of my research group 'Interoperability and Applications' at JSC, namely Ahmed Shiraz Memon, Mohammad Shahbaz Memon, Sonja Holl, and Daniel Mallmann. Many of their development efforts of prototypes, planned concepts, and (partly even unstable) emerging open standards have led to many insights during my doctoral studies underpinning theoretical academic analysis with practical implementations.

I am also deeply indebted to Prof. Dr. Felix Wolf, University of Aachen, for his guidance and support of my doctoral studies. His suggestions, ideas and comments were more than helpful to me. Thanks to Dr. Bernd Schuller, Dr. Roger Menday, Dr. David Snelling, Dr. Thomas Eickermann, Bastian Demuth, Wolfgang Frings, and Mathilde Romberg for their support, ideas, and many technical discussions over the years. Thanks also to the members of the WISDOM initiative namely Vinod Kasam, Jean Salzemann, and Nicolas Jacq, and the many colleagues I have worked with during this scientific case study. I extend hearty thanks to our partners in the VPH case study, namely Prof. Dr. Peter Coveney, Steven Manos, and Stefan Zasada. Many thanks also to my collaborators from the EUFORIA project, most notably, Marcin Plociennik, Isabel Campos, and Dawid Seijnfeld. I would also like to express my appreciation to all the partners of the UniGrids, OMII-Europe, DEISA2, EMI, and XSEDE projects.

Above all, I would like to express my deepest gratitude to the people I love to whom I would like to dedicate this thesis: my parents Horst and Christa, who have aided me throughout my doctoral studies, my lovely fiance Elin Sif Kjartansdottir, and my grandparents Günter and Margot. *I don't know much about heaven, but I do believe in angels and thus I believe that my mother in some form or another is able to see the results of this PhD thesis.* Finally, I would like to offer my grateful thanks to all my international and JSC colleagues, and my dear friends, especially Renate Dornfeld and my Sauerland friends, who all believed that my studies would lead to fruitful results. I truly hope that this PhD thesis fulfills their expectations.

Morris Riedel
September 2012, Cologne, Germany

Contents

List of Figures

List of Tables

List of Definitions

Chapter 1

Introduction

Computational simulations and thus scientific computing is well accepted alongside theory and experiment in todays science. The term *e-Science* [19] evolved as a new research field that focuses on collaboration in key areas of science using so-called next generation data and computing infrastructures (i.e. *e-Science infrastructures*) to extend the potential of scientific computing. A wide variety of e-Science applications [301, 267, 206] take already advantage of various e-Science infrastructures that evolved over the last couple of years to production research environments in the sense of being successfully used as a tool by scientists on a daily basis.

The initial reference model of such infrastructures was coined as the *Open Grid Services Architecture (OGSA)* originally defined by Foster et al. in 2003 [171] and due to its slow adoption rate OGSA did not yet achieved a common basis for them. More recently the increasing demand of interoperability of numerous production e-Science infrastructures that have not adopted a common reference model like OGSA have found to be much more problematic than originally expected. End-users are often not able to understand the technical differences of such infrastructures and requesting their *seamless interoperable use* that enable stable e-Science applications across multiple infrastructures. But this interoperability does not broadly exist today although it bears the potential to tackle *grand challenge problems* of science and society today. Treating Aids or Alzheimers diseases, the localization of cancer, accurate global weather prediction, or determining the age of the earth, are those grand challenge problems where single e-Science infrastructures or large-scale High Performance Computing (HPC) resources have not made yet a significant breakthrough to either stop or fully understand them. Lacking interoperable e-Science infrastructures, their full potential of being jointly used can be not unleashed in order to tackle the aforementioned grand challenges and to make progress in research leading to significant scientific breakthroughs and innovation. Lacking these breakthroughs one needs to question the approach of e-Science infrastructures of whether it generates enough revenue to tax payers that indirectly financially support daily infrastructure operations.

Although OGSA represents a good architectural blueprint for e-Science infrastructures in general, this thesis argues that the scope of OGSA is too broad to be well focussed on existing production needs of e-Science infrastructures today. Two reasons for this are identified through investigations in technologies that aim to adopt OGSA. First, the process of developing the necessary high amounts of open standards that are conform to the large OGSA ecosystem takes rather long, including the precise specification of all its required service interfaces and their adoption by the respective technology providers. Second, the launch of OGSA-conform components within production e-Science infrastructures consumes substantial time after being evaluated for production usage having also no real aligned process of how to sustain interoperability and to improve them when the OGSA-based standardization groups are inactive after reaching their specification goals.

The *absence of a production-oriented and community accepted reference model* is diametral to the fundamental design principles of software engineering and has thus lead to numerous different non-interoperable architectures of production e-Science infrastructures in the last decade. One example of such an infrastructure is the Enabling Grids for e-Science (EGEE) [179] infrastructure, which recently concluded its process of being transformed into the European Grid Infrastructure (EGI) [25] but still using essentially the gLite middleware [212] in production. Another example is the Distributed European Infrastructure for Supercomputing Applications (DEISA) [184] which uses the UNICORE middleware [293] for the common access to computational resources in production since many years. Other infrastructures of this kind in the US are the TeraGrid [145] infrastructure, which uses the Globus middleware [167], but is evolving towards the Extreme Science and Engineering Discovery Environment (XSEDE) [29]. The Open Science Grid (OSG) [248] is also US-based and uses the Virtual Data Toolkit (VDT) [100] in production since many years. Yet another e-Science infrastructure was established in the nordic countries under the umbrella of the Nordic DataGrid Federation (NDGF) [61] that essentially uses the Advance Resource Connector (ARC) system [160]. Unfortunately, the aforementioned deployed technologies and their infrastructures are essentially not interoperable, mainly because of a *limited adoption of a common reference model in the last decade and being unable to agree on a common process of how interoperability can be established and then sustained*.

This *lack of interoperability is a hindrance* since there is a growing interest in the coordinated use of more than one infrastructures from a single client that controls interoperable components in different e-Science infrastructures. A classification of different approaches is provided in [268] and describes how e-Science infrastructures are used by scientists. Among simple scripts with limited control functionality (i.e. loops), scientific application plug-ins, complex workflows, and interactive access, there is also *'infrastructure interoperability'* mentioned as one approach to perform e-Science. A growing number of end-users (i.e. e-Scientists) would like to *benefit from interoperable infrastructures by having seamless access with their preferred tools to a wide variety of services and different underlying resource types.* This thesis reveal that many of these e-Scientists raise the demand to jointly access both High Throughput Computing (HTC)-driven infrastructures (e.g. EGEE/EGI, OSG) and High Performance Computing (HPC)-driven infrastructures (e.g. DEISA/PRACE, TeraGrid/XSEDE) from a single client (e.g. portal, desktop tool, etc.). The fundamental difference between HPC and HTC is that HPC resources (e.g. supercomputers, large clusters, etc.) provide a good interconnection of cpus/cores while HTC resources (i.e. pc-pools) do not. The e-Scientists also require seamless interoperability between access technologies that manage different types of computations and those that manage data stored in various resources used for computational simulations.

One goal of the aforementioned OGSA is to facilitate the interoperability of e-Science infrastructures, but in order to achieve interoperable e-Science infrastructures a reference model like OGSA needs to be specified much more precisely. This problem is addressed with the major contribution of this thesis that provides a reference model with an associated architectural blueprint that is much more detailed based on an academic analysis of lessons learned gained from production interoperability applications performed with different production e-Science infrastructures. A *detailed blueprint with a relatively abstract reference model is provided including an associated reference architecture with necessary building blocks to implement concrete instances of it that is all collectively named as Infrastructure Interoperability Reference Model (IIRM)*. In contrast to OGSA, the IIRM is much more focussed on *enabling production e-Science infrastructure interoperability* and the thesis describes necessary entities and relationships using derived architectures in applied research. A *complementary seven segment-based process of how interoperability can be theoretically established and sustained in production* is provided and *accompanying case studies demonstrate the impact of the proposed reference model.*

1.1 Terminologies

The initial introduction elements have already introduced key terms that we need to define more clearly at beginning of this thesis. These terms have similar meanings in different communities, but even can be differently understood within the same community.

The term *'interoperability'* needs to be precisely defined since otherwise this term can lead to various interpretations in this thesis. But the difficulty to define interoperability is best reflected in the fact that even IEEE had four definitions of interoperability in 2000 [200] at the time when the work around e-Science infrastructures largely began. According to IEEE [200], interoperability stands for:

- the ability of two or more systems or elements to exchange information and to use the information that has been exchanged.
- the capability for units of equipment to work together to do useful functions.
- the capability, promoted but not guaranteed by joint conformance with a given set of standards, that enables heterogenous equipment, generally built by various vendors, to work together in a network environment.
- the ability of two or more systems or components to exchange information in a heterogenous network and use that information.

A broad agreement on an exact definition of this term is non-trivial, because of the different contexts and expectations in a time where computer science is constantly changing. New functions and capabilities of systems influences this definition that in turn is another reason for the creation of a wide variety of community-specific definitions of this term. A community-specific definition relevant in this thesis is provided in [256], but for the purpose of this thesis extended with some aspects raised by IEEE as follows:

Definition 1 (Interoperability) *Interoperability is the native ability of e-Science technologies and infrastructures to exchange, understand, and use information directly via common open standard-based message exchanges within a network of interoperable services.*

Another often used term in context is *'interoperation'* that is often used as a synonym for interoperability, but is in this thesis specifically defined as the community-specific definition given in [256] highlighting its difference to 'interoperability'. Also the term 'interoperation' can have various meanings, but within this thesis the community-agreed definition is as follows:

Definition 2 (Interoperation) *Interoperation is what needs to be done to get production e-Science infrastructures to work together as a fast short-term achievement using as much existing tuned (i.e. hacks, workarounds, tweaks, etc.) technologies as available today.*

This definition is very different than 'interoperability', which can be seen as the 'perfect solution', in order to promote the understanding of interoperability issues and challenges by not neglecting important details during operations accross infrastructures. Interoperation is not a perfect solution and relies on workarounds, tweaks or hacks of technologies to get different technologies or infrastructures to interoperate as a short-term achievement. Throughout this thesis, the case of true interoperability is often discussed as rather long-term achievement and reasons why early interoperation success stories have to use 'unconvential' methods to achieve short-term achievements are provided in context.

Finally, some terms indicate a transition phase of infrastructures, e.g. EGEE/EGI means that during the time of the thesis studies a transformation of the infrastructure from EGEE to EGI took place. The same phrasing of terms is also used for other e-Science infrastructure transformation in this thesis.

1.2 Thesis Objectives and Contributions

The problem of e-Science infrastructure interoperability is *highly underestimated since it is related to issues that are part of applied research rather than pure academic research.* With its applied nature, the thesis contribution enables improvements in *quality* since interoperability is not any more build on small hacks, tweaks, or workarounds within technologies thus delivering significant value to end-users with solutions in a more stable way than before. This thesis offers also improvements in *quantity*, because specific instances of the proposed IIRM are more easier used in use cases compared to typical pair-wise infrastructure interoperability setups.

The academic studies start by *reducing the scope* from being relevant to all existing e-Science technologies to only those that are majorly important to perform e-research with *state-of-the-art production e-Science infrastructures*. In contrast to OGSA this thesis thus does not attempt to address all problems arising in e-Science infrastructures, but focuses on the significant basic aspects that enable e-Science infrastructure interoperability towards production usage by end-users. In other words, the reference model does not aim to replace OGSA but rather trims it down in functionality by dropping several less important parts of it and refining other parts that are mostly relevant to interoperability of production e-Science infrastructures today.

In a second step, the thesis *evaluates, extends, and enhances existing use-case oriented standards-based approaches* with aspects originating from practical field experiences obtained via production applications across infrastructures. Several contributions of this thesis are based on an academic analysis of lessons learned gained from scientific applications requiring resources in more than one e-Science infrastructure.

Apart from these initial studies [256], the three accompanying academic case studies contributed to the lessons learned and play a significant role in the verification and validation of the significance, impact, and applicability of the IIRM. These three case studies are in-silico drug discovery in the bio-informatics domain (i.e. WISDOM [259]), the human body as a single system (i.e. VPH [263]), and several codes of the fusion community (i.e. EUFORIA [225]) and all of them contributed to the experience of using emerging open standards in IIRM derived production architectures of infrastructures. In order to take this experience obtained over years into account, the thesis contribution is based on open standards that are in turn refined to add certain investigated previously missing concepts. The thesis thus provides feedback to the standardization process with its findings.

The third step that is remarkable in this thesis is the *creation and implementation of a whole process of how interoperability can be achieved with e-Science infrastructures in a sustainable way today.* This method is complementary to the design of the reference model and with this goes beyond a typical interoperability framework or architecture like the OGSA.

In addition to this rather new method, we also address so-called *'missing links'* with a new way of increasing the effectiveness of 'standardization' by linking the specifications of different technical areas where possible. Standardization groups consider challenges of their own specific area and perform often standardization in isolation from each other. In contrast, the contribution of this thesis tries to gain benefit from synergies that arise by using standards from different areas such as information, data, compute, and most notably security together. With this new way of *'performing standardization' as an inter-disciplinary research* a reference model is created that goes far beyond the functionality of a set of standards-compliant architectures that would only enable partial interoperability in production e-Science infrastructures today.

In a fourth step, *the typical orientation of reference models in e-Science is changed* from intra-Grid architectures (i.e. one technology) into a model that is specifically designed to enable interoperability across infrastructures (i.e. multiple architectures and technologies). In contrast to OGSA, which aims to design one technology architecture, the contribution in this thesis focus on functionality required to enable interoperability between different types of technology architectures by changing the focus to their functionality overlaps and as such on *'infrastructure-oriented architectures'*. This reference model foundation enable algorithms that take advantage of both HTC and HPC computational paradigms in one scientific workflow leading to the definition and application of a so-called *'e-Science design pattern'*. Similiar models in e-Science and e-business only address component-level interoperability aspects such as concrete functionality and semantics.

In contrast, the whole process view of reaching production e-Science infrastructure interoperability is changed in this thesis towards a concrete seven segment-based process of how to achieve and sustain it. True production e-Science infrastructure interoperability can be only achieved when operational constraints as well as policy restrictions are taken into account that are both part of this proposed process. The thesis contributions contain real production e-Science infrastructure impacts being achieved by an active implementation over years of the proposed process as another key contribution of this thesis. The reference model enables technical infrastructure interoperability, but also offers guidance on these operational and policy aspects that are necessary to sustain interoperability.

All these steps enable the formulation of the major research question of this thesis that is *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. Focusing on this general research question, but not losing sight of general standards-based methods and software engineering reference model basics, the research challenges, thesis objectives, and contributions of this thesis can be summarized as follows:

- First, the thesis investigates an approach for a reference model that represents a trimmed down version of OGSA in terms of functionality and complexity, while on the other hand being more production-oriented and thus easier to implement. Being applicable in production e-Science infrastructures that cope with change and policy constraints, the thesis offers also a complementary process of establishing and maintaining their interoperability.

- Second, the thesis integrates valuable field experience and lessons learned from production e-Science infrastructure interoperability into the IIRM as well as the seven segment-based process. The reference model consists of core building blocks that are based on early versions of refined open standards thus indirectly also providing feedback to standardization activities. This is particularly the case with several specifications such as the Job Submission and Description Language (JSDL) [115] that have been traditionally more used in HTC-oriented infrastructures instead of HPC-oriented ones.

- Third, the thesis aims to fill *'missing links'* between open standards by applying an interdisciplinary research approach addressing the areas of information, data, compute and security together. The IIRM aligned infrastructure interoperability process provides a new method in the e-Science architectural framework landscape thus realizing a reference model design that is also able to respond to *'dynamic changes'*.

- Fourth, this thesis changes the foundation of reference models in e-Science by providing concepts across infrastructures and a whole interoperability process instead of a purely intra-Grid architecture like OGSA does. A new form of algorithm is defined as an e-Science design pattern that refers to the joint use of HTC-based and HPC-driven e-Science infrastructures. The IIRM and associated architecture work changes the approach of a technology-oriented architecture to an infrastructure-oriented architecture.

- Fifth, the significance of the IIRM is shown with applied research with the three accompanying academic case studies WISDOM, VPH, and EUFORIA. The process implementation lead to a production relevant reference model design and its applications verify that the IIRM achieves real impact on production e-Science infrastructures.

1.3 Related Topics Out of Thesis Scope

Production e-Science infrastructures encompass a large field of technologies, organizations, and approaches that all in some form or another need to be inherently interoperable. This interoperability needs to be achieved on different levels leading to technical, semantic, legal, and ethical issues. This thesis tackles mostly technical and partly semantic issues, while related legal issues (e.g. work with patient-specific data) or ethical issues are out of scope of this thesis.

But this thesis focusses on the computational functionalities and only related data storage and transfer capabilities (e.g. data-staging [115]) for existing e-Science infrastructures with a particular focus to support production e-Science applications. In terms of the reference model and its associated architecture work the focus relies on core building blocks to support computational-intensive e-Science applications with processing functionality and related data-staging capabilities. The thesis is thus mostly covering topics in compute oriented e-Science infrastructures and does not enter the complex field emerging under the umbrella term 'big data'. Out of scope of this thesis are thus advanced data concepts that offer 'logical and physical data mappings' such as those from data catalogues sometimes used in conjunction with data storage systems (e.g. dCache [178]). This topic and its interoperability and synchronization is too complex to be tackled as part of this thesis. Also advanced functionality and requirements of large data infrastructures such as within the European Data Infrastructure (EUDAT) [22] are out of scope thus not addressing persistent identifiers [28], or policy-based data management like with iRods [229] for example. Although some parts of the proposed thesis elements might reach into these infrastructures it is considered out of scope. Another key related topic is the underlying network infrastructure where this thesis is build on like GEANT [1] in Europe for example. Although this thesis significantly relies on network capabilities, the thesis does not enter this complex technical field and assumes network connectivity as given.

In more detail, several important topics related to production e-Science infrastructures are out of scope of this thesis that are the following. The thesis does not tackle any education and training of e-Infrastructures as tackled by the e-Infrastructure Reflection Group (e-IRG) Education and Training Task Force (ETTF) [206]. Ticketing systems, help desks, and general support structures are another essential part of production e-Science infrastructures. Also these topics are kept out of this thesis although the interoperability of different ticketing systems is a concern today.

The objectives of this thesis also do not tackle any issues about low-level accounting, billing frameworks, pricing models, resource trading, and other related aspects that are commercially-driven and not directly relevant for e-Science infrastructures. Although we address the tracking of resource usage with a certain format, we do not provide details about its provisioning and distribution apart from the fact that it should be used together with messaging-based systems or via the use of information systems where applicable.

Another related field the thesis studies have been software licenses used in e-Science endeavors that are not directly usable in a shared fashion. Examples are those purchased by research institutes like FlexX [30] or GAUSSIAN [35], MATLAB [298], being rather expensive and that do not directly support a broad sharing of purchased licenses within e-Science infrastructures today. In some cases, special granted usage has been agreed with the software providers like within the case of FlexX that offered several free licenses for the work within WISDOM related to this thesis. Such (dynamic) licensing models are interesting and required for a broad usage, but are kept out of this thesis to remain focus on the rather technical aspects. Some aspects of such discussions are part of policy work that in turn is considered at least partly within the seven segment-based process of this thesis. For a more formal sound dynamic approach and valid methods, the SMARTLM EU project [90] provides more information and particularly works on exactly the aforementioned license problems.

Other fields of interest in context that are out of scope of this thesis are Service Level Agreements (SLAs) or issues of co-allocation and co-scheduling of compute resources that are related to this topic. Also not directly covered are workflows that use, for instance, workflow languages like the WS-Business Process and Execution Language (WS-BPEL) [110] often used in commercial setups. In this research field, the BISGrid EU project [2] provides useful pieces of work that can be used together with the proposed reference architecture core building blocks. We can also state that workflow engines in general might consider the IIRM-guided reference architecture as an opportunity to perform basic executions on e-Science infrastructures through open standards while the general handling of Directed Acyclic Graphs (DAGs) or other forms of workflows have been kept out for clarity.

Although this thesis briefly tackles the compilation problem of typical jobs submitted to computing resources, a full blown compile-debug interactive solution or related aspects of computational steering are also out of scope of this thesis. Those approaches often require some form of interactive channel to the applications that essentially require full frameworks (e.g. like being part of computational steering approaches [258]) that establishes and maintains interactive bi-directional channels in e-Science infrastructures. Such approaches have been out of the scope of this thesis to remain focus and a broader adoption of the reference model, but the work performed around the IBM Parallel Tools Platform (PTP) [74] is suitable in context of this thesis.

In production e-Science infrastructures it is important that services are properly monitored with various kinds of technologies (e.g. NAGIOS [126], etc.) but also this topic is out of scope while we indirectly provide its foundation via information provisioning in a dedicated information model (i.e. GLUE [113]). Monitoring is another topic for 'interoperability of production e-Science infrastructures', the focus however is set on partly enabling functionalities (i.e. information models) and not on monitoring systems and their information sources (e.g. NAGIOS probes [126]). Although the topic of an information model is covered in this thesis, it does not tackle the plethora of different services and approaches available in the field of information services and service registries. The wide variety of existing solutions in the field is too diverse to be used in the thesis and a standardisation of these interfaces have been tried for years without a common success. In the most cases, this thesis thus refers to mechanisms out of scope of this thesis such as information systems based on the Lightweight Directory Access Protocol (LDAP) [196] in context where required. More important is that the information retrieved by this service follows a common schema with information that in turn is in scope of this thesis.

In many cases, the existing standard specifications used in this thesis have been defined with a focus on supporting HTC-driven e-Science infrastructures and do not focus on HPC-driven e-Science infrastructure demands. This thesis therefore focusses on many concepts that extend the potential of existing specifications towards HPC environments while other HTC-oriented topics such as brokering are left out for clarity. But it should be noted that a full normative specification of these improvements is also out of scope of this thesis and needs to be provided from relevant Standard Development Organizations (SDOs) like the Open Grid Forum (OGF) [68] and existing specifications are references where possible. In this sense, this thesis rather suggests certain concepts and approaches that have been given as a unique contribution to the standardization process still ongoing within the Production Grid Infrastructure (PGI) working group of OGF to influence the next iterations of emerging standards like JSDL [115], GLUE2 [113], or OGSA-Basic Execution Service (OGSA-BES) [169].

1.4 Selected Publications and Demonstrations

All major parts of this thesis are published in peer-reviewed conference proceedings, scientific journals, book chapters, and magazine articles. The major journal publications and contributions to book chapters and magazines are listed in Table 1.1 while peer-reviewed conference contributions are listed as part of Table 1.2. Many of these publications have been co-authored with a many co-workers on technical or use case topics over the years. Both tables provide information in the context of their unique contribution to this thesis while evidence supporting these contributions is provided in Chapter 5 and 6.

All in all, these publication proof that the contributions are accepted by the e-Science community via the peer-review procedure. These publications also proof that the contributions have been found before any standardization community work has started and thus are able to claim that all the work around the reference model is truly uniquely the contribution of this thesis based on academic work over years. The publications over the years clearly illustrate that early findings have been published continously along the way of the academic process of creating the IIRM and its associated architecture work and validating it with production e-Science applications on existing production e-Science infrastructures.

Complementary to the academic studies, many real prototypes as well as application case studies in the context of real production e-Science infrastructures have been used to explore findings that arise in production usage of technologies. In order to proof that the work is rel-

Ref.	Publication Title	Published in	Thesis Material
[256]	Interoperation of World-wide Production e-Science Infrastructures	Concurrency and Computation Practice and Experience, No.21 Wiley, 2009	Interoperation prototypes; documented lessons learned; infrastructure standards analysis;
[272]	Advances by using Interoperable e-Science Infrastructures - The Infrastructure Interoperability Reference Model applied in e-Science	Journal of Cluster Computing Vol.12 (4) Springer, 2009	IIRM architectural design; Case studies WISDOM and VPH; Interoperability Approach Classification;
[284]	Recent Advances in the UNICORE 6 Middleware	inSiDE Magazine, Vol.8 GCS,2010	UNICORE and standard adoptions;
[270]	OGF Production Grid Infrastructure: Use Case Collection - Version 1.0	OGF GFD 180 2011	Our PGI evidence on OGF standard refinements
[254]	e-Science Infrastructure Interoperability Guide - The Seven Steps towards Interoperability for e-Science	Book Chapter in Guide to e-Science, Springer, 2011	Seven segment-based process for e-Science interoperability;
[235]	UNICORE in XSEDE: Towards a Large-Scale Scientific Environment based on Open Standards	inSiDE Magazine, Vol. 9 GCS ,2011	Open Standards in XSEDE; XSEDE architecture

Table 1.1: Relevant contributions to books, magazines, and journal publications.

evant for the e-Science community in general and for production e-Science infrastructures in particular, many demonstrations at events have been performed. The work around the WISDOM case study [259] was presented at the Supercomputing conference 2007 in Reno at the JSC booth [94]. While continuing the work around WISDOM, another case study named as VPH

Ref.	Publication Title	Published in	Thesis Material
[262]	A DRMAA-based Target System Interface Framework for UNICORE	ICPADS 2006	UNICORE DRMAA adoption lessons learned; DRMAA HPC experience;
[177]	LLView: User-Level Monitoring in Computational Grids and e-Science Infrastructures	GES 2007	UNICORE UR and RUS adoption lessons learned; UR HPC refinements;
[302]	Using SAML-based VOMS for Authorization within Web Services-based UNICORE Grids	Europar 2007	OGSA-BES and SAML prototypes; SAML-based VOMS;
[220]	Open Standards-based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE	e-Science 2007	OGSA-BES lessons learned; gLite and UNICORE interoperability; security plumbings;
[267]	Experiences and Requirements for Interoperability between HTC- and HPC-driven e-Science Infrastructures	Korean AHM 2008	Trimmed down OGSA idea of a reference model;
[191]	Benchmarking of Integrated OGSA-BES with the Grid Middleware	EuroPar 2008	OGSA-BES benchmarks and lessons learned; UNICORE proxies;
[259]	Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA	IEEE MIPRO 2008	WISDOM Use Case applications; IIRM instance for bio-med applications;
[268]	Classification of Different Approaches for e-Science Applications in Next Generation Computing Infrastructures	e-Science 2008	HTC and HPC interoperability identified as being crucial for end-users;
[265]	Concepts and Design of an Interoperability Reference Model for Scientific- and Grid Computing Infrastructures	ACC 2009	Manual data-staging concept;
[224]	Enabling Grid Interoperability by Extending HPC-driven Job Management with an Open Standard Information Model	ICIS 2009	OGSA-BES and GLUE2 prototypes;
[193]	Life Science Application Support in an Interoperable E-Science Environment	CBMS 2009	WISDOM and sequence concept; AMBER environments;
[261]	Improvements of Common Open Grid Standards to Increase High Throughput and High Performance Computing Effectiveness on Large-scale Grid and e-Science Infrastructures	HPGC 2010	JSDL refinements; JSDL with GLUE2 elements; e-Science application concepts; common execution environments concept;
[263]	Exploring the Potential of Using Multiple e-Science Infrastructures with Emerging Open Standards-based e-Health Research Tools	CCGrid 2010	VPH Use Case applications; IIRM instance for e-Health applications;
[266]	Towards Individually Formed Computing Infrastructures with High Throughput and High Performance Computing Resources of Large-scale Grid and e-Science Infrastructures	MIPRO 2010	plumbings concept; benefits of network of interoperable services;
[225]	Lessons learned from jointly using HTC- and HPC-driven e-science infrastructures in Fusion Science	ICIET 2010	EUFORIA applications; IIRM instance for fusion applications;
[184]	DEISA: e-Science in a Collaborative, Secure, Interoperable and User-Friendly Environment	e-Challenges 2010	DEISA interoperability work on standards and applications;
[264]	Requirements of an e-Science Infrastructure Interoperability Reference Model	MIPRO 2011	critical OGSA analysis; indicators and factors as reference model criteria;
[260]	e-Science Infrastructure Integration Invariants to Enable HTC and HPC Interoperability Applications	HPGC 2011	Production Infrastructure Integration Invariants and IIRM Constraints;

Table 1.2: Conference proceedings publications being directly or indirectly related to the thesis.

[263] emerged that was demonstrated together with VPH e-Scientists at the Supercomputing Conference 2008 in Austin at the JSC booth [95]. The contributions around the third case study named as EUFORIA [225] have been demonstrated together with EUFORIA project members at the Supercomputing Conference 2009 in Portland at the JSC booth [96].

The author also gained considerable profile in terms of e-Science infrastructure interoperability by not only chairing the renowned OGF Grid Interoperation Now (GIN) [256] group over years, but also by conducting several workshops on the topic including being member of various topic-related program committees. The International Grid Interoperability and Interoperation Workshop (IGIIW) 2007 at the e-Science conference 2007 in Bangalore [46] was organized by the author. The IGIIW 2008 that was held in conjunction with the e-Science conference 2008 in Indianapolis [47] was also organized by the author. A workshop in 2009 was not planned in favour of editing a special issue 'Grid Interoperability' [269] in the Journal of Grid Computing together with G. Terstyanszky. In 2010 another event in the specific field named as the Distributed Computing Infrastructure Interoperability Minisymposium [12] at the PARA2010 in Reykjavik was organized by the author.

1.5 Thesis Structure

Chapter 1 of this thesis introduces the problem domain and highlights the major thesis contributions that overcome several limitations in production e-Science infrastructure interoperability. As part of the introduction the terms 'interoperability' and 'interoperation' are defined, because they often lead to confusion when used in the same context. Chapter 1 also gives insights which technically interesting related topics are kept out of this thesis in order to preserve a clear and well-formed focus of the academic studies.

After the introduction in Chapter 1, the scene is set in Chapter 2 with a survey of state-of-the-art e-Science infrastructures and their demands for sustainable interoperability motivated by e-Science applications. This chapter introduces fundamentals from the scientific field such as e-Science itself, middleware technologies, or open standards that are all relevant to the scientific investigations of this thesis. It defines essential terms (e.g. computational Grids) and wordings (e.g. resource sharing) that are used throughout the thesis and brings clarity to several concepts that are sometimes aligned with different understandings.

Chapter 3 surveys existing work in the field of e-Science interoperability and provides certain factors and indicators how solutions that tackle interoperability can be verified. A deeper survey is provided for component-based approaches that tackle interoperability in many cases on a rather short-term pair-wise fashion that have all contributed significantly to the 'lessons learned' incorporated in the results of this thesis. Complementary to the aforementioned survey, we analyzed several existing reference models in order to understand if those might be a good basis to start from heading towards more preferred long-term solutions guided by reference models and associated architecture work that are considered as such. Conclusions of this particular chapter majorly influence the requirement setup as well as the design of the proposed reference model and its associated elements.

Based on the aforementioned lessons learned and the survey of related work in the field, Chapter 4 defines reference model requirements that satisfy the end-user requirements for interoperable production e-Science infrastructures. These requirement setup forms the foundation for the definition of the functionality that the core building blocks of the reference model design needs to provide.

The major contributions of this thesis are presented in Chapter 5 addressing the requirements of Chapter 4 thus providing an abstract model that is augmented with a guiding process of how such a model can lead to sustained interoperablity of e-Science infrastructures. The chapter gives insights into the proposed seven segment-based process to establish and sustain interoperability as a long-term achievement. Being essentially the first part of this process, the proposed reference model named as the Infrastructure Interoperability Reference Model (IIRM) is presented in detail.

Based on the abstract reference model and the seven segment-based process, concrete implementations can be formed out of them in Chapter 6 that are related to existing open standards and have a focus on the adoption of the model in the context of existing production infrastructures. The underlying research problem and its solution have practical consequences with 'applied research' contributions providing an impact that is verified via the three accompanying case studies (WISDOM, VPH, and EUFORIA). The experience to refine standard-based reference model entities to enable required concepts are outcomes of rather academic work based on investigating lessons learned gained by production experience from real e-Science applications over the last years. This is equally valid for the seven segment-based process towards sustained interoperability that complements the findings of the reference model itself. Both complementary approaches bear the potential to overcome many interoperability challenges stated in Chapter 2 and Chapter 3.

Finally, Chapter 7 provides conclusions of this thesis and offers interesting aspects for future work that can be based directly or indirectly on the findings of the academic studies of this thesis.

Chapter 2

State-of-the-art e-Science Infrastructures

The last chapter contextualises the scientific field that is commonly known as *'Grid computing'*, or more recently *'e-Science'*, with a focus on infrastructures and their technologies. The problem statement of the overall thesis research is introduced and the significance of a potential response to this problem is motivated. This thesis essentially tackles one particular research question that is, *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. But before suitable solutions to this research question are investigated during the course of this thesis, many aspects of the current practice in the Grid community are introduced, including a precise definition of various used terms.

This chapter defines and clarifies relevant terms as well as fundamentals of e-Science in order to lay a foundation for later chapters. Known sources are referenced where possible to keep the technical depth of each of the introductory topics to a reasonable level of understanding, required to follow thesis approaches. Hence, overviews are provided without too many technical details that can be looked up in other publications, technical documents, or even precise normative specifications.

The first section describes and defines key concepts of the state-of-the-art of production e-Science infrastructures. The motivation behind e-Science infrastructures is given and where they can be seen in the context of traditional scientific methods (e.g. theory, experimentation, etc.). Relevant e-Science infrastructure examples are described and several concepts that enable their *seamless sharing of resources* are briefly reviewed. Also, a classification of *e-Science applications* is given that demonstrates how e-Science infrastructures are used in daily science today.

A wide variety of technologies and open standards are important cornerstones of this thesis and therefore the second section reveals which particular set of them are important in this thesis. Overviews of numerous relevant technology concepts are introduced such as *resource management systems*, *middleware*, or *service-oriented technologies*. Real technological examples that play a significant role in production e-Science infrastructures are provided in context. Related to these technologies, important *common open standards* are described that are later used as a basis for the thesis reference model design.

The final section motivates more clearly investigations into solutions to the given fundamental research question. Along with a *classification of existing production e-Science infrastructures* that point to existing interoperability challenges, elements of the research question are precisely defined. All in all, a *model of the problem space* is created that goes beyond the problem statements presented in the introduction emphasizing on known limitations in achieving interoperability between production e-Science infrastructures today.

2.1 Grid and e-Science Infrastructure Concepts

This section introduces the basic concepts and ideas related to the term *'e-Science infrastructures'* like *'resource sharing'* and provides insights into how their end-users take advantage of them via so-called *'e-Science applications'*.

2.1.1 e-Science Infrastructure Fundamentals

In order to better understand the term e-Science, *computational science* is briefly reviewed and put it in the context of current *traditional science*. Scientists regard computational techniques and thus traditional scientific computing as a third pillar alongside experimentation and theory as shown in Figure 2.1. This does not mean that a theoretical and experimental scientist automatically becomes a computational scientist and thus not necessarily uses the illustrated e-Science infrastructures. Instead, the role of the computational scientists takes advantage of the findings of scientists from other pillars (i.e. certain theories) or proves experimental results (i.e. laboratory measurements) with computation thus combining these outcomes with the powers of computing. The roles of the different types of scientists are illustrated in Figure 2.1 in the context of the different pillars and their key contents.

The *first pillar* stands for a certain theory or specific model in a given research field. One example of this particular pillar is the outcome of scientists that use complex mathematical mod-

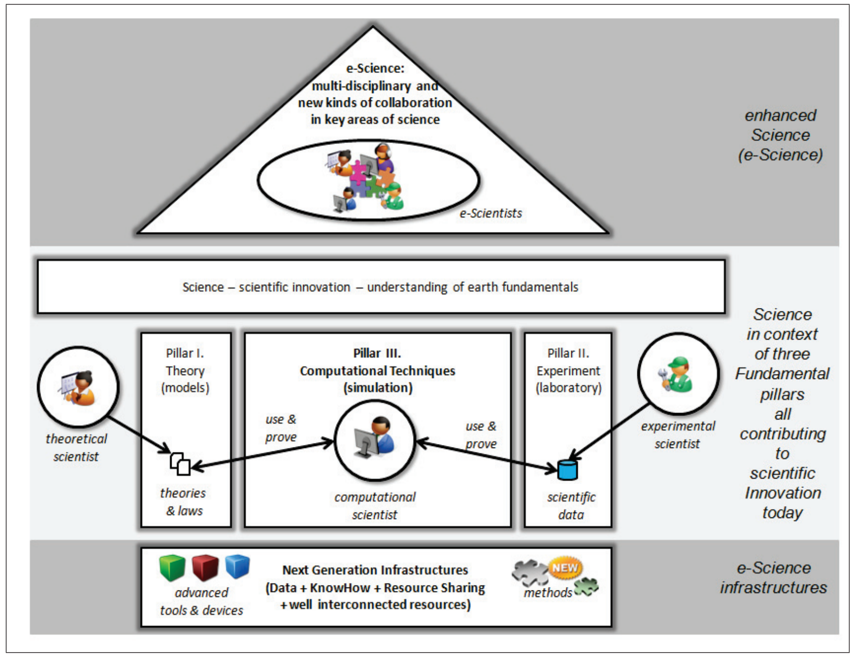


Figure 2.1: The three fundamental science pillars in context of e-Science.

els to predict the diffusion of harmful materials in soil. The *second pillar* points to experimental data taken in laboratories that use 'environments with certain conditions' or take advantage of technical support via 'specific instruments'. Examples of such experimental outcomes are measurements of aeroplane behaviour in air conducts or probes of material, for instance, of harmful materials in soil analysed via relevant instruments in a laboratory and their data results. The computational techniques in the *third pillar* allows for computer-simulations based on efficient numerical methods and known phyisical laws that is known as *traditional scientific computing* using elements of other pillars or proving them. One example are scientists that compute the flow of water underground and simulate the way in which various harmful substances react with potentially damaging consequences.

In addition to the aforementioned three pillars, the term *enhanced science (e-Science)*, sometimes also called *electronic science* has evolved in the last couple of years. A wide variety of different defintions and interpretations exist. The e-IRG describes 'e-Science' in the 2009 white paper as 'the invention and application of ICT-enabled methods to achieve new, better, faster or more efficient research, innovation, decision support or diagnosis in any discipline. It draws on advances in computing science, computation, and digital communications' [206]. Another interpretation of e-Science is given in the 7th EU concertation meeting report 2009: 'e-Science can be defined as science that may necessitate the utilisation of non-trivial amounts of computing resources and massive data sets to perform scientific enquiry; Science that requires access to remote scientific instruments and distributed software repositories; Science that generates data that may demand analysis from experts belonging to multiple organisations and are specialists in different knowledge domains - such Science is usually carried out in distributed environments...' [134]. The definition of e-Science used in this thesis is kept very simple and is based on the following definition given by John Taylor in [19]:

Definition 3 (e-Science) *e-Science is about global collaboration in key areas of science and the next generation infrastructure that will enable it.*

This definition has been often extended in several ways to include a particular focus or a dedicated technology. For instance, dynamic deployment features achieved in using *virtualisation techniques* within so-called clouds [176] as another form of next generation infrastructures have been added as another feature to this definition. An analysis by Foster et al. in [176] provides more information about clouds in context of e-Science.

The e-Science definition above is taken as a basis in this thesis, because it is the initial and mature definition of e-Science. When this definition is put in context to the traditional sciences, the aforementioned next generation infrastructures can be considered as a 'solid basement' for the three pillars. The infrastructure basement together with the elements of the three pillars enable e-Science that can in turn be seen as a roof, also shown for clarification in Figure 2.1. Hence, e-Science is about the collaboration in key areas of science (i.e. pillars) to extend the potential of traditional science with the help of next generation infrastructures. Collaboration in key areas of science can either be within a scientific discipline or across different scientific disciplines *sharing hardware, data, resources, approaches and knowledge*.

Over the years different names for such next generation infrastructures appeared. In the US these infrastructures are known as *cyber-infrastructures*, while in Europe they are named as *e-Infrastructures* [134] or *Grids*. More recently, the term *Distributed Computing Infrastructures (DCIs)* evolved that is not used in this thesis because of its lack of indicating the relevance of data in context. In the context of this thesis, next generation infrastructures are implemented using Grid concepts and are named e-Science infrastructures to reflect their scientifically-driven priorities. But all the different names share the same common methods that are based on several new *sharing techniques* used with traditional principles and paradigms of distributed systems [296]. The next generation infrastructures in this thesis are represented by *e-Science infrastructures* that are defined as follows:

Definition 4 (e-Science Infrastructure) *An e-Science infrastructure is a problem solving infrastructure that enables innovation through data, knowledge, and (dynamic) resource sharing via high performance interconnections and the provisioning of services that support this sharing.*

Examples of European e-Science infrastructures are EGEE/EGI, DEISA/PRACE or the NorduGrid infrastructure [157]. The EGEE/EGI infrastructure consists of numerous infrastructures managed by the National Grid Initiatives (NGIs) [58] that in turn also represent smaller e-Science infrastructures. *'In 2010 this infrastructure supports world class science in over 50 countries, consisting of about 300 sites, encompassing more than 150.000 processors, 25 petabytes of disk storage and 40 petabytes of long-term tape storage-enough to store 400 million four-drawer filing cabinets full of text'* [161]. In the US, known e-Science infrastructures that can be categorised according to this e-Science definition are TeraGrid/XSEDE and OSG. All these infrastructures are named as *'production e-Science infrastructures'* in order to emphasise on the fact that they are used by scientists to perform real science on a daily basis. They are different to non-production infrastructures that exist reaching from small e-Science testbeds to testing or pre-production infrastructures. These latter infrastructures are not used to perform e-Science but to test technology or deployment issues and are thus different from production e-Science infrastructures that are defined as follows:

Definition 5 (Production e-Science Infrastructure) *A production e-Science infrastructure is an infrastructure that is available on a 24/7 basis and used by scientists on a daily basis to perform e-Science.*

2.1.2 Resource Sharing in e-Science Infrastructures

The previous section mentioned sharing techniques that stand for a couple of concepts enabling *data, knowhow and resource sharing* being all relevant in e-Science. More details about resource sharing as one of the key concepts in e-Science infrastructures are given in this section. This concept enables solutions to scientific problems with having a more aggregated amount of resources. An e-Science infrastructure connects multiple smaller, regional, and national infrastructures together. Each consists of a set of resources that are considered to be *shared among the scientific communities*. Therefore, an *infrastructure resource* is defined as follows.

Definition 6 (Infrastructure Resource) *An infrastructure resource is any kind of resource (computational resource, data resource, large scientific device or instrument, etc.) as long as it provides a suitable interface for data and/or control exchange with e-Science infrastructures.*

Examples for such infrastructure resources are supercomputers, clusters, network devices, mass storage elements, large telescopes, high-end visualisation devices and magnetic resonance tomographs (MRT). The basic idea of an e-Science infrastructure is that such infrastructure resources are combined or shared among scientific communities to solve a specific scientific problem. Examples of such resources can be found in Figure 2.2. A more concrete example is the fundamental concept behind the *Worldwide Large Hadron Collider (LHC) Computing Grid (WLCG)* [104] that is one scientific application community that takes advantage of the e-Science infrastructure EGEE/EGI. *'The LHC is now fully online and the experiments will produce up to 15 petabytes of data per year (roughly 3 million DVDs or 20.000 years of music in MP3 format)* [161]. WLCG scientists analyse physical data on distributed computing clusters (infrastructure resource A) that are collected by a LHC high-energy physics experiment detector (infrastructure

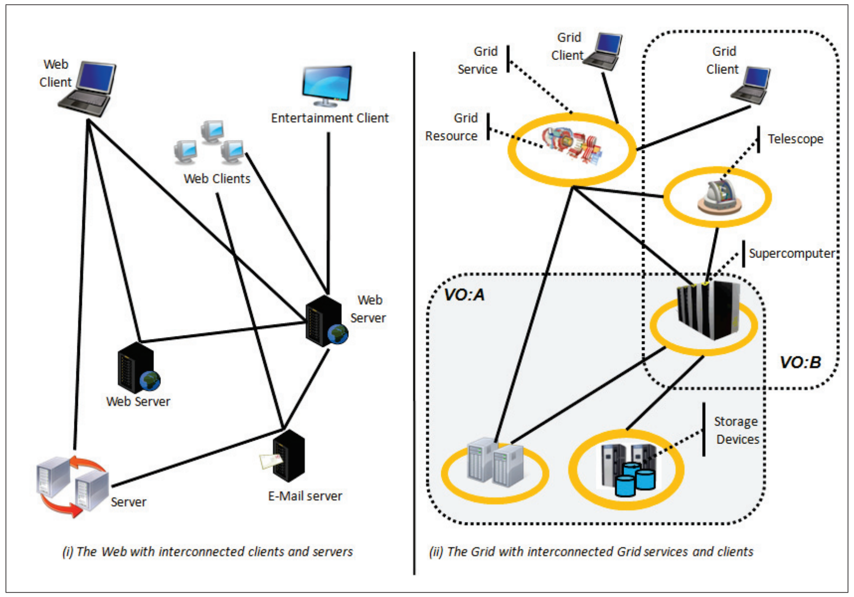


Figure 2.2: Examples of Web and Grid difference: Dynamic sharing in VO:A and VO:B of a supercomputer.

resource B) and then store results for later analysis in mass storage elements (infrastructure resource C). Powerful network devices and network connections (infrastructure resource D) are used for the transport of the data between the Grid Resources A, B and C. The overall benefit of this approach is the seamless integration of the aforementioned resources into one highly flexible, secure and reliable infrastructure representing a 'tool' for these scientists.

This example reveals the tight connection between e-Science infrastructures and so-called *'Grid infrastructures (aka Grids)'*. The introduced concept of e-Science infrastructures is very abstract in its definition and nature and a more concrete implementation of these abstract approach are Grids. e-IRG describes in the 2009 white paper Grid as follows: *'Grid is a system that federates, shares and coordinates distributed resources from different organisations which are not subject to centralized control, using open, general-purpose and in some cases standard protocols and interfaces to deliver non-trivial qualities of service. The Grid is used by Virtual Organisations, i.e. thematic groups of users crossing administrative and geographical boundaries'* [206].

In [172], Foster et al. introduced the Grid concept with an analogy to the electric power grid, in that it provides pervasive access to electricity and had a dramatic impact on human capabilities and society. In extended geographical regions, such as whole nations (e.g. Germany or greater regions such as north America), the power grid essentially forms a single entity that provides power to countless electrical devices. This is mostly done in an efficient and reliable fashion, while the power grid consists of thousands of different generators linked with billions of outlets via a very complex web of physical connections. Given this analogy and specific computing oriented e-Science infrastructure resources, the following definition of a Grid infrastructure can be defined:

Definition 7 (Grid Infrastructure) *A Grid infrastructure (aka Grid) integrates heterogeneous hardware and software components of geographically dispersed infrastructure resources that are owned by different organisations into one efficient and reliable network of connections. As a concrete implementation of an e-Science infrastructure, a Grid provides dependable, consistent, secure and pervasive access to a wide variety of infrastructure resources that can be seamlessly shared among its users.*

In the aforementioned analogy, the web of physical power lines is mapped on to a *web of physical network connections* between infrastructure resources as illustrated in Figure 2.2. It also offers a comparison of the Web with the Grid approach. Also the 'geographically dispersed power grid generators' are mapped onto infrastructure resources used for computing (e.g. supercomputer) or data storage (e.g. storage devices) provided by different world-wide universities or research facilities. While this analogy between an electrical power grid and a Grid infrastructure is very clear, many discussions question the truth of it. Electricity is very different from computation and data in many respects, and the world-wide power grid still has some interoperability limitations to overcome too despite the fact that commercial energy providers put many millions into this. It is a known problem that power plugs in Germany are quite different from power plugs in US and the different voltages and frequencies in these countries can also cause problems with electrical devices.

In contrast to the electrical power grid that 'only' transports electrical power, a Grid infrastructure provides a *wide variety of services*, including the use of standardised text-based protocols through to different proprietary binary encoded results of scientists. Grid infrastructures, and thus e-Science infrastructures, are *much more complex than electrical power grids* in terms of standard-compliance, reliability, efficiency, and most notably interoperability. They provide a wider spectrum of services and their access is therefore governed by more complicated issues such as security concerns, and a broader set of protocols, standards, or operational policies. Another important difference to the 'sharing of static power generators' is *dynamic infrastructure resource sharing* that is often performed via so-called *Virtual Organizations (VOs)* [172]. These are defined as follows:

Definition 8 (Virtual Organisation) *A virtual organisation (VO) is a set of individuals and/or institutions that share infrastructure resources within an e-Science infrastructure within a certain period in time. It represents a temporary community overlay over classical organizational structures. VOs can vary enormously in their purpose, scope, size, duration, structure, community, and sociology.*

As shown in Figure 2.2, VOs enable dynamic sharing of infrastructure resources such as supercomputers, but also clusters, mass storage systems or other devices within e-Science infrastructures. Dynamic sharing adds another factor of complexity to e-Science infrastructures, but this concept is not being used by every e-Science infrastructure. For instance, EGEE/EGI uses the VO concept while DEISA/PRACE does not. VOs often differ in many respects such as the number and type of participants, the types of activities, the duration and scale of the interaction, and also the different types of infrastructure resources being shared. Examples of the VO concept are the different experiments and detectors (Alice [106], Atlas [105], CMS [148], LHCb [111], etc.) of the LHC where for instance Atlas also forms a VO within the EGEE/EGI infrastructure. The VO concept implies many technical challenges due to sharing relationships and different ownerships in e-Science infrastructures. In particular, a VO can be small or large, short or long-lived, single or multi-institutional, and homogenous or heterogeneous. Figure 2.2 illustrates VOs that can be structured hierarchically and may overlap in membership of infrastructure resources (e.g. supercomputer).

2.1.3 Classification of e-Science Applications

The previous sections clarified the term 'e-Science' and its major concepts including Grid infrastructures, VOs and resource sharing. This section models the given problem space more precisely by giving insights about 'applications' in the context of e-Science.

The introduction already mentioned a couple of so-called *'grand challenge applications'* of scientists that take advantage of e-Science infrastructures. Examples of such applications tackle well-known problems of science and society such as treating Aids or Alzheimers diseases, the localization of cancer, accurate global weather prediction, or determining the age of the earth. Since production applications are an important element of this thesis a much more deeper analysis of there characteristics is needed in order to provide the foundation for a production-oriented focus of this thesis. Scientists use production applications using distinct approaches that is published in [268] as a *'classification of e-Science applications'*.

Figure 2.3 provides an overview of the classification that consists of five distinct approaches. Each of it is briefly reviewed in this section while more details can be found in [268]. The underlying basic usage paradigm in all approaches is the use of 'middleware' [296], which is defined in more detail in the next section while this section remains the focus on applications.

The first approach are *'simple scripts and control functionalities'* that use simple control functionalities (e.g. loops, if-then-else constructs, etc.) and submissions of simple UNIX-like scripts to computational resources. One example is an application in the field of 'hydrodynamics', which refers to studies of liquids in motion. A fluid dynamics code known as multi-particle collision dynamics (MPC) [159] is applied to simulate active biological system models named sperm. Experiments have revealed an interesting swarm behaviour of sperm when the sperm concentration is high [228]. The mechanism behind this experimental phenomenon is not clear. Therefore, this e-Science application studies the 'sperm cluster size' dependence for 2D and 3D systems in terms of studying the hydrodynamics interaction between sperm and explain its importance to the cooperation behaviour. These simulations in 3D are very time consuming such that systematic study raises the demand for powerful computational resources available in e-Science infrastructures.

As one example of this first approach, the aforementioned application intensively uses the Do-N control functionality, where the output of the previous job run is given as an input to the subsequent job. In some cases this approach is used with codes that are independent of the underlying resource type and in other cases the code is specifially optimised for dedicated hardware architectures. As part of the thesis studies, the aforementioned hydrodynamics application used this approach with the JUMP [52] supercomputer with 41 Symmetric Multiprocessing (SMP) nodes, where each node has 32 processors. This 1312 Power4+ 1.7 GHz CPU machine (8.9 Teraflop/s peak performance) was part of the DEISA/PRACE infrastructure during the time of the thesis studies.

Another application [282] in the field of theoretical fluid mechanics uses the same approach with a very similiar setup but with the JUGENE supercomputer [51], which, during the time of the thesis studies, consisted of 65536 processors of type 32-bit PowerPC 450 core 850 MHz (223 Teraflop/s peak performance). Also scripts with middleware are used including the Do-N control functionality loop. The reason for scientists to use such loop functionality in particular is twofold. First, once the program is defined, their executions are submitted to the middleware for each iteration without manual interaction, which is also helpful during weekends. Second, many resources have a limited program execution run-time (e.g. 12 hours), and the Do-N control functionality provides a way to partition long jobs (e.g. over 60 hours) into smaller chunks that do not exceed the allowed maximum run-time.

The second approach is named *'dedicated application plug-ins'* and simplifies usage of e-Science infrastructures via predefined application GUIs (e.g. GAUSSIAN [35]) and a wide variety of job configuration options (e.g. number of required cores). The previous approach described above, uses simple UNIX scripts for program executions. This implies that scientists have to create the scripts that run executables by themselves. Hence, in the first approach, the scientific domain scientists have to know the potentials and drawbacks of UNIX scripts or scripting languages such as Python and Perl, which are also used for program executions with middleware. The scientists that are experts in their research field, and thus fully understand the theoretical model of their research, also have to become experts in many computational techniques. In contrast, scientists that follow the second approach prefer high-level Web portals [240], also named *scientific Gateways*, instead of low-level computational techniques. This is helpful since the evolution of multi-core and many-core systems in general, and the various options of programming high-end computers in particular lead to more and more complex computational techniques that have to be used in order to gain maximum application performance.

Using convenient application plug-ins that abstract from many of these complexities motivates the approach. Many Grid clients such as the UNICORE Rich Client [152], g-Eclipse [185], and the GridSphere Web portal [240] enable client extensions that are named as scientific application plug-ins. One widely used scientific application is supported in such correspondent client technologies via a scientific domain-specific GUI plug-in. One example is the GAUSSIAN [35] plug-in for the UNICORE client [152] that enables scientists to easily submit GAUSSIAN-based programs to computing resources available in e-Science infrastructures.

The third approach takes advantage of workflow engines using *'complex workflows'* often defined via *DAGs* to express an execution of a scientific workflow including several

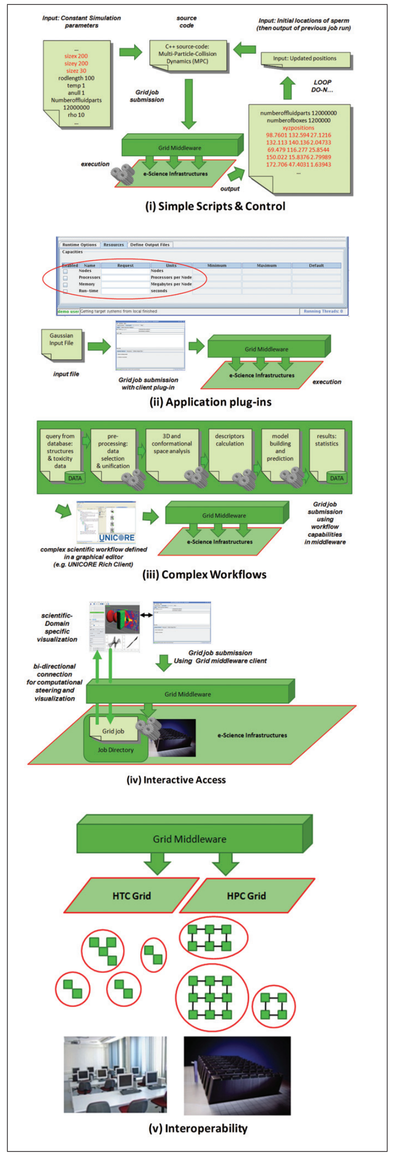


Figure 2.3: Classification of e-Science applications.

related application tasks. Hence, this approach defines dependencies between application tasks. A detailed example is given in [268] in the context of healthcare research using the *quantitative structure-activity relationships (QSAR)* [144] computational method.

The fourth approach is named as *'interactive access'* essentially meaning the use of a Secure Shell (SSH) [275] connection with Grid credentials. In this approach, the scientists utilize less support from the middleware, but often use this approach in order to check intermediate results during application executions initially started with middleware on computational resources.

A concrete example is gLogin [277], or the UNICORE SSH plugin [258] that offers the access to the application execution directory to review intermediate results. Important is that this approach satisfies the well known *single sign-on requirement* [131]. That means that connections are established by using the same set of security credentials that are used within the middleware to access the resources within an infrastructure.

The fifth approach, named *'interoperability'*, is unique in using different Grid infrastructures to perform different kinds of application tasks to finally achieve one overall common scientific goal. This approach eventually consists of elements of the other aforementioned approaches which are used in multi-Grid setups instead of one single e-Science infrastructure. The fifth approach is the underlying basis of this thesis and an introductory example can be found in [268]. Chapter 6 provides much more detailed application examples of this approach. It lists three distinct academic case studies that use the interoperability approach to perform e-Science out of different scientific fields: bio-informatics, e-Health, and fusion science.

This whole classification reveals details about how production e-Science infrastructures are used and the focus is set in the subsequent chapters of this thesis on this fifth type of applications. Based on the classification, e-Science applications are defined as follows:

Definition 9 (e-Science Application) *An e-Science application is a scientific program or code that is used with resources and services provided by e-Science infrastructures using one of the following methods with Grid middleware: simple scripts and control mechanisms, application plug-ins, complex workflows, interactive access and joint usage of multiple infrastructures in one scientific workflow.*

Definition 9 focuses on the scientific applications rather than on the functionalities of the infrastructures. Very closely related to this definition is the often used term *'Grid job'* that is not formally defined, because of its significant overlap with e-Science applications in this thesis. This term can be equally used in the context of applications, but is more often used when related Grid functionalities are necessary to run e-Science applications. It refers to a task or activity that should be executed on the Grid or e-Science infrastructure being submitted by scientific end-users that we define as follows:

Definition 10 (e-Scientist) *An e-Scientist is any scientist that takes advantage as an end-user of any kind of an e-Science infrastructure using Grid middleware for the execution and management of e-Science applications.*

As Definition 10 reveals, e-Scientists are end-users that take advantage of production e-Science infrastructures on a daily basis. They take advantage of collaboration and resource sharing as illustrated in Figure 2.1 within the 'e-Science roof'. But also administrators or support staff members can act in the role of an e-Scientist to perform evaluations or benchmark executions. From a certain perspective, e-Scientists can also be seen as a special kind of Grid resource due to their expertise and knowledge that they share through means of collaboration with international colleagues. But in this thesis, e-Scientists are end-users of e-Science infrastructures using various client technologies. It is important to understand that e-Scientists are not only those that use purely computational-focussed production e-Science infrastructures with some form of a client technology. Instead, end-users of different e-Science infrastructures such as those that use *research infrastructures (RI)* emerging from the *European Strategy Forum on Research Infrastructures (ESFRIs)* [274] activities can be also named as e-Scientists.

2.2 Key Technologies and Standards for e-Science Infrastructures

Several basic key technologies that are revelant in the context of e-Science infrastructures are introduced in this section. Relevant open standards that are used in general in the context of these technologies are introduced, because they play a crucial role in subsequent chapters. There are a plethora of technologies (e.g. workflow engines, middleware, resource management systems) and standards existing in the community while the focus is set on those that are directly relevant for the findings of this thesis.

2.2.1 Resource Management Systems and Grid Middleware

Computationally-driven shared infrastructure resources often require management technologies that enable the start, control, and monitoring of a program execution, or putting and getting files that are needed for its execution. Such technologies are often named as *Resource Management Systems (RMS)* that are defined as follows:

Definition 11 (Resource Management System) *A Resource Management System is a technology that provides distinct management mechanisms that enable the control and specific scheduling of e-Science application jobs and their program execution(s) on a compute infrastructure resource.*

In the majority of the cases, compute infrastructure resources have an RMS installed that ensures its integrity by fault tolerance models or by mutual exclusion of end-users that want to use specific parts of the resource at the same time. Well-known examples of RMSs for compute infrastructure resources such as supercomputers or large-scale clusters are LoadLeveler (LL) [205], PBSPro [75], Torque Resource Manager [98], and Load Sharing Facility (LSF) [53]. These RMS technologies provide mechanisms to query the status of the application job queue for different users (e.g. the command *qstat* in Torque).

Relevant for this thesis is that all these systems are responsible for the scheduling of submitted jobs to an infrastructure resource, including the distribution of jobs on the different nodes of parallel computers. More detailed pieces of information about the (partly unique) functionalities of each of these technologies can be found in the corresponding manuals.

Another important technology are *middleware systems* that follow the idea of abstractions as defined by Tanenbaum in [296]. The e-IRG blue paper 2010 describes middleware as follows. *'In a distributed computing system, for example, middleware is defined as the software layer lying between the operating system and the applications. In a broader sense, middleware is computer software that connects components or applications. Middleware is used to refer to the glue that enables virtualisation technology and services'* [210]. UNICORE [293], gLite [212], ARC [160], and Globus Toolkits [167] are all examples of *'Grid middleware'* that is defined as follows in the context of this thesis:

Definition 12 (Grid Middleware) *Grid Middleware is a technology that presents the Grid as a single system by hiding administrative and geographic boundaries and providing seamless, secure, and intuitive access by hiding their complexities in such a way that its corresponding e-Science infrastructures appears transparently to its users.*

Often, such middleware is based on a local RMS installed on an compute infrastructure resource, while storage infrastructure resources offer middleware access without being managed by an underlying RMS but via some dedicated storage technology.

Middleware architectures can be accessed using dedicated protocols and consist of several services that offer functionality like Grid job submission and management including the transfer of data to and from the infrastructure resource. In some cases, middleware provides software development kits (SDKs) and application programming interfaces (APIs) to simplify the development of Grid applications that take advantage of the various protocols and wide variety of services provided.

2.2.2 Service-Oriented Technologies

The majority of technologies (e.g. Grid Middleware) in this thesis are based on *Service Oriented Architectures (SOAs)* [276] and therefore important related topics are introduced as part of this section. While the fundamental architecture of Grids is defined as a layered protocol stack in [172], Foster et al. describes in [171] a more detailed Grid architecture using a *service-oriented approach* to interact between users, middleware and infrastructure resources.

In this model, each e-Science infrastructure resource has a form of *state* that should be exposed and accessible by the end-users or other services. Compute infrastructure resources (e.g. clusters), for instance, expose the status of current running jobs or the queue status of the underlying RMS to its end-users via middleware services that are stateful. *Grid services* are used to access and manage the status of infrastructure resources among the members of a VO. A first detailed definition of Grid services can be found in [299], which is nowadays already deprecated. There are actually many definitions like within [297] that also forms the basis of the following definition:

Definition 13 (Grid Service) *A Grid service is any kind of a service that operates in an e-Science infrastructure, manages stateful behaviour of infrastructure resources, and meets the requirements of the users or VO wherein the service is deployed.*

Definition 13 implies that every infrastructure resource is accessed by a Grid service and all components within an e-Science infrastructure are essentially virtual in service-oriented frameworks. Figure 2.4 illustrates the basic idea of this definition with some infrastructure resources in context. One key reason why such Grid services are needed is best explained in the e-IRG blue paper 2010 that points out that *'Science is increasingly global, and the rise in distributed research teams working with distributed data sources will continue to drive the need for distributed data processing and storage'* [210]. *'To service this need, the research sector has developed and deployed Grid services in Europe and around the world, supporting standards-based access to computers, storage, software, data and other non-IT resources, regardless of geographical location, administrative affiliation, and local management tools'* [210].

Another motivation is the ever increasing complexity of infrastructure resources as described in the e-IRG white paper 2011 [194]. In particular HPC resource complexity increases year by year reaching another peak in complexity when exascale systems will be in place towards 2020. One of the e-IRG proposed approaches is as follows: *'In particular, users, and especially those that lack a computer science background, cannot expected to program exascale computers effectively without appropriate software tools that hide complexity, facilitate parallelism, and let them concentrate on utilising their domain knowledge'* [194]. These appropriate software tools can be considered to be partly relying on Grid services although their name might differ but still similiar technologies might be used.

One set of specifications that describes a service-oriented Grid architecture that was defined to reduce some of the aforementioned complexities is the *Open Grid Services Architecture (OGSA)* [174]. This architecture is considered as 'the reference architecture of Grids' since the beginning of the Grid computing paradigm. OGSA is discussed in detail later, but the concept of Grid services is very fundamental and used to access the particular state of an infrastructure resource. OGSA defines capabilites of a service-oriented e-Science infrastructure that is needed for integrating and managing infrastructure resources within a VO. Its services are from many technical areas like execution management, data handling, file transfer, resource management,

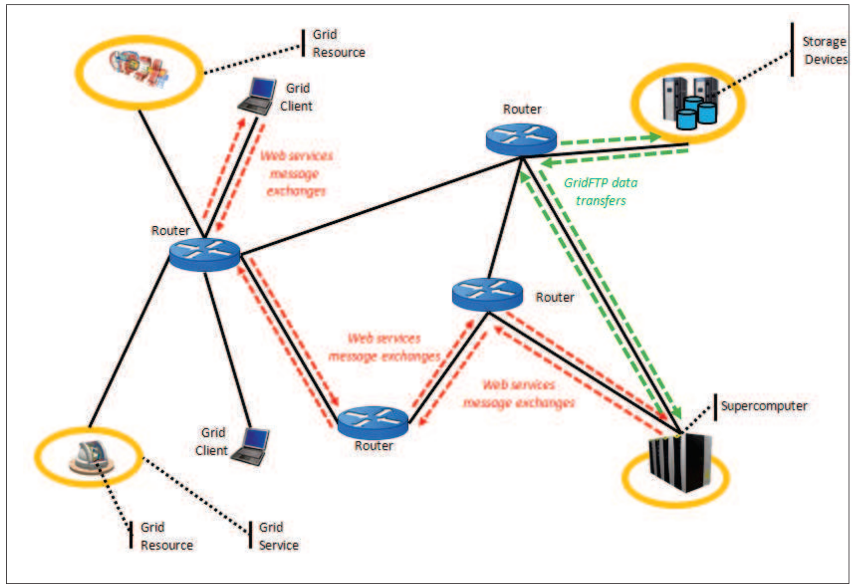


Figure 2.4: e-Science infrastructure that implements the concepts of OGSA with Grid services and Grid resources.

security, information, and self-management. The definition of the initial OGSA version 1.0 [173] has been updated to version 1.5 [174], and is driven by a set of functional and non-functional requirements defined complementary OGSA use case documents [168, 252].

The approach of Grid services is also used in this thesis as *major baseline state representation and communication model*. Grid services are implemented with *Web services message exchange technologies* [101] using the Simple Object Access Protocol (SOAP) [190], while large-scale data transfers are often performed with GridFTP [109]. Figure 2.4 illustrates the concept of Grid services and stateful infrastructure resources following the basic concept of OGSA.

2.2.3 Common Open Standards in the e-Science Domain

Since the beginning of Grid computing there have been many issues surrounding the integration of OGSA concepts in e-Science infrastructure and standardisation of Grid services within Grid middleware systems. This is also reflected in the e-IRG blue paper 2010 in a sense that *'A central theme of middleware development is the promotion of interoperability and standardisation of networked resources through a common base of protocols and services'* [210].

But working interactions among networked OGSA Grid services are non-trivial when based on Web services message exchanges using the Extensible Markup Language (XML) [296]. These exchanges represent an *XML-based Remote Procedure Call (RPC)* [296] where each single difference in the used XML-based protocol or schema can break the interconnection between services and clients. Standardisation of these XML-based protocols or schemas bears the potential to enable more functioning and stable interconnections between the Grid services that form an e-Science infrastructure. Standards are defined as follows in this thesis:

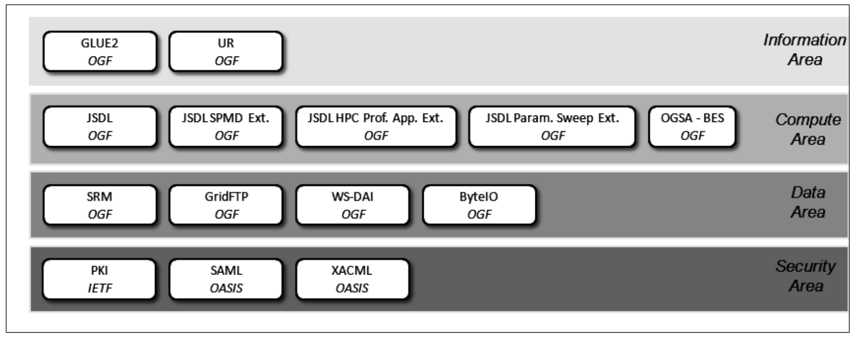


Figure 2.5: Open standards relevant in this thesis with their different technical areas.

Definition 14 (Common Open Standard) *A common open standard is any standard developed by a standardization development organisation following an open process and being commonly relevant to the e-Science community. It is normatively defined and publicly available.*

As stated within Definition 14, *common open standards* are normatively defined in specifications while some of them are *emerging open standards* meaning that these are not implemented by many technology providers yet. A clear definition about emerging open standards is not possible, but it is a known term in Grid computing.

Common open standards are mostly developed and released by so-called *Standardisation Development Organizations (SDOs)*. SDOs relevant in the context of this thesis are most notably the *Open Grid Forum (OGF)* [68] and the *Organization for the Advancement of Structured Information Standards (OASIS)* [73]. The particular relevance of OGF specifications in this thesis and in Figure 2.5 is widely recognised as the 5th e-Infrastructure concertation meeting report reveals: *'What is also clear is that almost all Grid Middleware standardisation activities are being taken through the OGF rather than any other standardisation body'* [132].

Key standards of other SDOs are also partly used like the X.509-based *Public Key Infrastructure (PKI)* [195] of the Internet Engineering Task Force (IETF). Key standards of the aforementioned SDOs are introduced in this section with a focus on those that are most important for this thesis and normative specifications are given in context. Figure 2.5 provides an overview of relevant standards and in which technical area they can be categorized.

One of the key standards in this thesis is the OGF specification GLUE2 [113]. It represents a *Grid information model* that defines attributes (i.e. properties) for certain important entities like computing or storage resources and Grid services. The GLUE2 schema is an evolution from the proprietary GLUE1.3 [140] schema that was created during the course of the EGEE series of projects. The OGF *Usage Record Format (UR)* [216] is a normative schema for tracking resource usage. It stands for *resource usage information*, often categorized in the accounting area, but in this thesis being part of the information area.

The *Security Assertion Markup Language (SAML)* [142] from OASIS is used in commercial setups, but also recently more and more in the scientific domain. It is a very extensive standard but mostly used in this thesis for the transfer of security attributes that describe the VO or project membership as well as the role posessions of end-users. SAML has much potential to be the next generation e-Science security standard and as such it is one cornerstone of the proposed reference model design. The *eXtensible Access Control Markup Language (XACML)* [234] is the counterpart to SAML providing a very strong language for the definition of security policies [131] used during authorization decisions. XACML is developed by OASIS and is also relevant to this thesis in order to define invariants of using common security attributes in reference model setups. Both aforementioned standards are the most important ones in the context of this thesis, but in [131] the plethora of *Web services security (WS-Security)* related standards are listed for more details.

Another important open standard schema is the *Job Submission and Description Language (JSDL)* [115]. This OGF standard describes how computationally-driven jobs can be submitted to Grid middleware. It is already used in production setups within e-Science infrastructures and several extensions have been defined in the past. These are the *Single-Program-Multiple-Data (SPMD) JSDL extension* [281], the *JSDL Parameter Sweep Extension* [155] and the *HPC Profile Application Extensions* [197]. Some aspects of these JSDL specifications are quite vague since they significantly overlap with GLUE2 entities leading to some disambiguities when GLUE2 and JSDL are used together in e-Science infrastructures. To resolve some of these disambiguities is a major technical goal of this thesis and as such the JSDL standard is also an important standard in this thesis.

Closely related to the JSDL standard is the OGF *OGSA-Basic Execution Service (BES)* specification [169] that makes use of JSDL in order to submit jobs to computational resources. OGSA-BES specifies exactly those operations that are required to submit and manage computational activities within Grid middleware adoptions. Initial OGSA-BES adoptions have been used in production setups that have contributed to many lessons learned of how this specification can be improved. Therefore, one element of this thesis is to propose extension to this specification to satisfy production e-Science infrastructures requirements.

The *Storage Resource Manager (SRM)* [286] is a storage management specification that is very well adopted by over five different implementations (e.g. dCache [178], Disk Pool Manager [11], etc.) that serve different end-user needs. Complementary to this storage standard is the access to relational databases using the *WS-Data Access and Integration Services (WS-DAIS)* [118] specification. The OGF *GridFTP* [109] defines mechanisms for the use of a tuned File Transfer Protocol (FTP) [295] for very large data amounts making it the de-facto large-scale data transfer standard available in distributed computing today. Another data transfer standard that is relevant is the *ByteIO* specification [230] that offers access to files with remote POSIX-based methods.

Finally, a couple of standards are not directly related to this thesis but are mentioned sometimes. These are *WS-Agreement* [114], *Distributed Resource Management Application API (DRMAA)* [250] and the set of specifications around the *OGSA - Resource Usage Service (RUS)* [65] that are still in the development phase.

2.3 e-Science Infrastructure Interoperability Challenges

The previous sections of this chapter introduce the major basic terms and approaches in the context of e-Science infrastructures. Based on these fundamentals, the aim of this particular section is to model the given problem space and thus clarify the formulated research question in Chapter 1 in more detail.

The model of the problem space provides a clear focus within the broader research field of e-Science infrastructures so that important issues within the given boundaries of this thesis can be understood. The thesis does not attempt to tackle *'all services of e-Science infrastructures'* and also it does not provide a solution to *'all known e-Science infrastructure interoperability problems'*. But the focused problem space of this thesis can be defined as follows:

Definition 15 (Network of Interoperable Services) *A network of interoperable services is a set of common open standards-based interacting Grid services within production e-Science infrastructures that manage and control heterogenous compute and storage resources.*

Definition 15 reveals the fundamental problem mentioning the lack of interoperability among a certain set of Grid services that enable access to various types of resources in production e-Science infrastructures. This definition clearly narrows the broader scientific field to one specific environment (i.e. production e-Science infrastructures) for which this thesis aims to provide some solutions.

Interoperability of production environments is a complicated matter, not only in computer science, but also in many other environments, such as the interoperability of electrical power environments. After decades of production usage and standardization activities, still different electrical power adapters are required world-wide while electrical power Grids *'only' provide one type of service (i.e. electricity)* compared to e-Science infrastructures that offer *multiple services to end-users*. A resulting network of services of various interoperable e-Science infrastructures is thus a very complex problem space with a highly heterogeneous set of interacting infrastructure resources (e.g. different types of resources specifically tuned for different computational paradigms) and a wide variety of deployed technologies (e.g. middleware, storage technologies, etc.). Different functionalities out of various computer science fields (e.g. compute, data, information, and security) exist that are part of the complex e-Science infrastructure environments. Such services form *'production environments'* that in turn raise the demand to *cope with dynamic changes in services and technologies* while at the same time preserving the ability of the infrastructures to serve the needs of end-users to perform their science with them on a 24/7 basis.

The overall aim of this section is to highlight the given boundary conditions that affect the interoperability of e-Science infrastructures (i.e. different resource types, or dynamic behaviours, etc.). Relevant interoperability challenges are explored in this section leading to more detailed models of the problem space. This forms a consistent basis for a meaningful survey of related work in the next chapter. While exploring the challenges of interoperable e-Science infrastructures, also a wide variety of benefits are revealed when comparing advantages and disadvantages of such setups during the course of this section. The understanding of the challenges that needs to be solved in turn motivates tackling the given research question of this thesis.

2.3.1 Classification of e-Science Infrastructure Types

A more thorough analysis of OGSA in general, and its comparison of e-Science infrastructures with electrical power Grids in particular, reveals that there is a difference in types. As within

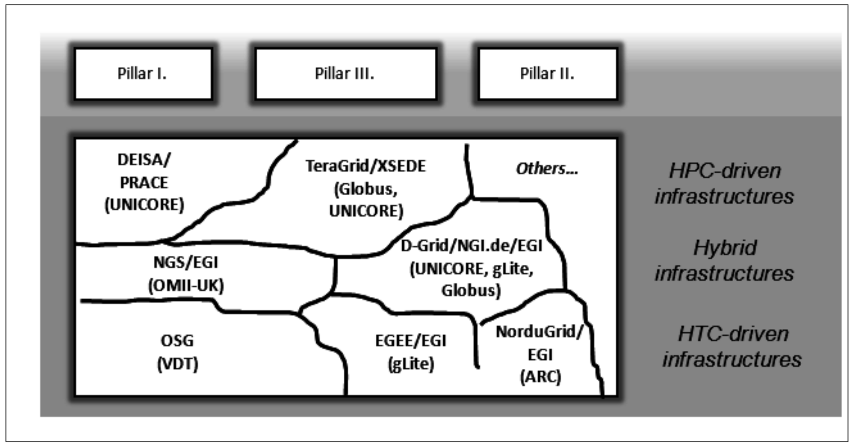


Figure 2.6: Incompatible e-Science infrastructures form a non-solid infrastructure basement.

electrical power Grids, the power is provided by coal-based power plants, atomic-energy-based power plants, and, more recently, also from 'green power sources' like solar collectors and giant wind wheels. The resource landscape of the electrical power Grid is in this sense also very heterogeneous like in the given problem space of this thesis.

The concept of e-Science is already introduced and its scientific innovation process is driven by three pillars (theory models, computational techniques, and experiments). In an ideal situation, these three fundamental pillars should be based on a solid e-Science infrastructure basement. But over the years, various types of e-Science infrastructures evolved leading to a classification in [272] according to their fundamental computing paradigms (i.e. HPC and HTC) they support. Storage and data would be equally important in the context of both computational paradigms but these are less considered to remain the focus on the computational aspects of the given problem space. Based on the classification, the solid basement is 'not solid' as illustrated in Figure 2.6. It consists of many different basement elements that stand for numerous e-Science infrastructures that can be 'stuck' together to form a single *'virtual next generation infrastructure'*.

As shown in Figure 2.6, the classification reveals three major types of e-Science infrastructures that are named as HPC, HTC, and hybrid. In theory it is hard to define the clear boundaries for this classification between those infrastructures. In practice, however, the boundaries and scope of these categories are fundamentally different, especially when the resource type is considered as well as the overall usage and access policies (e.g. peer-reviewed grant-based access only).

There are even boundaries between infrastructure examples of the same classification type as well. This is the case since each of the examples, even those in the same category, are often represented by different projects using a different set of technologies and often also policies. Hence, even infrastructures of the same category are non-connected basement elements. *'HPC-driven e-Science infrastructure types'* consist of large-scale clusters or supercomputers that are defined as follows:

Definition 16 (HPC-driven e-Science Infrastructure) *A HPC-driven e-Science Infrastructure is based on computing resources that enable the efficient use of parallel computing techniques through specific support with dedicated hardware such as high performance cpu/core interconnections.*

As Definition 16 indicates, HPC in this thesis refers to e-Science infrastructures that provide computing resources that offer excellent interconnection between the cpus/cores in order to support massively parallel applications. This is required for e-Science applications that take advantage of *parallel programming techniques*. Examples are the Message Passing Interface (MPI) [241] or OpenMP [147] that are both used in order to gain greatest efficiency during production runs on large-scale HPC resources. As shown in Figure 2.6, the two best known examples in this category are TeraGrid/XSEDE in the US and DEISA/PRACE in Europe. Both infrastructures deploy different middleware technologies namely UNICORE in DEISA/PRACE and Globus Toolkit within TeraGrid/XSEDE although there are plans to include more standardized middleware in the latter infrastructure [235].

In contrast, the second category is represented by *HTC-driven infrastructures* that focus on the broad support of 'farming Grid jobs', also sometimes known as being 'embarrassingly parallel' or 'nicely parallel' computing jobs. All these terms share the fact that they do not require a good interconnection between the cpus/cores, essentially being well suited for *'data parallel'* tasks possibly in a distributed fashion. As shown in Figure 2.6, the most known e-Science infrastructures of this type are EGEE/EGI and NorduGrid in Europe and OSG in US. Both examples deploy different technologies, namely gLite in EGEE/EGI, VDT in OSG and ARC in NorduGrid. In some cases, and more recently, EGEE/EGI overlaps with NorduGrid. NorduGrid can be considered as a regional Grid and is not a National Grid Initiative (NGI) [58], but many of its resources are also part of the EGEE/EGI infrastructure. HTC-driven e-Science Infrastructures are defined as follows:

Definition 17 (HTC-driven e-Science Infrastructure) *A HTC-driven e-Science infrastructure is based on commonly available computing resources such as commodity PCs and small clusters that enable the execution of farming jobs without providing a high performance interconnection between the cpu/cores.*

'Hybrid e-Science infrastructures provide access to a limited set of rather medium-scale HPC-based resources while still providing access to PC pools and smaller clusters commonly used for HTC. Known infrastructures of this category are, among others, the National Grid Service (NGS) [59] of the UK, and the German national Grid D-Grid/NGI-DE [238]. Infrastructures of this category are often regional Grids or NGIs that form the basis for the EGI infrastructure such as the Polish NGI [120]. They often deploy numerous technologies at the same time, like the Open Middleware Infrastructure Institute (OMII)-UK stack in the NGS and Globus, gLite, and UNICORE in D-Grid/NGI-DE. These types are defined as follows:

Definition 18 (Hybrid e-Science Infrastructure) *A hybrid e-Science infrastructure is based on computing resources that enable the execution of HPC as well as HTC computing jobs using techniques suitable for both computational paradigms where appropriate.*

2.3.2 Tightly Coupled Middleware Clusters in e-Science Infrastructures

The previous section indicated that e-Science infrastructures can be classified according to known computational paradigms that are very mature and are considered to not change in the near future. Based on this mature classification, this section provides a deeper analysis of the challenges reaching one single e-Science infrastructure basement. Particular *'infrastructure interoperability problems'* are introduced with details about relevant deployed and non-interoperable middleware setups and their interactions. Figure 2.7 offers an overview of the

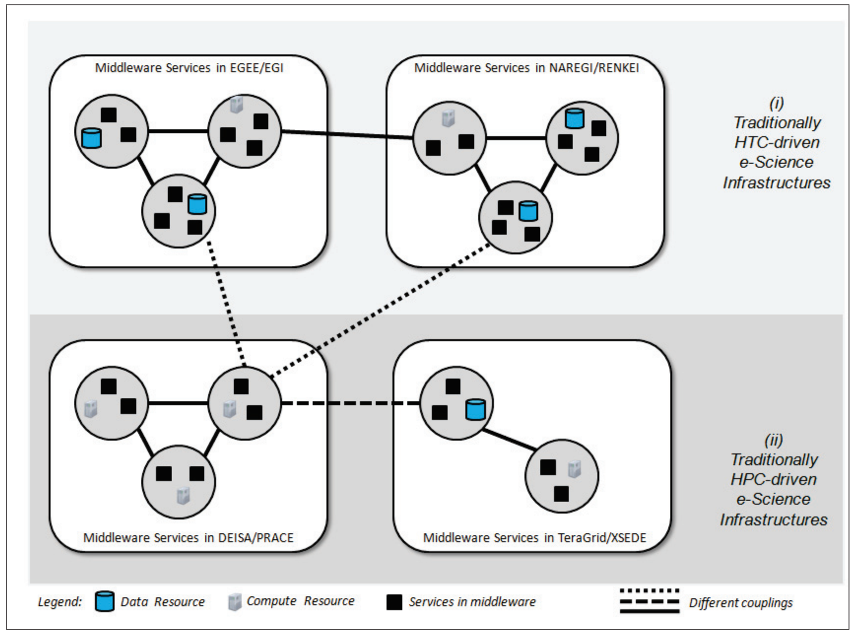


Figure 2.7: Tightly coupled clusters of middleware services in different production e-Science infrastructures.

interoperability problems between production e-Science infrastructures that are mostly related to their coupling.

As shown in Figure 2.7, the hybrid type of infrastructures have been neglected to retain a simplistic model of the problem space essentially highlighting compute and data resources around middleware. Nevertheless, as hybrid infrastructures (e.g. NGIs like D-Grid/NGI-DE or NGS) offer the same resource types (e.g. HPC and HTC), the findings of this thesis can be applied equally to them. Thus focusing on HPC- and HTC-driven infrastrucure types, Figure 2.7 illustrates the fragmented situation of the non solid e-Science infrastructure basement. The basement is represented by a set of world-wide non-interoperable *'Grid islands'* being interconnected in different ways depending on the couplings of the middleware technologies.

Present day Grid islands are the EGEE/EGI and DEISA/PRACE infrastructure funded by the European Commission, while OSG and TeraGrid/XSEDE are supported by US funding sources. The reason of having such Grid islands are manifold and closely related to known interoperability challenges of them. Models of this section focus on the most important aspects neglecting low-level details of communication between the services that are simplified as different types of couplings. The couplings are represented by Web service message exchanges.

The project-based funding by different funding bodies often leads to a focus of certain goals for these infrastructures following a precise description of work, thus enabling dedicated computing and data services. Intra-infrastructure issues are often more important leading to the deployment of tightly coupled middleware clusters within one infrastructure based on one chosen middleware. Interoperability between different infrastructure types is in many cases a secondary goal and thus in very minor cases a coupling between such tightly-coupled middleware services exist in a setups using more than one infrastructure. Even within one infrastructure category, different middleware technologies are deployed establishing tightly-coupled middleware clusters for every single infrastructure as illustrated in Figure 2.7. This deployment contributes to strong e-Science infrastructure borders that are non-trivial to technically overcome since many different interfaces and protocols are in use that are non-interoperable. The e-IRG white paper 2009 describes the need of change in this regard as *'The current multitude of interfaces from different vendors may otherwise endanger the e-IRG vision of an open e-Infrastructure that allows optimal use of all electrically available resources'* [206].

The aforementioned infrastructure boundaries are even hardened when their *different usage and access policies* are considered. Different resource usage policies are used when HTC- is compared to HPC-driven e-Science infrastructures. In HPC-driven infrastructures, costly computational time is only provided to particular research groups that pass a multi-step scientific peer review process of their research proposals. One implementation of this approach is the evaluation process of the *DEISA Extreme Computing Initiative (DECI)* [301]. It evaluates requests for computational time on DEISA based on proposals and their scientific excellence.

In contrast, HTC-driven infrastructures usually provide access to their resources based on individuals that are members of well-known VOs. Members of a VO do not need to have specific research proposals to gain computational time, although the whole VO as such must undergo some form of evaluation when it is initially set-up.

A more formal definition based on the aforementioned aspect around e-Science infrastructure boundaries is given as follows:

Definition 19 (Non-interoperable e-Science Infrastructures) *Non-interoperable e-Science Infrastructures are two or more infrastructures that have different usage policies and deploy non-interoperable technologies based on different (proprietary) interfaces, protocols, or schemas for communication.*

Definition 19 helps to understand that the e-Science infrastructures in Figure 2.7 are all non-interoperable. The reason is that, at the time of writing, all the different middleware technologies deployed on the illustrated infrastructures use incompatible interfaces and protocols and lack an adoption of the same common open standards. Atlthough different levels of established interactions exist (i.e. different lines in Figure 2.7), they are based on workarounds, adapters, tweaks or whatever needs to be done to interact with each other as described in [256]. In the majority of the cases end-users need to use different client technologies in order to use the resources of both infrastructures.

Incompatible interfaces, protocols, or schemas are in many cases due to proprietary developments rather than using common open standards. This is also mentioned in [272], following another important statement meaning that non-interoperable e-Science infrastructures are existing, because of to the *absence of a production-oriented standard-based infrastructure reference model*.

The state-of-the-art production e-Science infrastructures thus still struggle to provide e-Scientists with a stable (interoperable) infrastructure basement to fully leverage a wide variety of resources with compatible technologies.

2.3.3 Benefits of a Network of Interoperable Services

The previous section modelled the problem space more precisely, introducing details about the existing boundaries between production e-Science infrastructures today. In contrast, the *ideal situation* illustrated in Figure 2.8 would significantly support e-Science applications in multi-Grid setups better as possible today. The described infrastructure boundaries need to

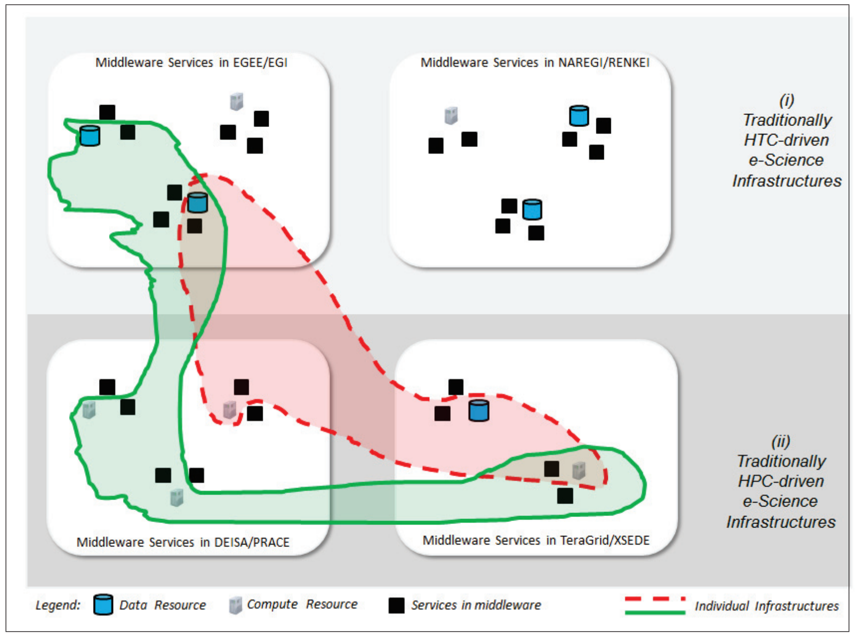


Figure 2.8: A network of interoperable services that realize the vision of individual infrastructures.

be significantly lowered or even completely removed. This in turn enables a wide variety of benefits that motivates the work within this thesis. In [266], we present such an ideal situation enabling optimal access to individually needed services as illustrated in Figure 2.8.

Compared to the initial model in Figure 2.7, the boundaries of infrastructures are fully transparent to end-users leading to *'borderless services'* enhancing end-users choice in using the established services. Previous tightly-coupled clusters of Grid middleware systems are exchanged with a loosely-coupled network of interoperable services dynamically formed by end-users. This optimal situation describes how end-users intend to work with multiple infrastructures that leads to the following definition:

Definition 20 (Individual e-Science Infrastructures) *Individual e-Science infrastructures are dynamically formed infrastructures according to the needs of end-users and are based on the services of different e-Science infrastructures that seamlessly interoperate.*

Definition 20 emphasizes on the desire that end-users would like to pick the services that provide best *'individual functions'* that they need to perform e-Science tasks, neglecting any form of existing infrastructure boundaries. End-users are mostly not computer scientists and thus are not interested in picking services because of technical reasons such as using dedicated security credentials or a limited set of client technologies.

Figure 2.8 models this flexibility and freedom to choose any kind of required service as an *overlay without a specific form* expressing its dynamic nature. With this setup, end-users are not bound to the services and thus the implied functions and resources of one particular e-Science infrastructure. This *flexibility* is experienced by end-users as one of the most fundamental benefits of a network of interoperable services enhancing their options for a real choice.

An analysis of the advantages and disadvantages that come along with such aforementioned individual infrastructures are pointing to a wide variety benefits for end-users. First, resource sharing across infrastructure boundaries enables access to a *broader spectrum of resources*, e.g. convenient usage of different types of computational resources (e.g. HPC and HTC). It *saves computational time* on rare and costly HPC resources when performing smaller evaluation jobs on HTC before going to 'full-blown' production runs on HPC infrastructures. Lowering the boundaries of infrastructures provides better load-balancing between different e-Science infrastructures and their resources (if this is possible on the policy-level). This enables an overall *lower time-to-solution* for e-Science applications by simply being able to use more systems in parallel, hence, a more aggregated usage of them, than possible before.

Interoperability as a smart way of extending the functionality of one infrastructure is another benefit. For instance, the use of brokering functionality in HTC-driven infrastructures while not being bound to the explicit choice of resources as in HPC-driven infrastructures.

There is also the possibility of taking advantage of the *unique capabilities* existing in compatible e-Science infrastructures. It enables the use of unique key resources or services that may lead to more realistic simulations (e.g. the high amount of computer core capability of one resource). It enables access to unique data repositories (i.e. obtained from specific scientific instruments such as the LHC) that are, in many cases, only available in a single e-Science infrastructure.

Hence, when the landscape of production e-Science infrastructures conforms to Definition 20, the *wide variety of benefits* motivates the investigation of solutions towards this vision in this thesis. But numerous challenges are hidden in the crucial service couplings, that have been neglected for simplicity in Figure 2.8 but that are full of open questions for which this thesis aims to provide some solutions. It is non-trivial to realise a network of interoperable services as an overlay on top of existing non-interoperable production e-Science infrastructures. A gap between the ideal situation and reality exists, but solutions can be explored that reduce this gap that in turn motivates this thesis. The key to achieve this is to increase the chance of having *interoperable e-Science infrastructures* that are defined as follows:

Definition 21 (Interoperable e-Science Infrastructures) *Interoperable e-Science Infrastructures are two or more infrastructures that share relevant usage policies and that deploy a necessary amount of interoperable technologies based on common interfaces, protocols, or schemas for communication.*

2.4 Conclusion

This chapter provides insights into e-Science in general and introduces the major e-Science infrastructure concepts. Conclusions of these fundamentals are that important concepts like resource sharing within e-Science infrastructures provide many advantages to e-Scientists today. The majority of existing production e-Science infrastructures are implemented using Grid methods while, more recently, there is interest in using also *cloud computing* methods. Grids and Clouds share several key approaches and are both essentially distributed compute and data infrastructures and as such many e-Science concepts are relevant to both. But the majority of the findings of this thesis have been gathered within existing production e-Science infrastructures, and given that the key objectives of clouds are mostly business oriented rather than purely science-driven, the focus remains on Grid methods in this thesis.

Another conclusion is that e-Scientists are specifically interested in the interoperability between HTC- and HPC-driven e-Science infrastructures for their research. But the major production e-Science infrastructures are non-interoperable and as such end-users are not able to freely choose the services and resources they require to perform e-Science. Performing effectively e-Science is only possible when not being limited by infrastructure boundaries. Instead, end-users need e-Science infrastructures that deploy technologies that implement common interfaces, protocols, or schemas for communication that are compatible with those of others. This raises non-trivial challenges since production e-Science infrastructures are complex in deploying a wide variety of services and offering access to different kinds of resources while conforming to distinct usage policies. This complexity should be hidden from end-users, offering an easy, but functionally attractive way, of using interoperable e-Science infrastructures with compatible interfaces and policies through middleware. This enables end-users to be able to seamlessly use multiple e-Science infrastructures with only one client technology instead of a plethora of incompatible access methods.

Another conclusion is that interoperable infrastructures provide benefits for e-Scientists, but also for infrastructure and technology providers. Having no infrastructure boundaries also paves the way for better economic service provisioning that avoids, for instance, duplicate technology developments and deployments. The re-use of specific services that are already available in an another interoperable infrastructure can in some cases avoid the deployment or new development of similiar services. Administrators and user support structures have the chance to benefit from interoperable infrastructures due to common configurations and potentially the same service monitoring features.

Relevant technologies have been surveyed including numerous interesting approaches like OGSA contributing to the understanding of why interoperable e-Science infrastructures are not available in production today. Although the idea of interacting abstract Grid services (like within OGSA) is relevant, real implementations of this approach shows many problems. The implementation of the abstract Grid service concepts with concrete Web service communication is an appropriate method while at the same time being very complex and non-trivial during interactions in distributed systems. Many standards are in place that support such interactions, but not many of them are adopted or provide enough functionality to be often used in day to day practice in e-Science infrastructures today. Hence, the solely standard-based approach to interoperability largely fails to deliver solutions until now. Many reference models in related fields are based on open standards and promoting interoperability but are not fully appropriate like the fundamental Grid reference model OGSA. Another conclusion is that identified solutions that lower infrastructure boundaries need to have a complete aligned process to sustain interoperability gaining the trust of end-users over time.

Chapter 3

Related Work

In the last chapter, the state-of-the-art in the field of Grid computing with a particular focus on topics related to production e-Science infrastructures is reviewed. A wide variety of terms (e.g. e-Science, Grid middleware, etc.) is introduced and several important concepts (e.g. resource sharing, Grid services, etc.) that are relevant to this thesis are defined. The last chapter aims to provide thus a reasonable level of understanding of current practice in the e-Science community. But it also points to the existing interoperability challenges in state-of-the-art production e-Science infrastructures that in turn motivate this thesis and this survey of related work.

While the last chapter models the interoperability problem space in detail, this chapter aims to compare the identified problems with similiar problems and existing solutions. One of the major reasons for the interoperability problem is the complexity of e-Science infrastructures that currently deploy a wide variety of services and offer access to many different types of resources. But while it seems obvious that open standard-based common interfaces, protocols, and schemas for communication are important to enable interoperability, this chapter explores in greater detail why standards alone are not sufficient to ensure interoperability. The research question of this thesis is therefore *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'* and it does not directly mention open standards, even though they indirectly play a crucial role as outlined during the course of this chapter. The identification of relevant approaches and their factors and indicators enable a comparison to be made as to whether an approach has the potential to promote interoperability in production e-Science infrastructures.

As a first approach, the relevance of reference models in the context of interoperability is discussed and important reference model key principles are identified. A *'foundational reference model ecosystem'* is introduced to understand how an abstract reference model can be implemented on different levels to be used with real e-Science applications. A critical review of the basic reference model in Grid computing known as OGSA is additionally provided to identify further relevant detailed indicators per factor pointing to a lack of adoption in production e-Science infrastructures today. Complementary to this OGSA review, several disadvantages are identified that enables an understanding of drawbacks of non standards-based approaches existing in various related *'component-based approaches'*.

Based on the identification of relevant approaches in the first section, the second section surveys reference models in the broader field of distributed systems from a theoretical point of view. The defined factors and indicators for comparing existing solutions are used throughout the survey of related reference models.

The third section provides a complementary, more practically-oriented, classification of component-based approaches towards interoperability. This survey enables an analysis of lessons learned in practical field studies that aim to take advantage of interoperability.

3.1 Identification of Relevant Approaches and Factors

This section identifies relevant approaches that have the potential to enable interoperability between production e-Science infrastructures. It provides insights into why abstract *'reference models'* and their more concrete related elements (e.g. reference architecture, patterns, etc.) promote interoperability from a general distributed systems perspective. Complementary to this general perspective, important *'factors and indicators'* of interoperability approaches from a particular Grid and e-Science perspective are identified.

3.1.1 Reference Model Foundations and Factors

The introduction already pointed to one of the major claims in this thesis, in that the *absence of a production-oriented and community accepted reference model* is the reason why current production e-Science infrastructures are largely non interoperable today. In this section, the actual meaning behind the broadly defined term *'reference model'* is contextualized with the problem domain of this thesis. Existing models in the field of distributed systems are surveyed. In addition, related architectural elements are reviewed in the light of 'how a reference model achieves the potential to promote interoperability'. All of these elements are summarized under the wider umbrella term 'reference model'.

Lessons Learned from the ISO/OSI and TCP/IP Reference Model Examples

One of the most well-known reference model examples in distributed systems and computer networks is known as the *Open Systems Interconnection (OSI)* [295] reference model developed by the *International Standards Organization (ISO)* [48]. This reference model deals with connecting so-called *'open systems'* standing for a wide variety of *'systems'* that are open for communication with other systems. It is organised in seven layers but what it represents is not a *'network architecture'* because it does not specify the exact services and protocols to be used in each layer and instead just describes what each layer does [295].

However, while the ISO/OSI reference model is widely known, the reference model Transmission Control Protocol (TCP) / Internet Protocol (IP) [295] is currently used for the world-wide Internet. A fundamental contrast to the ISO/OSI model is its *'more compact'* design, essentially described with a four layer model that still fits the more general ISO/OSI seven layer model. A detailed comparison of both models is out of the scope of this chapter but given in Piscitello et al. [246]. The lessons learned from the success of TCP/IP are reviewed in the following paragraphs based on insights revealed by Tanenbaum in [295].

The ISO/OSI and the TCP/IP model have much in common since both are based on the concept of a stack of independent protocols and the functionality of some layers is also very similar. But a more thorough analysis reveals many interesting differences about whether a reference model should be broad (i.e. ISO/OSI) or rather compact (i.e. TCP/IP). The TCP/IP model did not clearly distinguish between service, interface, and protocol, although some people tried to make it more like the ISO/OSI model [295]. That means that the protocols in the ISO/OSI model are better hidden than in the TCP/IP model and, more notably, the ISO/OSI reference model was developed before the corresponding protocols were invented. The ISO/OSI model was not biased towards one particular set of protocols and thus it is very general and completely in-line with software engineering principles. However the downside of ISO/OSI is that the *'designers did not have much experience with the subject and did not have a good idea of which functionality to put in which layer'* [295].

The reference model was not usable as planned and, what is interesting in the context of this thesis, *no thought was given to internetworking* [295]. This in turn points to its drawbacks regarding interoperability with other systems that are compliant with the ISO/OSI model. Tanenbaum nicely concludes this topic with *'things did not turn out that way'* [295] essentially meaning the difference between theory and practice.

The more successful history of the TCP/IP model was quite opposite to the ISO/OSI model. The protocols came first, and *'the model was essentially just a description of the existing protocols'* [295]. The actual protocol implementations fit the model perfectly while the model basically does not fit any other protocol stack. It is rarely useful to describe other, non TCP/IP networks, since even if TCP can be exchanged with the User Datagram Protocol (UDP) [295] it still relies on the specific IP protocol. Hence, neither the ISO/OSI nor the TCP/IP models are perfect, and thus *'criticism is directed at both of them'* [295].

The production (i.e. practice) focus points to a few important critiques that lead to design decisions in later chapters and that are summarised as follows. First, according to Tanenbaum [295], *'... it appears that the standard OSI protocols got crushed'* since the competing TCP/IP protocols were already in widespread use. Second, *'When OSI came around, they did not want to support a second protocol stack until they forced to, so there were no initial offerings ... OSI never happened'* [295]. Third, *'the choice of seven layers was more political than technical ...'* and more notably, *'The OSI model, along with the associated service definitions and protocols, is extraordinary complex'* [295]. Also, *'they are also difficult to implement and inefficient in operation'* [295]. *'The enormous complexity of the model and the protocols, it will come as no surprise that the initial implementations were huge, unwieldly, and slow'* [295]. There is some analogy between the OGSA and the ISO/OSI reference model in the context of the aforementioned issues. These are lessons learned that influence requirements and reference model design decisions later.

In contrast, *'first implementations of TCP/IP was part of Berkeley UNIX and was quite good (not to mention, free). People began using it quickly, which led to a large user community, which led to improvements, which led to an even larger community.'* [295]. This does sound positive, while there are also some critiques noted as follows. First, *good software engineering practice requires differentiating between the specification and its implementation, something that OSI does very carefully, and TCP/IP does not'* [295]. Another drawback is that the *'TCP/IP model is not at all general and is poorly suited to describing any protocol stack other than TCP/IP'* [295]. Despite the critics, the thesis outlines an intentional analogy between the proposed reference model and the TCP/IP reference model. The conclusion of the comparison of the ISO/OSI and TCP/IP example, when considering the use of production today and although being not perfectly designed tends to be the TCP/IP design approach in terms of production. One of the reasons is the success of the worldwide Internet that offers many *'interoperable services'* and is based on TCP/IP and thus it is a perfect example of how a *'reference model for a network of interoperable services'* should look like given its present practical relevance in science and business.

Credits of the initial Internet success also go to the interoperability studies in this field between different military areas and units within the US (e.g. Department of Defense), or, more recently, even between countries within the North Atlantic Treaty Organization (NATO). Military sources [233] related to interoperability and reference models have been analysed and share the same problems of establishing a network of interoperable services across various different systems contributing to the term *'systems of systems'* often used within this particular field. The notion of *'network of interoperable systems'* [233] is taken for this thesis while many military-specific protocol issues are surely not relevant in this thesis.

SOA Reference Model Foundations and Associated Architectural Design Elements

After the interoperability example based on a widely known reference model in the last paragraph, this paragraphs clarifies reference models foundations. Also associated elements are introduced to make reference model useful in production environments. The review of the state-of-the-art in production infrastructures in the last chapter revealed that service-oriented technologies are commonly used to provide access to services for infrastructure end-users. Grid middleware, or more generally, Grid technologies, are nowadays designed as SOAs [217] (cf. Section 2.2.2). SOAs characterise the foundation for the architectural design of e-Science infrastructures and their technologies. The well-known OGSA offers a basic Grid reference model and architecture that is based on the concepts of SOAs. But before analysing OGSA in more detail the basic SOA foundations are reviewed with a particular focus on existing reference model principles and associated reference model elements. In this thesis, SOA is based on the definition in [217].

Definition 22 (Service Oriented Architectures) *A Service Oriented Architecture (SOA) is a well-known paradigm for organising and utilising distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.*

The previously mentioned focus on SOAs, as defined in Definition 22, narrows the field for identifying relevant reference models. As a consequence, the most relevant related work can be actually found in a SOA-related publication [217] released by the OASIS SOA technical committee. Since OASIS is a major SDO in the e-Science community, this publication is extraordinarily useful in the given problem space. It defines a *'Reference Model for Service Oriented Architectures'*, and its introduction offers clear answers to the question, *'What exactly is a reference model'* in the context of basic SOA environments. Therefore, the information in this document is taken as a basis in order to identify general reference model principles and related architectural elements. The broad term *'reference model'* for which we use the definition from the mentioned OASIS SOA reference model [217] as a basis is defined as follows:

Definition 23 (SOA Reference Model) *A reference model for SOAs is an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. A reference model is not directly tied to standards, technologies or other concrete implementation details, but guides more concrete derived architecture work elements such as the reference architecture that can be based on open-standards, specifications, or profiles.*

Definition 23 lists the key aspects of a reference model (e.g. entities, relationships, etc.) and also points to the significance of standards supporting its derived reference architecture. Reference models are not to be directly tied to any concrete standards or technologies that in turn seems to be in contradiction to the aforementioned success of TCP/IP leading to its practical significance. One of the major critics of TCP/IP was basically at the same time a key factor of its success, namely the focus of a production-oriented reference model based on existing concrete standard protocol specifications such as TCP and more notably IP. Hence, Definition 23 and the contradiction raise several other questions for the investigation of related studies. The first question is *'If an abstract reference model alone does not specify any concrete standard specification, how can one achieve a reference model with practical significance useful for production e-Science infrastructures and its concrete applications?'*.

Another follow-on question is *'Which pieces of software architectural design elements actually make an abstract e-Science infrastructure reference model that much more concrete in order to be used with real e-Science applications?'*. Answers to these important question can be found in the OASIS reference model document [217]. It represents a *'frame of reference'* by defining the whole ecosystem around an abstract reference model that guides numerous associated architectural elements as shown in Figure 3.1. The frame of reference illustrated will guide the architectural

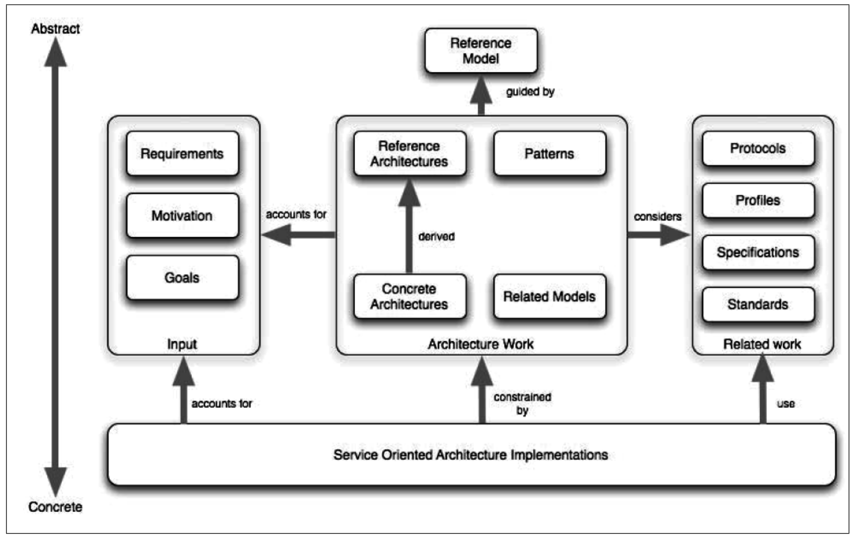


Figure 3.1: Overview of how a reference model relates to other architecture design elements [217].

design elements in the whole thesis and is therefore very important. The figure illustrates why many contents of this thesis are collectively defined under the *'umbrella term'* reference model by providing solutions for far more than the model alone on different levels. Many of the associated elements are necessary for real architectures and their applications.

Figure 3.1 provides an overview of the associated architectural design elements of a reference model in SOAs that can be mapped to thesis contributions. Several *'input'* elements have been already defined. The *'goals'* provide clear answers to the question *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. The *'motivation'* is introduced in Chapter 2, essentially enabling the freedom that e-Scientists can simply pick the services they need ignoring the boundaries of one or more production e-Science infrastructures. Concrete *'requirements'* will be provided in Chapter 4, after surveying related work in the field. The remaining architectural design elements are collectively named *'architecture work'*, *'related work'*, and finally *'SOA implementations'* within Figure 3.1. These are mapped in subsequent thesis chapters. The findings of this thesis offer a reference model along with reference architecture and concrete implementations that enable the use of applications with it. An abstract reference model is only one particular aspect of the whole thesis, while many other aspects in Figure 3.1 play a role in its definition to become gradually more concrete and thus useful for e-Science applications.

SOA Reference Model Key Principles

The comparison of existing reference models in the field starts with identifying some general key principles of reference models. The identification of such principles is not straightforward since many reference model definitions are available for a plethora of different specific fields in computer science. Also, known books on distributed systems, most notably Tanenbaum et al. [296], do not offer a reasonable general reference model definition and unfortunately focus more on the concrete ISO/OSI and TCP/IP reference model case. General principles are those that are *'not too specific'* for Grids and that are *'not too broad'* in order to still be applicable in Grid environments. Based on the introduction of the state-of-the-art, and the last paragraph, it is clear that SOAs matter in Grids since they are the blueprint for service-oriented technologies that offer Grid services (cf. Definition 13). In the context of these Grid services, open standards (cf. Definition 2.5) are highly relevant in the problem environment of interoperability for e-Science infrastructures. The OASIS SOA Reference model document [217] is a very suitable source that offers key principles that underpin Definition 23 and they are summarized in the following definition.

Definition 24 (Reference Model Key Principles) *The reference model key principles are abstractness, forming entities and relationships, in the context of a particular problem domain, and being technology agnostic.*

As defined in Definition 24, the principle of being *'abstract'* enables *'the development of specific or concrete architectures using consistent standards or specifications'* [217]. The first reference model principle is thus referred to as *'abstract'* meaning that the elements and concepts defined by the reference model are an abstract representation of its elements. An abstract reference model description as such describes the general *'service architecture'* (i.e. in e-Science infrastructures) and is not deployment specific for instances that can be found in realistic setups (i.e. not specific for EGEE/EGI or DEISA/PRACE).

The second principle is *'for understanding significant relationships among the entities of some environment'* [217]. Hence, another important principle is that a reference model should define *'entities and their relationships'* with each other. When only entities are listed, the relationships (e.g. layered like within the TCP/IP reference model or how standards work with each other) are neglected and as such the reference model and its derived architectures are less useful.

For the next principle, the focus is set on the latter part of the aforementioned sentence *'for understanding significant relationships among the entities of some environment'* [217]. A reference model consists of a couple of concepts but all of which are *'within a particular problem domain'* [217]. The principle of having a reference model for a specific environment with a *'clear focused problem space'* is also important. ISO/OSI provides an example that aims to solve problems for a specific field in communication. Another key principle of the reference model design that it is *'independent of specific standards, technologies, implementations, or other concrete details'* [217]. Hence, a reference model should be *'technology agnostic'* and thus make no assumptions about specific technology or standard implementation. This is in contrast to associated reference architectures where standards, profiles, specifications are used as shown in Figure 3.1. A reference model is a mechanism for understanding the problems faced, not the particular implementations involved.

The four aforementioned common reference model principles represent one valid foundational 'comparison metric' for relevant reference models in the field. This comparison metric is used in order to discover whether the existing models have common drawbacks. It is used to reveal why existing solutions failed to deliver a sustainable interoperability reference model of e-Science infrastructures or why those models are not used at all in the given problem domain. The reference model priniciples offer a theoretical point of view providing a foundation for comparing existing reference model elements.

3.1.2 Open Grid Services Architecture Analysis

As a complement to the previous, rather theoretical view on reference models key principles, this section focusses on the more practical community-specific lessons learned using OGSA as a basis. One decade ago, the technical field of Grid computing emerged with OGSA as the *'initial and fundamental Grid reference model and architecture'* proposed by Foster et al. [174] in 2003. Numerous related distributed systems publications and derived concrete architectures appeared that aimed at implementing the OGSA vision.

OGSA is therefore the basic reference model (and reference architecture) in this survey of related work. It serves as a valid basis for the identification of indicators that bear the potential to enable production e-Science infrastructure interoperability today. Such 'indicators' are defined as follows:

Definition 25 (Reference Model Design Indicators) *Reference model design indicators are facts about interoperability characteristica of a given reference model, its associated reference architecture, and their more concrete derived architectures.*

For a long time, OGSA was known to promote interoperability through a broad framework based on a wide variety of open standards. But despite being well-known and the basic model in Grids, only a slow adoption rate was observed over the years so that OGSA has not yet achieved a common basis for production e-Science infrastructures. OGSA is analyzed in more detail in order to explore the reasons for its slow adoption and to understand the major critics of OGSA that have hindered its broad adoption in the light of roughly one decade after its initial definition. OGSA is critically reviewed in [264], and this section identifies relevant factors based on indicators (cf. Definition 25) for a production e-Science oriented reference model and its associated architecture elements.

Definition 26 (Reference Model Design Factors) *Reference model design factors group several indicators as defined in Definition 25 in terms of SOA key requirements, scope and relevance to e-Science, applicability in production environments, open standards conformance, and uptake in existing production ecosystems.*

Since the beginning of the creation of OGSA during 2002 and 2003, critiques of OGSA have been continously discussed during the years within the community. But only a minority of these community critiques have actually been published, like the early work by Gannon et al. [181, 180]. These two contributions critically analysed OGSA with a particular perspective of its initial conceptual design and implementation around the nowadays outdated *Open Grid Services Infrastructure (OGSI)* [299]. One of the results of these and related critiques, as well as subsequent activities, was the deprecation of OGSI in favour of the more recent *Web Services Resource Framework (WS-RF)* [123]. In contrast to OGSI, WS-RF is still in use by some middleware systems today, like the Globus Toolkit 4 [167], Genesis [231] and UNICORE 6 [293]. Complementary to the aforementioned publications, OGSA is reviewed in this thesis from a production e-Science infrastructure perspective.

The analysis is based on the most recent definition of OGSA published by the OGF in 2006 under version 1.5 [174] that is in turn based on the evolution of the initial publication [171]. The analysis results are shown in Table 3.1, listing reference model and associated architecture interoperability factors. These factors are underpinned by identified indicators that are partly derived from OGSA critiques or production Grid experience in context (e.g. from concrete derived OGSA-based architectures). The identified factors and indicators are in turn used throughout the survey of related work for a comparison with other related approaches.

Identifying Service-based Reference Model Indicators

The most fundamental factor from OGSA 1.5 is essentially referred to as *'(a) Service based'* since the key architectural design in distributed systems like OGSA-conform Grids are SOAs (i.e.

Relevant Factors	Indicators	OGSA
(a) Service based (Reference Model)	(1) Service Oriented Architecture (SOA)-based design for a distributed system	Yes
	(2) Entities offer service interfaces	Yes
	(3) Clear unique service semantics	Yes
(b) e-Science Context (Reference Model)	(4) Focussed on scientific use cases	No
	(5) Grid execution management elements	Yes
	(6) Grid data management elements	Yes
	(7) Grid security elements	Yes
	(8) Grid information elements	Yes
(c) Specified relationships between different relevant functional areas (Reference Model)	(9) Information and Compute	No
	(10) Information and Data	No
	(11) Security and Compute	No
	(12) Security and Data	No
	(13) Information and Security	No
	(14) Compute and Data	No
(d) Details for implementation (Reference Architecture)	(15) Based on concrete Web services (WS)-based Architecture as SOA implementation	Yes
	(16) Concrete specifications with referenced portTypes (e.g. Operations) or schemas	No
	(17) Invariants and constraints for the use of information and security data exists	No
(e) Production-oriented (Reference Architecture)	(18) Number of core service entities is lower than 5 (model not too broad)	No
	(19) Definition of core entities that must be in place to form an infrastructure	No
	(20) Used normative specifications are already defined and publicly available	No
	(21) Project that implements core entities and relationships (funding exists)	No
(f) Standards based (Reference Architecture)	(22) Based on normative standard specifications	No
	(23) Specifications from real SDOs	No
	(24) No break of existing established standard specifications	No
(g) Adoption in e-Science production technologies (Derived Concrete Architectures)	(25) EGEE / EGI technologies	No
	(26) DEISA / PRACE technologies	No
	(27) Production middleware systems (e.g. ARC, gLite, Globus, UNICORE)	No

Table 3.1: Reference model interoperability factors and indicators in the context of OGSA.

Indicator (1)). This key design principle remains and thus this indicator is relevant for current production Grids. Within OGSA 1.5, these services are described with clear service interfaces at a rather high-level but while being sufficient detailed to understand their unique service semantics. This leads to the requirement for reference model entities that offer a good description of the service interface (i.e. Indicator (2)) while each of the services must have clear unique semantics (i.e. Indicator (3)) in the infrastructure (e.g. relevance of an information system).

e-Science Infrastructure Reference Model Indicators

The next important factor *'(b) e-Science context'* stands for a couple of indicators that are 'focussed on scientific use cases' (i.e. Indicator (4)). When e-Science was becoming more and more known, other models including e-Business, or e-Government also became interesting and therefore many use cases have influenced the architectures of the last decade in Grid computing. Hence, OGSA was also not only driven by e-Science but also by e-Business use cases, as the abstract of the OGSA 1.5 document [174] already indicates. More evidence is given in the OGSA use case documents [168, 252] that also includes business use cases in addition to scientific ones. This partly influenced its broad design, where several core service entities (e.g. self-management) and functionality (e.g. Service Level Agreements) are, while being relevant, not a major priority for current e-Science infrastructures. In contrast, the more and more emerging cloud computing paradigm [176] is taking up Grid computing concepts that seems to satisfy exactly e-Business and e-Goverment use cases as the Standards and Interoperability for e-Infrastructure Implementation Initiative (SIENA) roadmap indicates [87].

The lack of focus of the OGSA model on pure scientific priorities represents a hindrance for adoption in scientifically-driven production infrastructures. But in comparison with production Grids, the majority of services listed in OGSA 1.5 are required. These observations are transferred into indicators with service elements for Grid execution management (i.e. Indicator (5)), Grid data management (i.e. Indicator (6)), Grid security (i.e. Indicator (7)), and Grid information (i.e. Indicator (8)). The only one missing from OGSA 1.5 are resource management service elements, meaning aspects of resource models (i.e. WS-RF and stateful Web service models) being only rarely adopted in production Grids. This does not necessarily influence interoperability, because often the resource model is an internal implementation detail rather than an external service model and thus not transferred into an indicator.

Relationships between Different Technology Areas

The requirement for *'(c) Specified relationships between different relevant functional areas'* is also relevant. *'Relevant functional areas'* from OSGA are provided above but their difference in terms of handling within production environments is important. Not all are on the same layer or can be treated by technologies in the same manner. While they are all at the same level of importance, production e-Science infrastructure experience over years reveals that information and security are actually considered to be orthogonal to compute and data services. This important lesson is published in [267] that introduced the concept named *'plumbings'* that later also plays a role within this context for the reference model. For example, information must be available to the whole infrastructure about available compute and data services, while at the same time these services must be secured by an infrastructure wide solution at each level. This is encoded in the definition of the relevant indicators (i.e. (9)-(14)) shown in Table 3.1. In contrast, OGSA 1.5 basically defines all its services on the 'same service level' and only roughly mentions relationships across different functional areas that are by far not sufficient in the light of production infrastructure requirements. This is also the case for having a clear relationship regarding how information about security handling of available services can be obtained. Also, a clear definition from OGSA of how compute and data services actually interact beyond the usual data-stagings (e.g. storage interfaces, etc.) is also not available.

OGSA addresses these critiques about interactions and the others previously mentioned is the so-called *'profiling approach'* as described by the 'OGSA Profile Definition Version 1.0' [218] document. It outlines how normative OGSA profiles should be written for describing collections of specifications and their interactions. The OGSA profile document [218] reveals that the still missing normative definition of OGSA should be provided by a number of OGSA profile documents modelled along the lines of Web Services Interoperability (WS-I) Profiles [63] (e.g. WS-I Basic Profile [122] or WS-I Basic Security Profile [125]). The WS-I set of specifications itself are too general, because they do not address several specific issues of e-Science infrastructures and Grid security setups (e.g. remote job submission or identity delegation). But the idea was to define a similiar set of profiles that addressed Grid-specific functionality [218]. The approach is clear and provides benefits since it enables quite a flexible set of standards to be relevant for production infrastructures. But over the last years, only very minor profiles have been published as part of the OGF process, leaving OGSA basically undefined.

In more detail, the *'OGSA WS-RF Basic Profile 1.0'* [175] is very much focussed on the implementation of OGSA concepts at the resource level with WS-RF. As mentioned above, this adoption of WS-RF, and in turn this profile, was not as broad as initially intended and thus essentially did not manage to become a community agreed basic profile. Much more relevant until today is the HPC Basic Profile [154] that profiles the use of OGSA-BES [169], JSDL [115] and some of its extensions for HPC applications [197]. This profile is used in some middleware systems (e.g. UNICORE, GENESIS, etc.), but could be signficantly improved to satisfy HPC production environments thus not fulfilling the promise given by its name. OGF also released a few other profiles in the field of security [226, 227, 288], while all of them essentially specify very particular setups and do not address the important production requirement of attribute-based authorisation required for virtual organisations (cf. Definition 8). All profiles have not even defined the basic architectural aspects of OGSA and thus this 'profiling approach' is not taken as a major design principle in this thesis.

Details for Implementation Indicators

After the definition of OGSA, the Grid community raised an increasing demand on it for *'(d) details for implementation'* leading to the aforementioned OGSI. But as critics of OGSI reveal [31], it was too atomic and it was making changes to standards like the co-called *'gwsdl concept'* [299] that breaks the backwards compatibility with the well-known and broadly used Web Services Description Language (WSDL) [150] concept. After several years in existence, OGSI was marked as *'deprecated'* in favour of the WS-RF set of specifications [123]. But while the resource management aspects of WS-RF are much better, a lack of detailed service definitions apart from the stateful behaviour of services. The introduction of OGSA 1.5 acknowledges this gap, referring to this document as being *'one component of a set of documents that, over time, will fully define OGSA, both informatively and normatively'*. But after five years, during which initial Grids have evolved into fully-grown production infrastructures, there is no clear set of service definitions or specifications to fulfil the whole OGSA vision.

Based on these experiences over the years, there are a few relevant fundamental indicators added to Table 3.1. The core assumption that OGSA-conform Grids implement the abstract SOA concepts with the help of concrete Web service (WS) technologies (i.e. Indicator (15)), like those mentioned in the OGSA 1.5 introduction [174]. This kind of communication baseline essentially stands for XML-based message exchanges using the Simple Object Access Protocol (SOAP) [190] over HTTP(S). But the work in the field of WS-I [63], provided evidence that the agreement on using WS is by far not enough to reach the interoperability between two or more production infrastructures. This can be at least partly explained by the growing collection of technical WS-* specifications where it is very seldom the case that the same set of specifications (or profiles) is adopted by the same technology providers or Grid infrastructures. In this context, a community agreed reference model and architecture has the potential to guide such technology providers when clearly referencing those relevant WS-* specifications (i.e. Indicator (16)) that enable interoperability in those infrastructures relevant for those communities.

More recently, services in some Grids also consider RESTful services [291] that re-use a lot of HTTP concepts while being very general in its usage. But more specifications are needed to reach an equally powerful 'semantic richness' of WS-* specifications such as using REST for job submissions simliar to the level provided by OGSA-BES and JSDL. When interoperability needs to be achieved, many more specifications (e.g. security) together with REST need to be defined together to be of equal use like WS.

Much more relevant than RESTful approaches is greater awareness in the use of data exchanges in WS message exchanges. Hence, in addition to the basic message exchanges, many 'data content' exchanges such as those encoded in information model schemas (e.g. GLUE2 [113]) are important for interoperability. Such models, like the shipped security attributes (e.g. with SAML [142]), require a concrete definition of their use and are thus important for interoperability beyond the fundamental WS portType (e.g. operations) level. As a consequence, their use in interoperability setups between production Grids requires clear invariants and constraints (i.e. Indicator (17)) as published in [260]. Although OGSA indicates the relevance of these data, it does not provide clear constraints for their use in the architecture.

Production-oriented Indicators

All the aforementioned factors contributed to the slow adoption of OSGA in production Grids over the years. While it remains open wheather there will be full OGSA implementation in the future, it seems thats its scope is actually too broad to be specifically *'(e) production-oriented'* in the context of present production environments. The broad OGSA provides many services and approaches that not all are specifically useful for production. Therefore, the number of core service entities are set to be below five (cf. Table 3.1, Indicator (18)) actually referring to the clear scope of four services covering execution management, data, security, and information. While it is difficult to explicitly set this border to four core entities, it is still a reasonably good indicator of whether a Grid reference model focuses on the core technical areas instead of a wide variety of not production relevant entities.

Also added is the definition of which core entities (i.e. Indicator (19)) should be available to create a working infrastructure ecosystem to Table 3.1. A reference model needs to specify exactly which services are required to form a core infrastructure and not leave it completely open like OGSA did. OGSA 1.5 does not force the requirement that all its services are present in an OGSA system, but it also does not specify those core services that must be available to promote basic interoperability. It only states that there *'might be a core set of not null interfaces, standards, and common knowledge/bootstrap that services must implement to be part of an OGSA Grid'*. Production e-Science infrastructures, for example, require concrete execution and data management services that are well protected via concrete security services, that in turn all work based on up-to-date information obtained from information services. The very high-level and rather theoretical, broad view of an OGSA-conform infrastructure raises, for example, the demand for information services as only a minor point beside many others. Information services and common information models are the foundation in this thesis, and nearly equally important as the communication baseline (i.e. Web services) itself.

All in all, the broad definition aspect of OGSA actually leads to another drawback and critique that is transferred into an indicator (i.e. Indicator (20)). OGSA 1.5 defines a plethora of services where specifications are still not currently available and are also not expected to be available in the near future (e.g. resource selection services, fault detection and recovery services, standardised advance reservation services, etc.).

Finally, all the aforementioned points culminated in the indicator (i.e. Indicator (21) for whether there are projects that implement the whole reference model with its associated architecture. This would significantly increase the probability of being realistically implementable. Despite providing a basis for service and component re-use in e-Science, this indicator also covers a certain grade of evidence as to whether the reference model is practically implementable (e.g. TCP/IP [296]) or rather a purely theoretical reference model model (e.g. ISO/OSI [296]). The example of TCP/IP and ISO/OSI provides evidence that the interoperability is a good example of where the theorists and practitioners are completely polarised. Until today, no project has implemented, or is implementing, the whole OGSA ecosystem, except some middleware (e.g. GENESIS [231], UNICORE [293]) that implement parts of it.

Standards-based Indicators

As mentioned in OGSA 1.5, the key to the fulfilment of the OGSA vision is standardisation and being thus *'(f) standards based'* (cf. Definition 14). This is surely critical to creating interoperable and reusable Grid components and distributed systems forming e-Science infrastructures that are sustainable and where scientists can select their standards-based technology of choice. OGSA 1.5 addresses this need for standardisation by defining a set of core capabilities and behaviours that address key concerns in Grids. This high-level requirement of standardisation is a valid basis, but by using relevant indicators more evidence is provided of whether a reference model through its reference architecture considers open standards or not.

First, a reference architecture like OGSA should be clearly based on open standards, meaning that its core entities should be well-specified with normative standard specifications (i.e. Indicator (22)) developed in an open process. While this is relevant for scientific environments, there is also the requirement for specifications that are released from SDOs (i.e. Indicator (23)) like OGF, OASIS, IETF, or the World-Wide Web Consortium (W3C) [103]. This prevents vendor-locks via *'pseudo-de-facto'* standards enabling the freedom to choose technologies as required. Also, it is important that standards do not break backwards compatibility (i.e. Indicator (24)) with established standards, such as in the case with the Grid proprietary definition of *'gwsdl'* in OGSI [299] or the *'httpg'* protocol of the Grid Security Infrastructure (GSI) [170].

Indicators for Production Infrastructure Adoption

The final factor is related to more concrete derived architectures from the reference model and its adoption in *'(g) e-Science production technologies'*. As production environments are daily used by scientists, it is very unlikely that a completely new reference model will influence our given infrastructures. Hence, it seems reasonable to analyse more deeply which existing e-Science infrastructures and technologies have actually been influenced by OGSA 1.5 and its principles. Closer investigation on the services and setups available in the EGEE / EGI infrastructures (i.e. Indicator (25)) reveals in [256] that only minor aspects of OGSA have been adopted and a wide variety of concepts still are not available (e.g. stateful resource management like that within WS-RF, advance reservation services, etc.). As a consequence, EGEE / EGI is not yet a fully OGSA-conform Grid, because only basic OGSA services are available (e.g. OGSA-BES) while other OGSA features such as data management or common security are missing.

The same is actually true for DEISA / PRACE (i.e. Indicator (26)) that also shows limited adoption of OGSA services and principles. Despite some basic OGSA services that are deployed (e.g. OGSA-BES), not many services according to the full OGSA are actually implemented and deployed from the four major middleware systems (i.e. Indicator (27)) relevant for production e-Science infrastructures. Hence, these systems are all not yet fully OGSA compliant technologies. All in all, the aforementioned findings provide evidence that the OGSA adoption in e-Science infrastructures is low.

3.1.3 Component-based Approach Review

The alternative approach to standard-based reference models and OGSA is a wide variety of *'component-based approaches'* often used in a pair-wise setup between different production e-Science infrastructures. Such approaches using non-standard solutions such as adapters, hacks, workarounds that share in common that they are rather short-term oriented and not guided by an overall reference model or more concrete reference architecture that ensures interoperability more on the longer term. There are many 'interoperation approaches' in these approaches with serious limitations often leading to solutions that are not sustainable. In contrast, standard-based approaches are more feasible in providing sustainable solutions, but often take longer to realize. This section provides details about the key differences between both aforementioned approaches highlighting the drawbacks faced when working only on short-term solutions.

Transformation Logic to Describe the Problems Faced

The majority of the existing component-based approaches that are known to realize the interoperation of production e-Science infrastructures often share one overall limitation that is modeled as *'transformation logic'* in this thesis to describe the problems faced throughout the different approaches. As shown in Figure 3.2 it inherently represents a valid comparison criteria using drawbacks for the component-based approaches, that all have failed to deliver a sustainable interoperability for the e-Science infrastructures relevant in this thesis.

Definition 27 (Transformation Logic) *Transformation logic is a dedicated functionality within a component responsible for transforming one protocol or schema of type A into another protocol or schema of type B. It stands for a number of drawbacks, which are processing time overhead, error-proneness, maintenance overhead, complex handling of different deployment versions, and semantic loss.*

Definition 27 defines the criteria for the comparison of different approaches, as transformation logic stands collectively for a number of drawbacks influencing a given setup more or less depending on its extensive use. Transformation logic is one key limitation and raises serious concerns, among several others, especially in terms of sustainable interoperability solutions. The logic often gets more and more complex by an increasing amount of supported protocols and schemas and thus their implied technologies. The only advantage seems that often such approaches are rather easy to implement especially in pair-wise ad-hoc e-Science infrastructure interoperability setups. This is the reason for the wide variety of existing approaches that have become available over the last decade.

The drawback *'maintenance'* is relevant, because transformation logic is subject to many changes when proprietary incompatible protocols or interfaces, as well as schemas, change over time within the affected components. It is even more of a challenge when the same component is deployed on production infrastructures in various versions, thus adding another layer of complexity due to the requirement of supporting older interfaces and schemas in order to be backwards compatible. In the context of performance, the logic needs much processing time where parts of one protocol or schema have to be mapped to another protocol or schema. Correctness is an issue too since the transformation logic implementations are very error-prone and often the transformation only covers a subset of the original protocol or schema essentially leading to true semantic loss. The summary of the implied drawbacks of the transformation logic criteria are shown in Figure 3.2, that models one major limitation in existing component-based interoperability approaches.

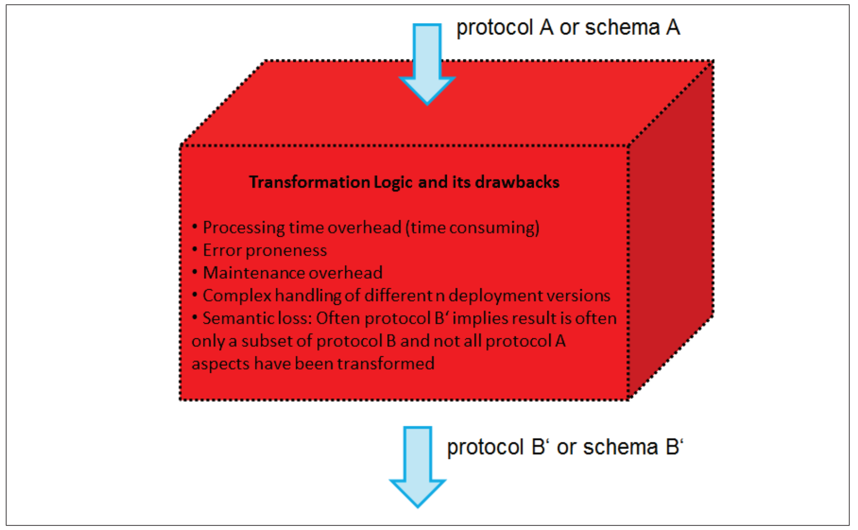


Figure 3.2: Transformation logic implies many drawbacks during its execution.

Open Standards avoiding Transformation Logic

Transformation logic as described above and published in [272] should be avoided to enable a long-term sustainable production-oriented interoperable solution. Such a logic can only be avoided when open standards are used within the different components, at least within the parts that are relevant to interoperability and interoperation with other components. But Standards are difficult to define and agree upon requiring often several years to become adopted. But without requiring transformation logic as illustrated in Figure 3.3 standards are the only path to reach sustainable interoperable solutions.

Differences between the both aforementioned approaches become more clearer when the different component-based approaches are surveyed in this section. The survey often identifies concepts that are particularly bound to certain components that can be found in a dedicated Grid technology deployed on a specific e-Science infrastructure. The concepts often thus only satisfy interoperation in a pair-wise fashion between specific technologies and thus is not a generic solution for more then the direct e-Science infrastructures involved. The survey of this related work represents a good source to learn about issues around the problem of e-Science infrastructure interoperability existing since many years. Many of these approaches only represent short-term solutions that have been created via workarounds, hacks, or non-maintainable component modifications (i.e. software tweaks on top of an official software release). But they all highlight certain point of interests (e.g. difference in job description formats) that are inputs to later parts of this thesis meaning essentially requirements and the reference model design.

The Benefits of Open Standards

Before the survey, the benefits of open standards as defined in Definition 14 are important to understand. The review of the benefits in context of the given problem space points to key

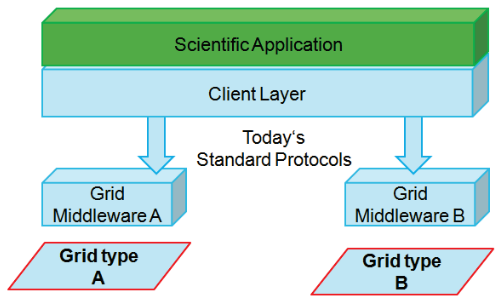


Figure 3.3: The open standards-based approach does not require transformation logic.

contributions to the vision of *'individual e-Science infrastructures'* (cf. Definition 20). Since the benefits are truly important they also influence later reference model design decisions. This section thus reviews the major benefits with a particular perspective of e-Science.

There is a wide variety of general benefits when technology providers (e.g. middleware providers), implement the same set of standards (e.g. the OGSA-BES job submission and management interface). One benefit is that the adoption of open standards prevents *'vendor-locks'*, since end-users of technologies can select the technology they want to use as long as this technology adheres to the same standard. The standard itself can be an agreed common interface, protocol, API, or even simple schema.

The key benefit relies in the possibility for end-users being able to access different services that adopt the same standard with the same client and security credentials. Administrators are better able to guarantee stability when standard-based service technologies can be switched while end-users can stick to the same client using the same standard adopted. Here the benefit relies on the change of underlying technologies without the necessity of exposing this change to end-users. This is an ideal situation where end-users do not even know whether there is a specific standard used and, more notably, whether the technology has been switched to another. All in all, the standard-based solutions are also easier to maintain then transformation logic approaches.

Another benefit is that it offers better protection against *'proprietary data silos'* created by a certain technology. Technologies can disappear over time (e.g. from lack of funding), but if they provide access to key data the access needs to be further guaranteed. If the data format is proprietary, time-consuming transformations (if possible at all) must be done from the old proprietary format into the format of the new technology to be used. This in turn can lead to significant migration costs that also includes the time of transformation and the transfer from data from source to the new sink. Standard-based middleware thus enables that valuable data (e.g. compute usage records) with formats that are standard-compliant can be correctly used by another middleware technology.

End-user application porting (e.g. using an MPI program with middleware) is also much easier from one middleware to another when standard protocols and interfaces are used. Apart from the functional interface, the standard itself follows with it certain guidelines and characteristics of approaches. In other words, standards can provide the benefit of stabilizing a *'way of using'* technology directly through particular standards (e.g. better understanding of certain parameters). Switching technologies is easier since the skills learned from one middleware can be utilised with less training than learning another proprietary interface and *'way of using'* (e.g. necessary MPI options such as required set of cores/threads).

Open standards are a key contributor to interoperability that complement the aforementioned approaches around reference architectures guided by a broader reference model. The use of this complementary nature is thus of major importance in this thesis and plays a role throughout the next chapters.

3.2 Survey of Related Reference Models

Apart from the already reviewed OGSA, there are not many reference models and reference architectures in the field of distributed systems and Grid computing that are fully applicable to the given problem space. This also partly contributed to the major motivations of this thesis in terms of investigating applicable models.

Models that have been published in the context of e-Science or those that are well-known in distributed systems are surveyed. The identified reference model interoperability factors and their indicators (cf. Table 3.1) are used to understand whether a model bears the potential to be applicable to the given problem space. The academic analysis of the models also identifies advantages and disadvantages of each model that in turn leads to the fullfillment (yes) or limitation (no) for each indicator. Table 3.2 provides an overview of the findings and the subsequent sections give more details. It shows that none of the existing models is able to satisfy all relevant factors and indicators.

This thesis considers the term *'reference model'* as an umbrella for a wide variety of different associated elements around a reference model following the basic ideas of the OASIS SOA

Relevant Factors	Indicators	OGSA	EGA	CSA	RM-ODP	CCA	CPN
(a) Service based	(1)	yes	yes	yes	no	yes	no
(Reference Model)	(2)	yes	yes	yes	no	yes	no
	(3)	yes	yes	yes	no	yes	no
(b) e-Science Context	(4)	no	no	no	no	yes	yes
(Reference Model)	(5)	yes	no	no	no	no	yes
	(6)	yes	no	no	no	no	yes
	(7)	yes	no	no	no	no	no
	(8)	yes	no	no	no	no	no
(c) Specified	(9)	no	no	no	no	no	no
relationships between	(10)	no	no	no	no	no	no
different relevant	(11)	no	no	no	no	no	no
functional areas	(12)	no	no	no	no	no	no
(Reference Model)	(13)	no	no	no	no	no	no
	(14)	no	no	no	no	no	no
(d) Details for	(15)	yes	no	yes	no	no	no
implementation	(16)	no	no	yes	no	no	no
(Reference Architecture)	(17)	no	no	yes	no	no	no
(e) Production -	(18)	no	yes	no	no	yes	no
oriented	(19)	no	no	no	no	yes	no
(Reference Architecture)	(20)	no	no	yes	yes	yes	no
	(21)	no	no	no	no	yes	no
(f) Standards based	(22)	no	no	yes	yes	no	no
(Reference Architecture)	(23)	no	no	yes	yes	no	no
	(24)	no	no	yes	yes	yes	yes
(g) Adoption in e-Science	(25)	no	no	no	no	no	no
production technologies	(26)	no	no	no	no	no	no
(Derived Concrete	(27)	no	no	no	no	no	no
Architectures)							

Table 3.2: Relevant reference models with comparisons of factors and indicators.

reference model [217]. As part of the survey also associated design elements are thus taken into account such as reference architectures, standards, goals, patterns, etc. (cf. Figure 3.1). This allows a more effective analysis on the overall usefulness of the reference model and its derived reference architectures or concrete architectures for each of the different models. All factors in Table 3.2 are based on the outcomes of evaluating indicators, clearly pointing either to yes or no. The OASIS SOA reference model is not part of this related work survey since it is introduced earlier and it is the baseline being thus very fundamental for this thesis.

3.2.1 Enterprise Grid Alliance Reference Model

The standardisation landscape of the Grid community was different back in 2006 than today. The Global Grid Forum (GGF), was largely driven by academic needs, and the Enterprise Grid Alliance (EGA) was driven by commercial and enterprise needs, also partly influencing OGSA as some business use case elements reveal [168, 252]. Since June 2006, both SDOs merged and formed the Open Grid Forum (OGF) that is the major SDO of the e-Science community today. The reference model working group of the EGA published a reference model that is defined in two documents [127, 128] and that is analysed in this section in detail.

The aforedefined general reference model principles as defined in Definition 24 are used to analyse the EGA model. It follows the *'principle of being abstract'* in the sense that it provides a common context in terms of defining the set of components to be managed within one enterprise Grid, their life cycle and how they are managed. Although being very specific in terms of functionality it makes no assumptions on dedicated deployments in one particular existing enterprise. It thus enables the development of more concrete architectures, but those have been not emerged yet.

As shown in Figure 3.4, the EGA model defines an enterprise Grid using mappings between abstract business application definitions and elemental Grid components representing thus entities and their relationships. It defines such *'entities and relationships'*, with very basic

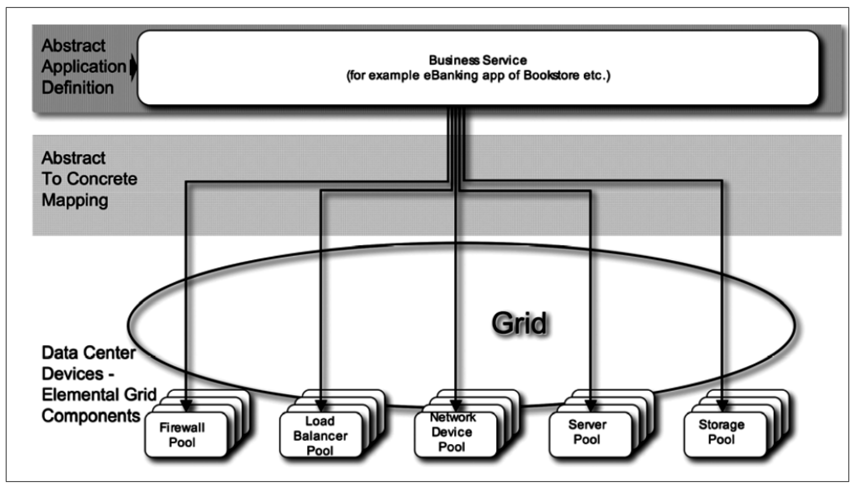


Figure 3.4: EGA reference model and some of its elemental Grid components [127].

IT components (i.e. server, firewall, etc.) that all contribute to a business application and are not e-Science specific. The model also enables to understand relationships between entities even if they are defined at a very high-level. The clearly *'focussed problem space'* of the reference model is defined as an enterprise Grid and thus only has very minor (specific) overlaps with entities and relationships of relevant e-Science technologies. The reference model is defined in a *'technology agnostic way'* using only general Grid components rather than specific technologies of specific vendors. The EGA model satisfies all the general principles and thus can be considered as a real reference model from a theoretical point of view.

From a more practical perspective, it would be important that the EGA model is in-line with the defined reference model interoperability factors and their identified indicators. The summary is provided alongside other models in Table 3.2. The reference model is modelled alongside the SOA design (i.e. Indicator 1) and several entities are introduced that offer service interfaces (i.e. Indicator 2) with clear semantics (i.e. Indicator 3). The EGA model is thus service-based and relevant to the given problem space.

But the EGA reference model is oriented towards e-Business and not focussed on scientific use cases (i.e. Indicator 4). It is concerned about component and service lifecycle thus lacking important reference model elements that are relevant in this thesis (e.g. Grid security). The idea of *'units of work'* [128] is introduced, but lacks sufficient detail on execution management (i.e. Indicator 5), Grid security elements (i.e. Indicator 7), and Grid information elements (i.e. Indicator 8). Despite the fact that the focus of the EGA reference model is services around data centres, it also lacks the significant Grid data management elements (i.e. Indicator 6) required in the given problem space.

Important other factors regarding the specified relationships between different relevant functional are not described in sufficient detail. The EGA model does not specify relationships required between relevant areas (i.e. Indicator 9-14) and only rather general relationships between elements are given.

With both of the aforementioned important factors being rather unsuitable for e-Science infrastructures, the EGA model as a whole lacks sufficient detail for its implementation too. There is no concrete reference architecture available for the EGA model as part of its two documents nor as a separate specification. Alhough SOAs are the basic concept in the document, it does not reference any concrete WS-based architecture used for the implementation of these SOA concepts (i.e. Indicator 15). Concrete specifications with portTypes are missing (i.e. Indicator 16) for its operations, as well as invariants for the use of information and security data (i.e. Indicator 17).

From the aforementioned academic results analysis it is clear that the EGA reference model is not oriented towards production e-Science infrastructures. Although the model is not too broad (i.e. Indicator 18), the documents give details on commissioning and decommissioning services, provisioning and event flow, lifecycle management or other aspects that satisfy several e-Business use cases, but that are not directly relevant for e-Science. It lacks the definition of core entities to form an e-Science infrastructure (i.e. Indicator 19) and does not make use of normative specifications (i.e. Indicator 20). There is no known project that currently implements the reference model (i.e. Indicator 21) with a more concrete reference architecture so that it could be used in production environments.

Nevertheless, the EGA model mentions standards, e.g. *'The EGA aims to drive interoperability through the use of standards'* [127]. But a more thorough analysis reveals that in an absence of a reference architecture, it fulfills none of our factors of being standard-based (i.e. Indicator 22-24). Some activity within the OGF Reference Model working group [66] was started to redefine some of the concepts to a more specific use of OGF standards, but this work was stopped years ago.

The whole e-Business-driven EGA model has an ambiguous scope and definition. It is thus not relevant in any form to the given production environments in this thesis since after several years it still lacks many relevant details. Current production infrastructures do not consider elements of this model (i.e. Indicator 25-26), nor are production middleware technologies capable of providing implementations that are guided by this reference model (i.e. Indicator 27).

3.2.2 OASIS Service Component Architecture

A general reference model from OASIS is called the SOA reference model [217] that provides a lot of fundamental pieces of information in this thesis based on SOAs. In addition to this model, OASIS also defines the Open Service Component Architecture (CSA) [71] as a set of specifications which describe another reference model with a more concrete reference architecture in context. In contrast to the rather general SOA reference model, CSA describes a more specific model for building applications and systems using SOA approaches that largely build on the Web services technology but also allow others.

The CSA is a model that encompasses a wide range of technologies in terms of the service components and access methods that are used to connect them. The overall idea is described in its assembly specification [129] that is one of many other specifications that collectively define the CSA. The general reference model principles as defined in Definition 24 are used to analyse the model in the next paragraphs.

It follows the *'principle of being abstract'* meaning that it makes no specific assumptions on deployment while this specification enables the development of concrete architectures. More concrete reference architecture elements are provided as an aligned set of specifications that even offer bindings for different languages (e.g. C++, Java, etc.). An overview of the different specifications in context to the rather abstract assembly specification is shown in Figure 3.5.

As shown in Figure 3.5, the CSA provides a huge set of *'entities and their relationships'* that are defined in a number of specifications, collectively published in March 2007 as CSA 1.0 specifications that are standardised within OASIS. The CSA itself is focussed on building SOA-based applications as composed networks of service components leading to composite service applications. The specification gives many hints on how the different entities relate to each other leading to a complex set of related specifications. The reference model has a very clear *'focussed problem space'* that is mainly business scenarios. It has a close relationship with the Service Data Objects (SDOB) set of specifications [72] that provide a unified model for the handling of service data in a service environment. The reference model is *'technology agnostic'* and thus makes no assumptions on concrete CSA implementations. It is vendor-neutral and supported across the industry being also language-neutral since components can be written using differ-

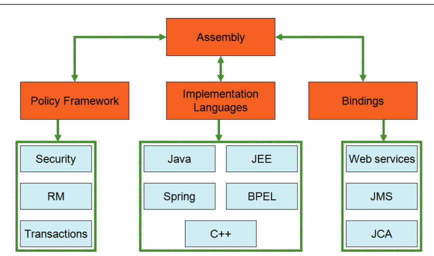


Figure 3.5: Specifications that collectively represent the OASIS Service Component Architecture (SCA) [71].

ent programming language bindings that are all provided in different specifications. From a theoretical perspective, the CSA reference model and its associated reference architecture elements can be thus classified as a reference model.

The more practical-oriented academic analysis in this section takes advantage of the factors and indicators defined earlier while the summary of results are shown in Table 3.2. The overall CSA reference model is based on the SOA design for a distributed system (i.e. Indicator 1). Entities are components or so-called composites that offer service interfaces (i.e. Indicator 2) while the specifications describe how these can be implemented in the corresponding languages. The general description of service interfaces also provide clear unique semantics (i.e. Indicator 3) in the context of the greater reference model although the services themselves are not defined in detail. OASIS CSA is service-based, even if its general applicability is described so general that it is not suitable for a reference model and associated reference architecture in e-Science infrastructures.

The reference model set of specifications is e-Business oriented and not optimised for an e-Science context (i.e. Indicator 4). Many ways in how services according to the CSA need to be defined are described, but it lacks certain concrete reference model elements that make its use feasible in the given problem space (i.e. Indicator 5-8). Some specifiations are related to relevant aspects, e.g. the security policy elements in [130], but it is far from being in-line with production e-Science use cases.

The CSA specifications lack the dedicated elements required in e-Science and therefore the important relationships between them are unspecified too (i.e. Indicator 9-14). A concrete derived architecture from the reference model can be created to satisfy e-Science needs, but it is not clear how the CSA then improves interoperability.

The reference model is very clear when it comes to details on implementations, since it is largely building on the Web services technology (i.e. Indicator 15). Concrete specifications with portTypes are provided (i.e. Indicator 16) and certain policy invariants on security and information are given (i.e. Indicator 17).

Although the details, by even providing language bindings, are very specific, it is not oriented towards production e-Science infrastructures. The required e-Science service elements are missing although the whole reference model is broad rather than compact (i.e. Indicator 18). This is especially true when also the related set of SDOB family of specifications is taken into account for real implementations. There is no definition for core service entities that must be in place to form an e-Science infrastructure (i.e. Indicator 19). Nevertheless, all specifications are publicly available from OASIS (i.e. Indicator 20), but there is no e-Science project that take the OASIS CSA as a basis for implementation in the given production e-Science infrastructure environments (i.e. Indicator 21).

The reference model including its associated reference architecture is standards-based because they consider several normative specifications (i.e. Indicator 22). Since the model is supported and standardised by OASIS, it is a reference model with specifications from a real SDO (i.e. Indicator 23) without breaking any existing standards (i.e. Indicator 24).

Because of the major aforementioend critiques there is no relevant adoption within the given problem space. This is the case not only for the production infrastructures (i.e. Indicator 25-26), but more notably for the middleware used by them (i.e. Indicator 27).

The CSA as such is a e-Business-driven model and very specific (e.g. WSDLs, etc.), but at the same time too broad (i.e. advice on how general services can be implemented) to be used in the production environments in the given problem space. CSA is a good basis for concrete architectures, but it is not precise enough how the associated reference architectures supports interoperability of e-Science infrastructures with Grid functionalities.

3.2.3 Reference Model for Open Distributed Processing

Another standard-based related work is the so-called Reference Model for Open Distributed Processing (RM-ODP) [82]. It is a joint effort by ISO and ITU-T [49], and provides a coordinating framework for the standardisation of open distributed processing (ODP). The reference model supports any form of ODP via the support of distribution, interworking, platform and technology independence as well as portability. 'Grid activities' in general and a wide variety of 'production e-Science infrastructure activities' in particular are closely related to *'open distributed processing'*.

The general reference model principles as defined in Definition 24 are used to analyse the RM-ODP model from the theoretical perspective. The four RM-ODP specifications (i.e. ITU-T recommendations and ISO international standards) make no assumptions on specific deployments and thus follow the *'principle of being abstract'*. Instead, the RM-ODP defines international standards that define essential concepts that specify open distributed processing systems with five abstract 'viewpoints' [141]. The reference model is too abstract in this context and very basic with respect to distributed systems design. To provide one example, Figure 3.6 defines a very general channel-based communication for distributed systems of the reference model including the usual building blocks for traditional remote procedure calls (i.e. stubs, binder, etc.).

The reference model is huge with many *'entities including their relationships'*. The RM-ODP defines a *'clear problem space'* by defining the environment as every service used in distributed processing using communication principles of distributed systems essentially referring to remote procedure calls. The reference model and its various supporting elements are *'technology-agnostic'*, pointing to no specific vendors or dedicated technologies. From a theoretical perspective, it is a very interesting model since the four core specifications and related elements are standardised in extraordinary detail by ISO. On the other hand, it is very basic in nature (e.g. clarifying terms, communication channels, etc.) and thus it is only of minor relevance to the very concrete problem space in e-Science.

Although the theoretical perspective of the RM-ODP point to several issues (i.e. too basic), a more practical perspective review is below based on the aforedefined factors and indicators.

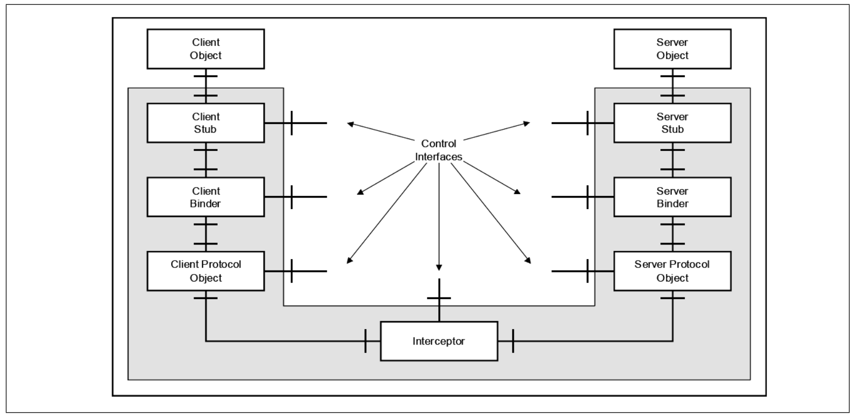


Figure 3.6: The RM-ODP is very basic but represents a strong ISO/ITU-T standard [141].

The summary is provided together with other models in Table 3.2. According to [203], the reference model *'is based on precise concepts derived from current distributed processing developments'*, but as it is very general it lacks a precise focus on SOA-based systems (i.e. Indicator 1). As the specifications speak about general systems, they lack a precise service interface definition for entities having clear service semantics (i.e. Indicators 2 and 3).

The set of specifications mentions pieces of information about enterprises, customers, and warehouses, and this indicates that the major concepts are more e-Business-based than e-Science-based (i.e. Indicator 4). Also, no particular e-Science relevance can be found around execution and data management (i.e. Indicators 5 and 6) as well as security or information (i.e. Indicators 7 and 8). Although information elements are partly addressed, it provides insufficient details for a particular production e-Science infrastructure context.

Whilst there are plenty of relationships between several parts of the specification, RM-ODP lacks specific relationships between required relevant functional areas in e-Science technologies (i.e. Indicators 9-14).

In terms of details for implementation, a lot of information is provided. But these pieces of information are all related to very foundational concepts rather than specifically important aspects such as a specific Web services architecture implementation (i.e. Indicator 15). Concrete specifications with portTypes are missing that would be required to implement RM-ODP in given e-Science environments (i.e. Indicator 16). Although general elements like *'security functions'* [203] or *'information viewpoints'* are provided there are no concrete constraints or invariants that can be used to perform a valid implementation (i.e. Indicator 17).

Based on previous findings, the reference model is not production-oriented in terms of concrete e-Science environments. Not only is the RM-ODP a very broad model (i.e. Indicator 18), it is also not clear about which exact services must be in place to run a core e-Science infrastructure (i.e. Indicator 19). Despite of these disadvantages, the majority of specifications are normative specifications that are publicly available (i.e. Indicator 20). But there is no project in the e-Science community that implements these concepts and would contribute to the likelihood of its production impact in practical implementations (i.e. Indicator 21).

So far, many relevant reference model factors in the context of this thesis pointed to the fact that RM-ODP is not suitable. But the RM-ODP is based on normative standards (i.e. Indicator 22) and has specifications from high impact SDOs (i.e. Indicator 23). As such it is one of the most detailed standardised reference model examples without any break of fundamental existing standards (i.e. Indicator 24).

All the aforementioned aspects already point to the non-existing adoption in e-Science production technologies. The reference model or its associated architecture specifications is not adopted in EGEE / EGI technologies (i.e. Indicator 25) nor is it relevant to DEISA / PRACE technologies (i.e. Indicator 26). Also, with respect to adoption in production middleware, ARC, gLite, Globus or UNICORE do not take RM-ODP into account (i.e. Indicator 27).

RM-ODP is very general and the reference model is in part even old with specifications that date back to 1994. Based on this academic analysis it is clear that this reference model has only minor relevance to the given e-Science environments in this thesis.

3.2.4 Common Component Architecture

The Common Component Architecture (CCA) forum [4] defines a minimal set of standard interfaces within e-Science environments, with a particular focus on traditional scientific computing (cf. Chapter 2). The overall idea of this reference architecture is to bring the benefits of component-based software engineering to HPC application developers. The design is oriented to preserve the performance of components running in these specialized environments. Its major feature is to incorporate existing HPC codes into environments driven by the CCA reference model. The whole model and its reference architecture is thus low-level (i.e. source code level) towards the specific use on HPC resources.

The general reference model principles as defined in Definition 24 are used to analyse the CCA model in the following paragraphs. Although the CCA is such low-level, it is *'abstract'* in the sense that it enables the development of specific architectures that are not bound to any concrete deployment. It is applicable only in HPC-based environments defined by several CCA elements, as shown in Figure 3.7. The use within distributed systems in the sense of Grids is very limited since it neglects HTC-based systems and distributed data-driven elements.

Figure 3.7 illustrates the relationships between the CCA elements that are required to establish component-level interoperability. Each of the CCA components defines data inputs and outputs via a scientific IDL, while these are used in conjunction with a repository using a CCA repository API. Definitions within this IDL are used as input to a proxy generator that generates component stubs, which in turn are the component-specific parts of *'GPorts'* [119] (i.e. CCA ports in Figure 3.7). GPorts encompass all the functionality necessary to organise component interactions within a CCA-compliant framework and thus define a uniform model of component interactions. The overall architecture clearly defines the *'entities and relationships'* in order to understand the significant relationships among its entities. The *'problem space'* is clearly defined since the CCA is a model with reference architecture for environments that make specific use of HPC with scientific applications. It is very low-level but still *'technology-agnostic'* by not being designed towards the use of one dedicated tool, compiler, etc. The whole CCA model is thus only partly applicable within the given problem space and lacks other necessary important concepts (i.e. distributed system w.r.t. HTC and data).

From the theoretical perspective, the applicability of the model in the given environment of this thesis does not promote interoperability across infrastructure resources or services. Analysing the CCA model from the more practical perspective highlights other facts using the aforedefined reference model factors and indicators as a basis. The summary of the CCA analysis is provided together with other surveyed models in Table 3.2. CCA is closer to the SOA-based basic paradigm (i.e. Indicator 1) than ascertained by the initial analysis above. The CCA components can directly use framework services through a so-called *'CCA Framework*

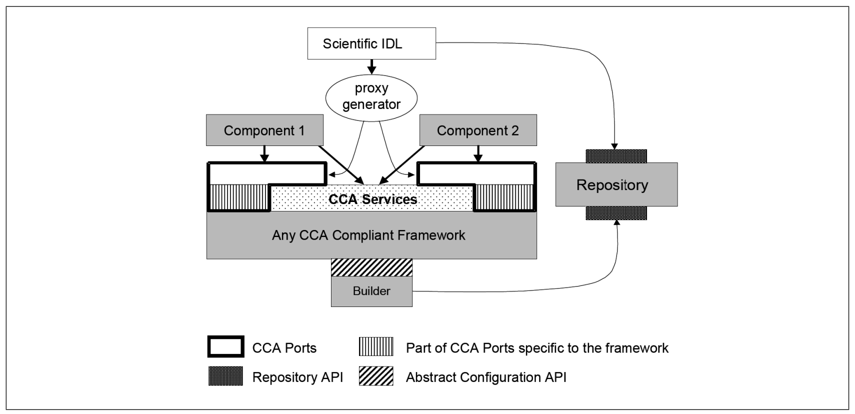


Figure 3.7: Elements of the CCA including their relationships to enable component-level interoperability [119].

Services interface' [119]. Specifying such interfaces enables a clear definition of the service entities (i.e. Indicator 2) that a CCA framework implementation needs. This in turn enables a common platform for any kind of component. Examples of such services are communication, security, thread creation and management, memory management, and error handling. Hence, the service semantics are very clear and understandable (i.e. Indicator 3).

In traditional HPC, many scientific applications are assembled from large blocks of hand-crafted codes within monolithic applications, and software re-use is only obtained by linking software libraries. The major disadvantage with this approach is that the software boundaries (e.g. global function interfaces) are not well defined, which in turn leads to internal code dependencies which make such monolithic applications difficult to modify and maintain. To overcome these limitations, several e-Science use cases (i.e. Indicator 4) pointed to a component-based programming model. This enables the development of applications built from modularised pieces, that in turn raise the requirement for a larger CCA framework. Although partly relevant in an e-Science context, elements on a higher level such as Grid execution management, Grid data management, security or information elements (i.e. Indicators 5-8) are missing.

The CCA framework provides the glue that binds the different components together, and that can be used to combine components from any component pool into a non-monolithic application. The CCA framework includes certain entities and relationships but none of them are relevant to the technical areas in the given problem space (i.e. Indicators 9-14).

The CCA provides necessary details for implementation such as the *'builder concept'* [119], also shown in Figure 3.7. The *'CCA Configuration API'* [119] provides functionality necessary for components to interact with the builder. Examples of such functionality are notifying components that they are added to a program, redirecting interactions between components, or notifying the builder of a component failure. But no concrete WS implementation details are available (i.e. Indicator 15) and no specification with relevant portTypes and operations (i.e. Indicator 16) exists. CCA does also not provide any details on security or information constraints (i.e. Indicator 17) for distributed systems.

The model itself is production-oriented since it is well specified, but only on a very low level (i.e. source-code level). Figure 3.7 illustrates that the number of core service entities are lower than 5 (i.e. Indicator 18) and with them being defined it is, in principle, possible to form a CCA-based HPC infrastructure (i.e. Indicator 19) that is part of e-Science. The CCA forum also provides normative specifications that are publicly available and well defined (i.e. Indicator 20) and several projects implemented it (i.e. Indicator 21). Examples are in the Climate community [211], that implemented the CCA considered for production use.

Although CCA is considered as 'standard' [119], there are no normative standard specifications (i.e. Indicator 22) mainly because the specifications are not released from a real SDO (i.e. Indicator 23). But it does not break existing established standards (i.e. Indicator 24).

The adoption within EGEE / EGI (i.e. Indicator 25) is unlikely due to the HPC focus of the CCA model. The model could be relevant for DEISA / PRACE environments but also in this context there are no broad adoptions known in the e-Science community (i.e. Indicator 26). Adoptions in middleware technologies are also not existing (i.e. Indicator 27).

3.2.5 Coloured Petri Nets Reference Model

Another rather academically-driven, Grid reference model is described by C. Bratosin in [138]. The motivation behind this model is a good theoretical conceptual model for the Grid that allows for precise evaluations. It provides a formal description of a Grid reference model in terms of the coloured Petri Nets (CPNs). Petri Nets [253] are a graphical formalism, able to model concurrency, parallelism, communication, and synchronisation. CPNs extend Petri Nets with data, time, and hierarchy, and combine their strength with the known strength of programming languages. CPN is thus a suitable 'formal approach' for modelling Grid reference architectures with their reference model.

The general reference model principles as defined in Definition 24 are used with the CPN as follows. Although the CPN reference model is formal and resolves ambiguities and provides semantics it also follows the principle of being *'abstract'*. An overview of the basic reference model design and reference architecture is illustrated in Figure 3.8 that consists of three distinct layers namely the application, middleware and resource layer. For each of these layers submodules exist to describe the correponding context.

The reference model is specifically tuned for process-mining applications [138] and that is used in one dedicated simulation environment. It does not make concrete assumptions on specific deployments, but assumes many other details (e.g. executables always pre-installed, data catalogue exists, global resource information database, etc.). Although formally the reference model is well described using a known layered-approach, it lacks a precise definition of *'entities and their relationships'* between each other. It has a *'clear focussed problem space'* that is supporting scientific applications with computationally-driven resources as well as some data capabilities. The scope of the reference model is thus very close to the given problem space in e-Science. It is also *'technology-agnostic'*, being driven by theoretical simulation measurement environments but unfortunately only with a 'future aim' to create production middleware [138]. Despite good aspects of the model, the theoretical analysis of the CPN model clearly reveals a lack of entities and relationships to form any production infrastructure or architecture.

The following paragraphs describe a more practical perspective using the defined reference model factors and their indicators. The summary of the CPN analysis is provided alongside other models in Table 3.2. According to the [138], the reference model seems to be service-based, but very little information about the SOA design aspects can be found in context (i.e. Indicator 1). Hence, the CPN model is not service-based in the sense of providing reference model entities with service interfaces (i.e. Indicator 2) and clear semantics (i.e. Indicator 3).

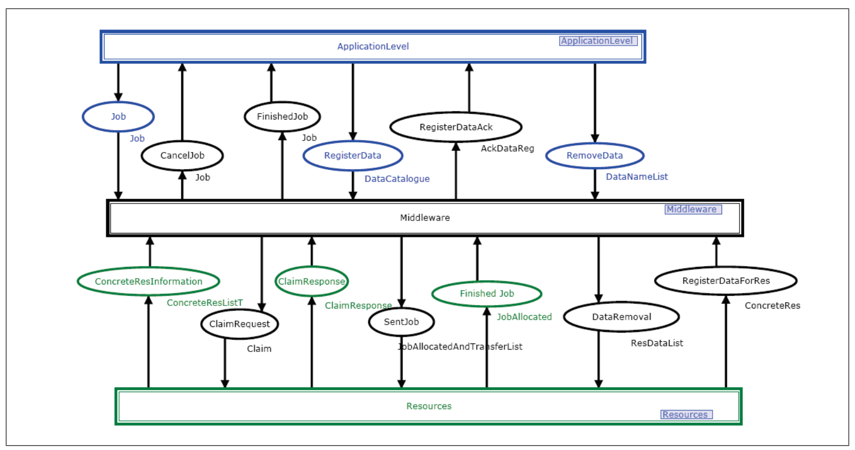


Figure 3.8: CPN reference model design with the application, middleware, and resource layer [138].

It takes into account use cases from e-Science environments (i.e. Indicator 4) and scientific applications. The e-Science context is also shown in Figure 3.8 pointing out that middleware is the centric concept in e-Science today. The CPN model includes roughly defined elements for Grid execution management (i.e. Indicator 5) and Grid data management (i.e. Indicator 6), but lacks details about important security elements (i.e. Indicator 7) and information elements (i.e. Indicator 8).

It defines relationships within one functional area (e.g. compute), but none between different functional areas (i.e. Indicators 9-14). But the relationships are rather use case driven and functionality driven such as with scheduling in the compute context. They do not reflect relationships between the reference model elements in general.

It is detailed enough at a formal level, but implementations within Grid middleware is not possibly because of not accurate interfaces. The CPN model is very vague in this regard. Concrete WS-based architecture elements to implement SOA concepts (i.e. Indicator 15) are missing. Concrete specifications with operations (i.e. Indicator 16) are also not provided although, as part of the three distinct layers, some hints on interface operations are given (e.g. removeDataFromRes()) [138]. There are also no concrete, necessary constraints for information flow or security details (i.e. Indicator 17). The only statement that can be given is that *'the place GlobalResInformation models an information database containing the current state of resources'* [138]. But such a high-level statement needs to be underpinned with much more information before it becomes realistically implementable.

The reference model is not production-oriented towards e-Science infrastructures. But [138] reveals that their model is not only suitable for validation purposes but also for conducting preliminarly simulation experiments based on process mining applications. Without concrete core service entities (i.e. Indicator 18), the model lacks details on which services are required to form a CPN-based e-Science production infrastructure (i.e. Indicator 19). No normative specifications (i.e. Indicator 20) are provided, and project implementations of core entities and their relationships (i.e. Indicator 21) are not existing in the given problem space.

Another significant drawback for a generally acceptable reference model with an associated reference architecture is the absence of standard specifications in the CPN model (i.e. Indicator 22). No specifications from real SDOs are referenced (i.e. Indicator 23) that would point to the exact specification of core building blocks of the reference architecture. Backwards compatibility with existing specifications should be still possible though (i.e. Indicator 24).

Adoption of this reference model in production e-Science infrastructures such as EGEE / EGI or DEISA / PRACE is not feasible (i.e. Indicators 25-26), because of the lack of many aforementioned significant details. Middleware adoption (i.e. Indicator 27), is unlikely, because these significant details have to be improved within CPN. This includes but is not limited to reflecting much more production e-Science infrastructure behaviour and its relevance instead of being a purely theoretical model only tested in simulation environments.

The CPN model clarifies basic Grid concepts at a conceptual level using three tiers, but it has no practical relevance at all for the interoperability between existing production e-Science infrastructures. It is thus not relevant in this thesis.

3.3 Classification of Component-based Approaches

Complementary to the previous section and its academic analysis of existing reference models, this section surveys the more practical-oriented approaches. The survey is published in [272] and based on practical field studies that aim to take advantage of interoperability between various components deployed in different e-Science infrastructures. Many of these approaches are done in a pair-wise fashion between those components and infrastructures. Many projects and approaches are listed as part of this section, but also Chapter 6 provides insights of transformation logic for the three accompanying case studies of this thesis. For the three case studies WISDOM, VPH, and EUFORIA, Chapter 6 reveals much more details with respect to limitations and provides information in context how these limitations can be solved using the proposed reference model and associated architecture work presented in Chapter 5. Nevertheless, this section gives first insights into the issues and different approaches when interoperability is not guided by a broader reference model approach.

A similiar survey of related work is in Field et al. [165] that also lists production-oriented approaches in the context of e-Science. This work is based on earlier work also published by Field et al. in [164]. In contrast to these contributions, this section and the survey presented in [272] is broader and focuses more on transformation logic as defined in Definition 27. The model of these limitations helps to identify specific problems that are a valuable input to the proposed reference model design later in this thesis.

Table 3.3 provides a classification of the approaches, while the subsequent sections provide details where the transformation logic exists in the used components and what the drawbacks of this approach are. It presents a classification of known component-based approaches towards interoperation that all use the aforementioned transformation logic (cf. Definition 27) in some form or another. All these approaches do not rely on any open standards implemented as native interfaces within middleware and thus this classification also surveys the problems that occur when standards are not used. All of these approaches share the same problems by using transformation logic including its drawbacks as defined in Definition 27. It is even worse in the given context where production infrastructures deploy different versions of the same components at the same time. Only one approach in Table 3.3, is different from the others that is part of this survey for the sake of completeness. This approach is named as *'middleware co-existence'* and does not requires transformation logic. At the same time it is not a solution to the interoperability problem, but an approach that provides a workaround for many interoper-

Approach	**Transformation Logic Location**
Additional Layer	Located in an additional layer on top of middleware
Neutral Bridge	Located in neutral bridge to contact all middleware and is contacted by one dedicated neutral protocol
Gateway	Located in the central gateway contacted with every protocol and can contact any middleware system
Mediator	Located in a mediator component being able to contact many middleware and is contacted with one dedicated protocol
Adapter	Located in one middleware in order to contact another specific middleware
Middleware co-existence	*n/a*

Table 3.3: List of component-based approaches and their transformation logic locations.

ability problems. Nevertheless, the challenges are just shifted into other related fields such as middleware maintenance and support and is thus not a real solution to the problem.

The focus of the survey is mainly related to the functionality surrounding the core elements defined as part of the reference model factors (cf. Table 3.1). Relevant for e-Science infrastructures is thus the focus on Grid execution management as well as Grid data management elements and their interoperability challenges (e.g. different job description languages). Complementary to these aspects, the focus is also on security and information elements and their interoperability challenges within components (e.g. different information models).

3.3.1 Additional Layer Concepts

The first concept of enabling interoperation between two different technologies is the additional layer concept. Academic analysis of existing work in the field reveals that this concept is often used together with a common portal or client that represents the additional layer. The use of two (or more) technologies within this additional layer is enabled through transformation logic that is inside this layer to transform different protocols and schemas. The additional layer is virtually on top of different Grid technologies and thus technically provides access to different types of e-Science infrastructures using transformation logic. The concept is illustrated in Figure 3.9 in which the additional layer is able to interact with Grid technologies (A, B,... n). that use different schemas and/or protocols.

The additional layer concept is often used by Grid clients, portal technologies and higher level APIs. For example, it is implemented in the GridSphere [240] Grid portal technology and used to access different Grid middleware technologies (Globus, UNICORE, gLite, etc.), thus allowing access to different types of e-Science infrastructures.

The transformation logic within GridSphere consists of transformations between different kinds of job description languages such as Globus Resource Specification Language (RSL) [167], UNICORE Abstract Job Object (AJO) [294], or gLite Job Definition Language (JDL) [212]). Three different job control and data control interfaces for these three middleware systems are also supported as part of this transformation logic. More recently, GridSphere is based on the Vine toolkit API [278] also using 'transformation logic', but gradually implementing more standards.

Similiar transformation logic is often included in higher level APIs like JavaGAT [239] and the Grid Programming Environment (GPE) [251] being also part of the additional layer concept. The transformation logic is part of these API technologies that have different proprietary adapters to access several different Grid technologies and thus e-Science infrastructures.

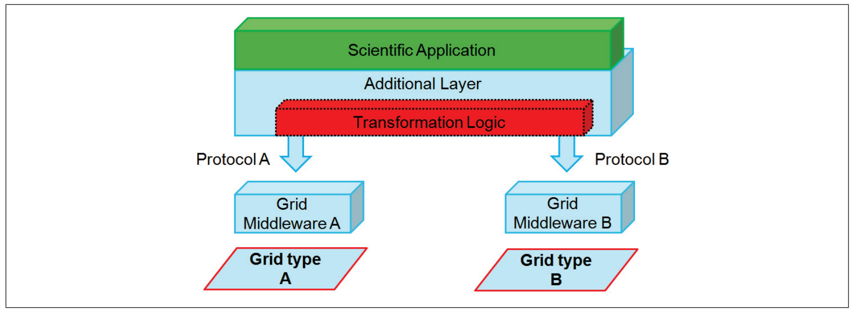


Figure 3.9: The additional layer concept uses a layer on top of different technologies with transformation logic.

Another example of this approach by Field et al. can be found in [163] in the context of Grid information systems. In this contribution, the approach is used to enable information exchange of the production Grids EGEE and NorduGrid that both adopted different Grid technologies (gLite and ARC) including different Grid information models. EGEE used the GLUE 1.1 information model while NorduGrid used a proprietary information model that was known as the 'NorduGrid model' [163] in the past. The Lightweight Directory Access Protocol (LDAP)-based [196] information system is used in both cases and the information schema was different and transformation logic was in the form of translating information providers within a kind of additional layer. These transformed the GLUE 1.1 and NorduGrid schema to a native common format that in turn provided the information for the LDAP-based information system.

Use Case WISDOM and its major hacks, tweaks, and workarounds

WISDOM [102] is one of the three major case studies of this thesis that aims to take advantage the different e-Science infrastructures EGEE/EGI and DEISA/PRACE with one broader scientific workflow. Detailed information about workarounds, tweaks and limitations can be found in [259] and [270] as well as in Section 6.3 that focusses on this case study and also describes how limitations are addressed by thesis work. The aim of this section is thus just to present a brief overview of the major hacks, tweaks and experiences of the WISDOM use case.

The WISDOM use case is another example of using the 'additional layer' approach using the proprietary UNICORE atomic services [293] together with gLite proprietary interfaces of the Computing Resource Execution and Management (CREAM) [212]. This additional layer of bridging both systems is implemented in the WISDOM scientific portal using GridSphere [240] and the VINE toolkit [278]. Transformation logic was necessary to transform the end-user job submission requirements into different job description languages that have different ways of specifying executables. Low-level information about the installed executables such as FleXX [30], or Assisted Model Building with Energy Refinement (AMBER) [242]) was needed to configure and tweak the job subscriptions so that they actually are able to be executed on EGEE/EGI and DEISA/PRACE. For the submission of these jobs, 'transformation logic' was used in VINE and GridSphere to use different security credentials for both middleware, including different job management interfaces and protocols.

Other applied workarounds via scripts have been performed to enable 'manual data-staging' that enables WISDOM e-Scientists to evaluate outcome of EGI/EGEE jobs and to enable the transfer of a subset of this data to DEISA/PRACE for computation. But also in many cases, the e-Scientists themselves have to perform many manual steps on the low-level of machines that could have been better supported with Grid middleware. One example in context is the location of the job sandbox directory in both EGEE/EGI and DEISA/PRACE in order to evaluate results. Phone and e-mail communication was partly necessary to communicate the location of the job session directories.

Many complex UNIX-scripts have been used to work with AMBER in DEISA/PRACE particularly for the reasons of having job sequences. That means different AMBER executables run after another within the same job sandbox enabled with complicated UNIX-scripts within the submission process that are error-prone. More details about this issues can be found in [193] including also a more thorough description why sequences are different from Grid workflows. On top of those issues, the versions of AMBER on different resources was needed, but not easy to retrieve and environments on the low-level machine need to be pre-configured by the e-Scientists again and again.

The summary of the major hacks, tweaks, and workarounds of the WISDOM use case are listed in Table 3.4.

No.	Short description of use case challenges
(a)	Transformation logic was necessary in the additional layer to convert between different job description languages, job submission and management interfaces and security credentials
(b)	E-mail communication was used to communicate Grid sandbox directory location to enable the evaluation of results of the EGEE/EGI job outcomes and to manually transfer only a subset of the data after evaluation to DEISA/PRACE
(c)	Error-prone large UNIX-scripts have been used to enable AMBER job sequences and their creation process need to know many low-level AMBER configuration aspects (e.g. executable locations)

Table 3.4: Summary of major tweaks, hacks, and workarounds in the WISDOM use case.

Use Case VPH and its major hacks, tweaks, and workarounds

VPH [92] is another of the three major case studies of this thesis that aims to take advantage of different e-Science infrastructures such as the NGS of EGEE/EGI and DEISA/PRACE. Detailed information about workarounds, tweaks and limitations can be found in [263] and [270] as well as in Section 6.4 that focusses on this case study and also describes how limitations are addressed by thesis work. The aim of this section is thus just to present a brief overview of the major hacks, tweaks and experiences when working with the VPH use case.

Also the VPH use case is one example of using the 'additional layer' approach using the proprietary UNICORE atomic services [293] together with the GridSAM [213] proprietary interface. This additional layer of bridging both middleware systems is implemented in the VPH scientific client tool called Appliation Hosting Environment (AHE) [219] and thus its client situation is slightly different than the one of the aforementioned WISDOM use case. But transformation logic is also necessary to transform the end-user job submission requirements into different job description languages that have different ways of specifying executables. Information about the large-scale HPC resources involved was cumbersome to collect (e.g. E-mail communication) and to use this information in turn with an MPI code together with Grid methods was a long tedious process. Many manual interactions with the corresponding HPC resources have been necessary to exploit possible hardware configuration options and available features that have been required for the run of an MPI code named HemeLB [219] in different varieties on different NGS or DEISA/PRACE resources. The information was present in different formats following no common information model and thus even for the same HPC architectures sometimes the scripts have been again tweaked in a complicated manner since the understanding of the different formats was not easy. Tweaked scripts have been created with this information and the submission of these jobs used 'transformation logic' within AHE to use different security credentials for both middleware, including the use of different job management interfaces and protocols.

On top of those issues, there was a clear lack of having compilations directly supported in middleware in a standard way since also the method for compiling required manual resource access, knowledge of different local compilers, and different security information (i.e. no single sign-on). In this context, some error-prone scripts have been created that failed to deliver the maturity needed for production and as such SSH was still used throughout the whole process in combination with the AHE tool and middleware underneath.

The summary of the major hacks, tweaks, and workarounds of the VPH use case are listed in Table 3.5.

No.	Short description of use case challenges
(a)	Transformation logic was necessary in the additional layer to convert between different job description languages, job submission and management interfaces and security credentials
(b)	Knowledge about HPC resource architecture details (e.g. network, shapes) on systems available in NGS and DEISA/PRACE was manually collected from Websites or retrieved by long E-mail communications with administrators and the use of this information with Grid methods is based on tweaked scripts
(c)	Information received from large-scale HPC resources as part of e-Science infrastructures was in different formats and syntax even for those with the same architecture leading to error-prone scripts
(d)	Compilation on different sites encoded as part of the job submission process via scripts was very error-prone and cumbersome for e-Scientists and as such SSH was used in combination with AHE

Table 3.5: Summary of major tweaks, hacks, and workarounds in the VPH use case.

3.3.2 Neutral Bridge Concept

The neutral bridge approach introduces a neutral protocol that can be used by clients in order to become independent of any schema and protocol changes in the underlying technologies. Many approaches in the field of technology interoperability are using such a neutral protocol to contact a 'neutral implementation' entity that is often named as a *'bridge'*. A bridge uses transformation logic to transform the neutral protocol in the different proprietary protocols for each of the technologies as shown in Figure 3.10. This enabled in turn the interaction with Grid technologies (A, B,... n). These transformations and thus changes are well encapsulated from the neutral client that uses the neutral protocol and only affect the bridge.

In [202], Jha et al. describes Grid interoperability on the application-level using the Simple Grid Application API (SAGA) [186] OGF standard. This approach uses the SAGA standard as a neutral protocol that in the SAGA implementation use different middleware adapters. SAGA only provides a POSIX-style API/protocol to the most common Grid functions, thus representing a rather Grid middleware-agnostic neutral protocol for use by 'neutral Grid clients'. The

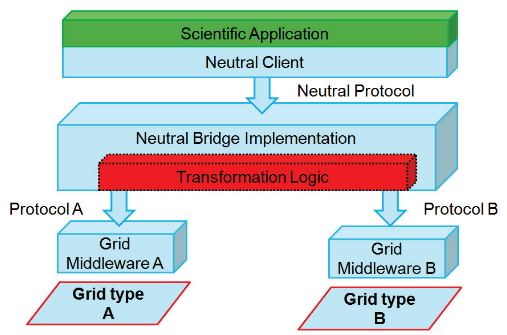


Figure 3.10: The neutral bridge approach uses a Neutral Bridge implementation with transformation logic.

neutral bridge itself and its SAGA implementation provides the functionality in order to submit and handle different Grid job descriptions and control interfaces. This functionality is provided with SAGA middleware adapters representing the transformation logic. They interact with different Grid technologies using the native protocol of the corresponding technology. Although SAGA as open standard is used, it does not solve the interoperability problem, because this standard is not currently natively supported within the middleware systems.

Another example is described by Stone et al. in [292], describing the interoperability between the not broadly used 'Integrade middleware' [292] and Globus. In this approach, the so-called 'switch' [292] component uses a neutral protocol for communication thus acting as a neutral bridge on top of Integrade, Globus, and potentially many other middleware systems.

Use Case EUFORIA and its major hacks, tweaks, and workarounds

EUFORIA [247] is another of the three major case studies of this thesis that aims to take advantage of the different e-Science infrastructures EGEE/EGI and DEISA/PRACE with several cross-infrastructure scientific workflows. Detailed information about workarounds, tweaks and limitations can be found in [225] and [270] as well as in Section 6.5 that focusses on this case study and also describes how limitations are addressed by thesis work. The aim of this section is thus just to present a brief overview of the major hacks, tweaks and experiences when working with the EUFORIA use case.

The EUFORIA use case is another example of the 'neutral bridge' approach using the proprietary UNICORE atomic services [293] together with gLite proprietary interfaces, most notably CREAM via the gLite UI [212]. The neutral bridge is implemented in the Resource Allocation Server (RAS) [225] being used with the KEPLER workflow tool [215] through a neutral submission library that in turn uses gLite tools and the VINE toolkit [278]. Transformation logic was necessary to transform the end-user job submission requirements into different job description languages that have different ways of specifying executables. Low-level information about the installed executables had similiar difficulties as described in the VPH and WISDOM case studies. For gLite submissions the RAS service directly used the gLite UI while for UNICORE submissions the VINE toolkit was used.

Other applied workarounds via scripts have been performed to enable the use of different libraries on different resources and to specify Grid job applications in a meaningful way al-

No.	Short description of use case challenges
(a)	Transformation logic was necessary in the neutral bridge to convert between different job description languages, job submission and management interfaces and security credentials using different tools to maintain such as VINE and the gLite UI
(b)	E-mail communication was used to get information about the location of executables and software and specifying Grid applications was a cumbersome task using tweaked scripts
(c)	Error-prone UNIX-scripts have been used to enable jobs to different Grid sites that partly rely also on low-level environment variables for the seamless execution
(d)	Tracking usage across different HPC sites in DEISA/PRACE with additional dedicated HPC machines (e.g. HPC-FF) and including the use of EGEE/EGI was difficult

Table 3.6: Summary of major tweaks, hacks, and workarounds in the EUFORIA use case.

though many of them have been pre-configured on the corresponding sites. But hacks in the job descriptions have been necessary to enable the use of Grid applications in conjunction with low-level environments available at the given Grid sites. The e-Scientists themselves have to perform many manual steps on the low-level of machines that could have been better supported with Grid middleware. One example is the use of different systems within EGEE/EGI and DEISA/PRACE with the same executables (e.g. ILSA [199] or HELENA [198]), but having different execution environments with differet locations of pre-installed software (e.g. libraries, compilers, etc.). Phone and e-mail communication was partly necessary to communicate the location of installed software. Many complex UNIX-scripts have been used to work with the wide variety of fusion applications in conjunction with EGEE/EGI and DEISA/PRACE requiring a more common way of e-Science application support. On top of those issues, it was not clear how resource usage tracking is performed in a cross-infrastructure setup, including also dedicated fusion community HPC machines (e.g. the HPC-FF system in Juelich [42]).

The summary of the major hacks, tweaks, and workarounds of the EUFORIA use case are listed in Table 3.6.

3.3.3 Gateway Approach

The *'gateway'* approach seems to be theoretically an ideal concept, but is practically non-trivial in terms of maintainability and support issues. The 'gateway' stands for one central entity that is able to translate any technology protocol/schema into any other protocol/schema using its transformation logic. This approach is shown in Figure 3.11 in which the central gateway is able to contact, and be contacted by, any different Grid technology (A, B,... n). It is hard to maintain since any change in protocols affect the central gateway.

This approach was used to realise the interoperability between the European infrastructure EGEE with gLite and VEGA [307], which is the Grid Operating System (GOS) for the CNGrid infrastructure in China [3]. Kryza et al. describes in [209] that interoperability is achieved via a central implementation instance named the *Grid Abstraction Layer (GAL)* which can be seen as one instance of a central gateway. The GAL enables interoperability between EGEE and VEGA and enables the integration of any other Grid environments. The extensible design is driven by the requirement to add further technology support in the gateway.

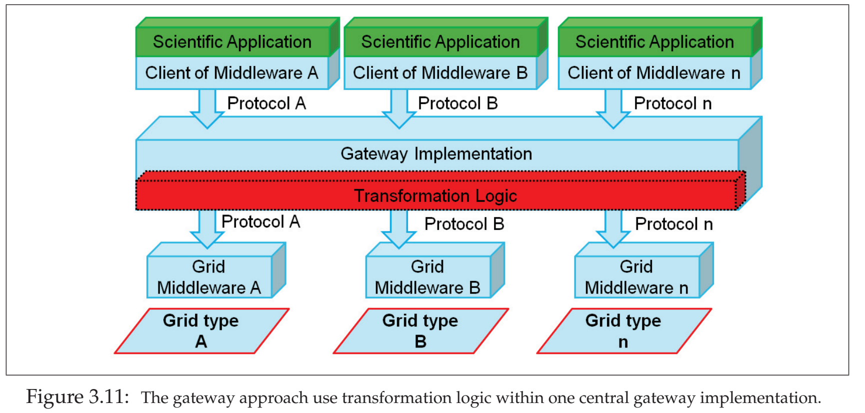


Figure 3.11: The gateway approach use transformation logic within one central gateway implementation.

Regarding the authentication and authorization problem, two different security modules have been defined in the Gateway. This leads to known problems w.r.t. maintenance of these modules, especially when one middleware changes its security layout. Grid job control and management challenges are implied as well as the specific adoption of Berkeley Database Information Index (BDII)-specific clients [212] to get up-to-date information about resources in EGEE in parallel to those from VEGA using other methods.

The approach leads to the adoption of a wide variety of different service interfaces and clients that are available within Grids. At the same time this approach leads to maintenance problems within the central entity, since a protocol and schema change is likely to occur if the central gateway really supports several proprietary middleware interfaces in the future.

3.3.4 Mediator Approach

The mediator approach is similar to the neutral bridge approach, but instead of using a neutral protocol, the respective client technology sticks to one specific protocol named for simplicity as 'protocol A'. This protocol can be used to access all Grid middleware systems that natively supports this protocol A, but it can also be used to access known mediators that also offer this protocol type. These central mediators are always used via one specific protocol, but are in turn able to translate it into any other protocol with their implemented transformation logic. The mediator approach is illustrated in Figure 3.12. Technologies based on protocol A can be normally accessed, but a central mediator can also be used that transforms protocol A into any other protocol/schema with its transformation logic to contact different Grid technologies (A, B, ... n). It is thus a specialization of the aforementioned gateway approach and easier to maintain, but also having limited functionality.

This approach is adopted in the technologies that make EGEE interoperable with Berkely Open Infrastructure for Network Computing (BOINC)-based infrastructures [112], as described by Kaczuk et al. in [204]. The approach is implemented as part of the Enabling Desktop Grids for e-Science (EDGeS) project [18] that establishes interoperability between the EGEE e-Science infrastructure and a wide variety of so-called desktop Grids (i.e. BOINC-based Grids). The BOINC client is modified in such a way that the EGEE Grid appears to be a powerful PC as part of the BOINC-based Grids and thus can be transparently used like home PCs.

The initial implementation overloaded the central gLite component within EGEE named as

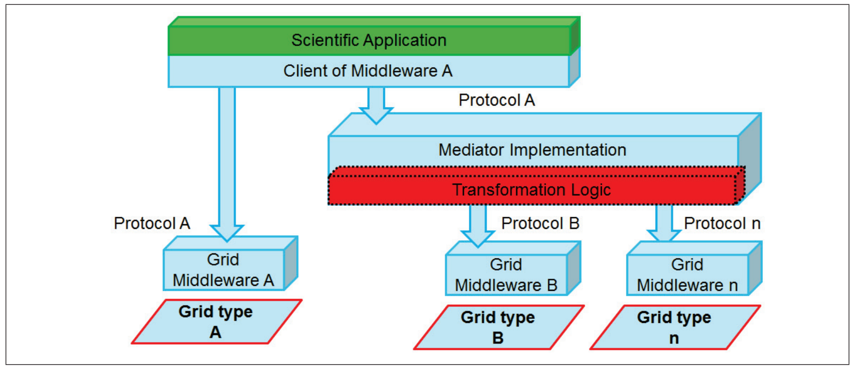


Figure 3.12: The mediator approach uses one protocol as a basis to contact the mediator (if needed).

the Workload Management System (WMS) [212], basically by submitting one job after another. Therefore the mediator implementation was improved to use a special feature from the WMS in terms of receiving a full collection of jobs at once. This in turn is better suited for the BOINC-based design, and thus prevented the design and implementation from regularly occurring Grid overloads.

One of the lessons learned of this approach is the problem that the 'way of using' an architecture is often different too. Would the aforementioned work be guided by a reference model and reference architecture, the way of using the elements would be more clear. This particular component-based approach example is succesful in establishing interoperation with non Grid community-based infrastructures basically gaining access to voluntarily provided resources by end-users.

3.3.5 Adapter Approach

Another, often applied, approach is the adapter concept, because from all of the surveyed approaches it is the easiest one. This means a typical Grid technology client submits with 'protocol A' its job to the respective 'Grid middleware A', which in turn, after processing the job description, forwards it to a dedicated adapter for another middleware. This adapter provides the transformation logic that transforms the job into the protocol or schema format of the corresponding 'Grid middleware B'. Hence, the difference to other approaches such as the mediator is that the Grid job is actually processed in one middleware stack before being forwarded to another 'Grid middleware B' for execution. The well-known adapter approach is illustrated in Figure 3.13.

One example of using the adapter approach to enable interoperability is described by Gronager et al. in [189] and used in the CMS experiment [148] at CERN utilising the EGEE and NDGF infrastructures. Two schemes are presented to enable job submission from gLite to ARC by using the WMS gateway scheme and the Computing Element (CE)-gateway scheme that both implement the adapter approach. In the WMS gateway scheme the gLite WMS can directly submit jobs to ARC-CE, because an adapter based on the Condor-G technology was used to submit to ARC-CEs. In contrast, the CE-gateway scheme is an adapter included as part of the gLite CE that translates gLite jobs directly into xRSL scripts [160] that represents the proprietary job description language of ARC.

Another example is described in [256] that enables the interoperation between EGEE and

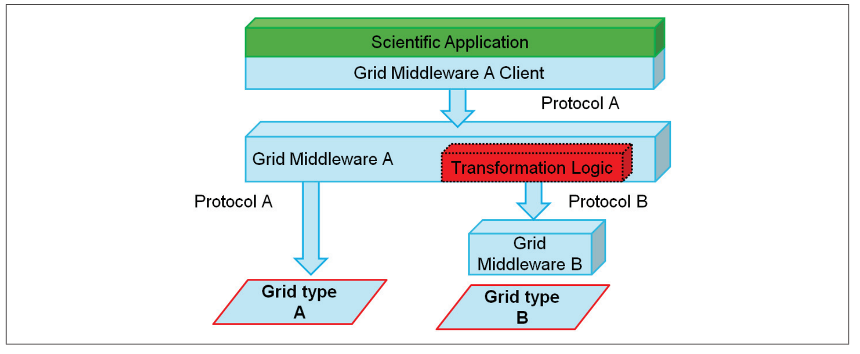


Figure 3.13: The adapter approach uses transformation logic after fully processing the job in one middleware.

DEISA. This approach was implemented by a dedicated gLite Target System Interface (TSI) [294] environment within UNICORE 5. After jobs are processed in the core UNICORE 5 components, such as the Network Job Supervisor (NJS) [294], they are forwarded to gLite CEs for job processing as part of EGEE. Here, the gLite adapter consists of transformation logic to transform job submission commands into the JDL language [212] used by the gLite middleware.

Another example by Wang et al. [304] describes the interoperability between the middleware Grid Resources for Industrial Applications (GRIA) [89] and GOS using an adapter approach. In this example, GRIA acts as a 'front interface' for GOS and forwards received jobs to CNGrid in China for processing.

3.3.6 Middleware Co-existence

Finally, although not exactly a direct solution for the interoperability problem, the middleware co-existence approach is worth being considered alongside the others. It circumvents the interoperability problem by the provisioning of each desired different Grid middleware system with a respective Grid client for each resource. Each of these Grid resources provides access to a particular RMS that is then used in parallel with multiple Grid technologies. This implies a major amount of efforts for deployment as well as maintenance overhead, and as a consequence this approach is rarely used in production. Parallel middleware deployment in this case avoids having any form of transformation logic. This approach was listed for the sake of completeness in terms of approaches in the context of the given interoperability problems. It is illustrated in Figure 3.14 in which no transformation logic is used. But this approach raises serious concerns in terms of deployment and maintenance overheads for infrastructures.

In order to circumvent the interoperability problem between UNICORE, gLite and Globus, that all have been required by different VOs, the German national D-Grid [238] uses the middleware co-existence approach. D-Grid provides deployments of each of the three middleware systems for each of its Grid resources in parallel. The e-Scientists within the different D-Grid VOs have the freedom to choose the middleware they want to use. This is a benefit in this particular approach so that e-Scientists have the freedom to select the technology of choice. But in this particular approach, in contrast to interoperability approaches, this freedom comes with a high price of maintenance and deployment efforts. It is not native interoperability where only one middleware could be used that is standard-based and thus interchangeable with other standard-based middleware systems thus only requiring one deployment in D-Grid. Although

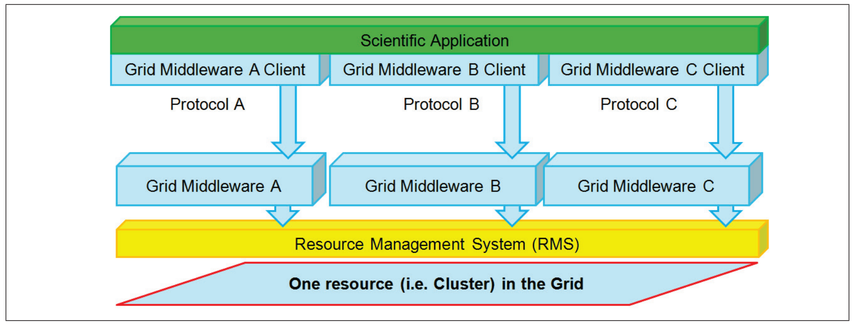


Figure 3.14: The middleware co-existence uses parallel middleware deployments.

the benefit looks the same for end-users, it is essentially a high overhead for administrators and infrastructure providers. More recently, there is a new organization called NGI.DE [60] that represents the German national Grid activities nowadays. Future deployments for the German national Grid will possibly evaluate a change avoiding the middleware co-existence.

Another example is the Grid INFN Laboratory for Dissemination Activities (GILDA) infrastructure [124]. Different technologies co-exist in this infrastructure that share the same computational resources and RMS systems. As described in [124], the infrastructure deploys gLite, Globus and OMII-UK [70] in parallel, while the Torque RMS [98] is used underneath.

3.4 Conclusion

The major research question of this thesis is *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. The identification of known approaches in the context of interoperability leads in this context to two major views. These two major views consider that open standards are important but alone are not sufficient to improve interoperability. Firstly, the survey of known reference models represents a top-down view on the problem space. In contrast, the survey of existing component-based concepts provides a bottom-up view facing low-level interoperability issues. With this views, the conclusion is that the converging point are open standard adoptions, but guided by a reference model and implemented in components avoiding proprietary interfaces and schemas wherever possible.

Related work approaches firstly lead to reference models and what problems exist when non-standard component-based solutions are used in production Grids. General key reference model principles (e.g. being abstract, having entities and relationships, independent of concrete deployments, etc.) have been identified based on the OASIS SOA reference model. This foundational model provides insights how the broad term 'reference model' can guide a complete set of associated elements, such as reference architectures or standards. These elements represent a guiding line throughout this thesis essentially forming a 'frame of reference' for thesis contributions in all subsequent chapters. A more precise classification of the reference model ecosystem has been done to understand interoperability issues across different production e-Science infrastructures. Component-based solutions that do not use standards have many drawbacks (e.g. error-prone, semantic-loss, etc.). Open standards, as a major part of associated reference architecture elements, need to be a necessary cornerstone in the model design presented in this thesis.

Academic analysis of existing work of long existing reference models around TCP/IP and the ISO/OSI model leads to conclusions that a compact reference model has a better chance of being useful in production e-Science infrastructures. One of the major approaches derived from this conclusion is that in this thesis, a compact approach needs to be followed. This is required even if it is at the boundary of being in-line with traditional software engineering w.r.t. reference models (like with the TCP/IP case). Critically analysing the non compact OGSA reference model and architecture provided further evidence that the impact of non compact models for production infrastructures is minimal despite being known for about ten years.

Given its initial definition, OGSA is still considered as the foundational reference model and architecture. Some elements of it are derived taking into account experience from production Grids in context. 27 relevant indicators have been defined that form the basis for seven distinct reference model factors that are the key metrics whether a reference model is able to provide solutions to the research question of this thesis. Taking these indicators and factors as a basis, known reference models are surveyed in the field and the conclusion is that none of them actually fulfil all the important factors and indicators. There is no reference model able to provide an answer to the given research problem today. Further related work is taken into account by surveying component-based approaches leading to valuable lessons learned from existing interoperation approaches. The main conclusion of this chapter is the evidence that the lack of interoperability in e-Science infrastructures can be at least partly explained by the lack of a standards-based reference model and associated architecture work within production Grids today.

Chapter 4

Requirements

A model of the given problem space is presented in Chapter 2 and similiar problems and known related approaches have been reviewed in Chapter 3. One major conclusion of this analysis is that the challenges in e-Science infrastructure interoperability can at least be partly explained by the lack of a production-oriented and standard-based reference model. The academic analysis of existing experience and related work culminate in a set of requirements for a standard-based reference model which is presented in this chapter. The requirements lay the foundation for having solutions that have the potential to overcome many limitations of the current approaches described in Chapters 2 and 3.

The requirements precisely define key elements of a solution that is aimed at providing an answer to the major research question. This Chapter thus partly answers the question of *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. The precise set of requirements enables in subsequent chapters the creation of a well-specified standard-based reference model, including its related elements. These reference model requirements are defined on different levels from the abstract to the more concrete. Requirements are defined on the abstract reference model level, concrete reference architecture level and finishing with those for derived concrete architectures. Many of these requirements are published in [267] and [264].

Defining a reference model is the first, most important step towards interoperability of e-Science infrastructures in this thesis. But in order to sustain the established interoperability of such a model, it is important to define a process alongside the work of the reference model architecture. Otherwise, the reference model and its associated architecture will only be usable for limited time, because e-Science infrastructure setups are known to change over time, although in a very slow manner. This is one of the key challenges achieving the right *'Balance: Maintaining equilibrium between stability of services and innovation in the e-Infrastructure itself, especially as technology and user requirements evolve'* [210]. A set of complementary requirements are defined for such an associated process. They are mainly based on results of the analysis of existing solutions in general, and the ecosystem of the e-Science infrastructures in particular. In addition to the architecture work requirements, the survey of related approaches revealed detailed functional elements that need to be supported in production e-Science infrastructures.

The first section of this chapter defines the requirements for a standard-based reference model using different levels of abstractions in order to address the findings of the previous chapter. The second section focuses on the more concrete functional requirements, leading to detailed requirements where the current surveyed solutions can be considerably improved. The final section provides a complementary set of requirements that address different segments that are necessary to enable and sustain e-Science infrastructure interoperability.

4.1 Reference Model and Associated Elements Requirements

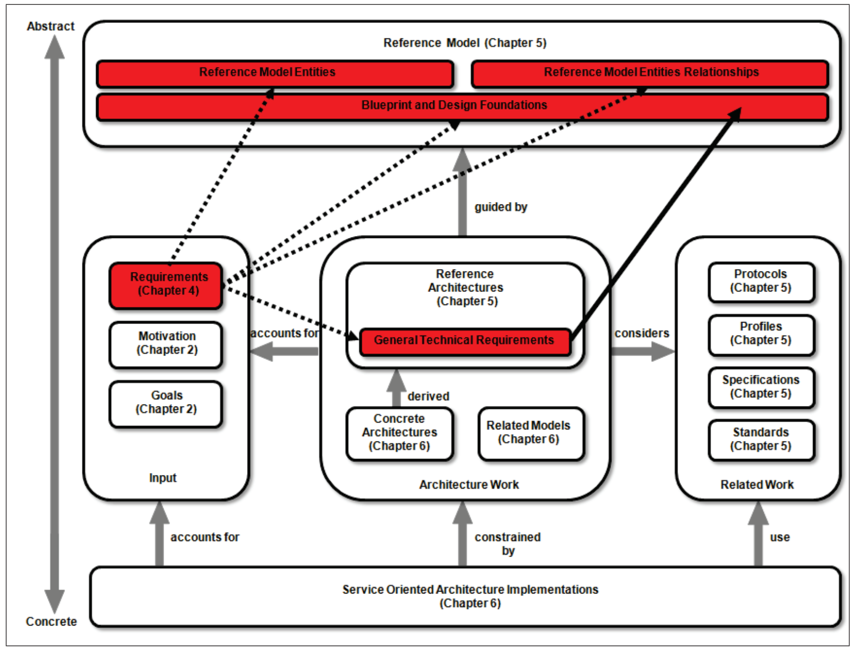


Figure 4.1: Requirements influence the reference model and general technical reference architecture design.

This section defines concrete requirements of the broader reference model and its associated elements using different levels of abstractions. They are defined on different levels according to the blueprint of the OASIS SOA reference model [217] and as illustrated in Figure 4.1. It shows how the requirements defined in this section influence the blueprint design and its foundations as well as entities and their relationships on the reference model level. The figure also provides a hint that the foundations guide the more technical design of the *'general technical requirements'* on the reference architecture level.

4.1.1 Reference Model Blueprint and Entity Requirements

This section defines the requirements for the reference model blueprint that guides the design of its associated architecture. The requirements are thus for the level of the reference model itself as illustrated in Figure 4.1. The requirements defined on the reference model level in this section are summarized as part of Table 4.1.

The foundational requirement is that the reference model in this thesis follows key design principles as mentioned in the OASIS SOA RM [217] . This requirement is defined as follows:

Definition 28 (General Reference Model Design Principles) *The General reference model design needs to follow the reference model key principles as defined in Definition 24.*

Definition 28, requires to define a reference model in an abstract manner without mentioning concrete deployments in e-Science infrastructures such as DEISA/PRACE or EGEE/EGI. It mandates the concrete definition of entities including their relationships in order to understand the important *'links'* between them. It further points to defining one particular problem domain, thus adding focus to proposed solutions avoiding having to tackle all Grid problems that exist today. Definition 28 also mandates to make no assumptions on particular middleware technologies such as UNICORE, gLite, ARC, Globus, or GENESIS being thus technology agnostic.

Several relevant factors with indicators for reference models (see Table 3.1) are published in [264] and have been described in Chapter 3. These are requirements that need to be concretely defined in this section. General service-based reference model requirements are defined as follows:

Definition 29 (Service-based Reference Model) *A service-based reference model is based on a SOA-conform design in the sense of Definition 23 and enables a distributed system consisting of entities that offer service interfaces while their services have clear, unique semantics.*

Definition 29 requires that a reference model and its associated elements need to follow the design principles of SOAs. This is in-line as reported by the e-IRG in the Blue Paper 2010: *'In all e-Infrastructure domains there is a move towards service-orientation and away from the traditional technology- or product-orientation'* [210]. Entities must offer service interfaces in order to invoke remote operations or to request pieces of information through this interface. Each service semantic is unique being able to distinguish the functionality of a given service and thus avoid overlaps of similar functionality as part of different services or simply unnecessary service duplication.

Another definition is related to its major use cases that drives its design process. E-Science is the focus of this thesis and thus the scope of the particular reference model is as follows:

Definition 30 (e-Science-Driven Reference Model) *An e-Science-driven reference model is focused on e-Science applications in the sense of Definition 9 and not on e-business or commercial use cases.*

Definition 30 limits the scope on the reference model design to environments that are driven by scientific use cases. It can be used by commercial vendors or be beneficial for use in business, but its requirements are focused on the needs of e-Scientists as defined in Definition 10.

The aforementioned requirements influence the overall design of a reference model while in the next paragraphs entities and relationships are defined that are required for a particular solution in the given problem domain. One major entity of the reference model in our given environment is published in [267] and defined as follows:

Definition 31 (Grid Execution Management Entity) *A Grid execution management entity offers specific functionality for the execution and management of e-Science applications according to Definition 9 on computational infrastructure resources as defined in Definition 6 using a Grid job description language and a well-defined Grid job execution environment.*

Definition 31 raises the demand for an entity that is able to submit and manage computational activities on resources provided by an e-Science infrastructure. Examples are the UNICORE Atomic Services (UAS) [293] or the job submission and control interfaces in ARC known as A-REX [160], or CREAM [212] of gLite. All these take some form of a job description language as input in order to accurately define the computational activities that are intended to run on computational resources. In many cases, the underlying computational resources are managed by RMS systems (cf. Definition 11) and thus the entity defined in Definition 31 also raises the demand for a powerful execution backend that is able to work together with RMS systems.

The second key entity requirement for the reference model covers the area of data management, which is different from the computational area in many respects as described in [267]. The fact that once stored data must be migrated to other storage systems in a very time-intensive manner is a huge difference to computational entities where jobs can be just re-submitted again to another resource if necessary. Data management entities and their interfaces thus must be carefully chosen in e-Science infrastructures. The entity requirement is as follows:

Definition 32 (Grid Data Management Entity) *A Grid data management entity offers specific functionality for storing, retrieving and managing data with storage-related infrastructure resources as defined in Definition 6 including functionality for large-scale data transfers and POSIX-based access.*

The entities that are required by Definition 32 are used for various purposes within production e-Science infrastructures (cf. Definition 5). They are used to store measurement data from large experiments (e.g. like LHC experiments [104]) used subsequently for computational activities using entities such as those defined by Definition 31. The data management entity could even act as a shared central VO storage according to Definition 8 where multiple users can obtain this measurement data in order to analyze it with computational methods. The analysis results of such computational activities are also stored in data management entities defined by Definition 32. The data management entity offers data transfer functionality in order to be used with geographically dispersed compute or storage resources. Examples of such an entity are dCache [178], Disk Pool Manager (DPM) [11], StoRM [143], or iRods [229].

Another entity requirement in the field of security is identified in [267] being *'orthogonal'* to the previous defined entities. Authentication and authorization of end-users are important during any of the activities described with the aforementioned entities. The security entity requirement is as follows:

Definition 33 (Grid Security Entity) *The Grid security entity is a trusted service that releases security attributes of the end-users bound to the identity of an end-user. These attributes are then used for attribute-based authorisation and authentication within other reference model service entities in order to realise access control as required by setups according to Definition 8.*

Security attributes of end-users within a Grid infrastructure (cf. Definition 7) vary and can be a role possession (e.g. VO admin) or a dedicated group/VO membership (e.g. member of VO Atlas). Entities defined in Definition 33 provide interfaces (and mechanisms) to obtain attribute statements that are signed with the service identity of the entity. This entity could be realized as a service one can trust by using an X.509 certificate [195] that in turn can be used during the signing process of the attributes for end-users. Examples are VO Management Service (VOMS) [108], UNICORE VO Service (UVOS) [293], or Shibboleth [232].

The fourth entity raises the demand for functionality that provides the current status within Grids, including the capabilities of each available service as described in [267]. Status information about which services (e.g. type) and resources (e.g. resource load) are available is crucial. An information-related entity requirement is defined as follows:

Definition 34 (Grid Information Entity) *A Grid information entity publishes the status of Grid services as in Definition 13 and its underlying infrastructure resources as in Definition 6 with an information model that is able to express their capabilities. It thus provides static information (e.g. resource location) and dynamic information (e.g. resource load), including the tracking of resource usage.*

No.	Requirement Definition Title
28	General Reference Model Design Principles
29	Service-based Reference Model
30	e-Science-Driven Reference Model
31	Grid Execution Management Entity
32	Grid Data Management Entity
33	Grid Security Entity
34	Grid Information Entity

Table 4.1: Reference model blueprint and entity requirements (reference model level).

Definition 34 mandates an entity that knows about all other entities as previously defined by Definitions 31, 32, and 33. An information entity must provide an interface that can be queried in order to determine what the (current) computational or data resource status is at a given site. It must offer information about services including their security setups. Such queries can be performed by end-users or by brokers (e.g. WMS [212]) that take advantage of such pieces of information. Examples are BDII [212], ARC Information Service (ARIS) [160], or the Common Information Service (CIS) [223].

Finally, Table 4.1 summarizes the aforementioned requirements that are part of the reference model level.

4.1.2 Reference Model Entity Relationships

Fundamental entities that are required for a SOA-based reference model in e-Science infrastructures have been defined in the last section. This section defines relationships of these entities since there are important interactions between these entities that are partly *'orthogonal'* to each other [267]. The relationships are also defined on the reference model level as shown in Figure 4.1. The relationship requirements defined on the reference model level in this section are summarized as part of Table 4.2.

The Information entity (cf. Definition 34) and the Execution management entity (cf. Definition 31) relate to each other as follows:

Definition 35 (Information and Execution Management Entity Relationship) *The information and execution management entity relationship is the exposure of the static and dynamic information about the corresponding available execution management entity, including its underlying computational resources and capabilities.*

Definition 35 describes one of the most fundamentally important relationships about the reference model entities. It can be n:m, meaning that several information entities can describe several execution management entities. The information model used by the information entity must provide a rich semantic way of describing execution management entities. To provide an example of technical realisation, Definition 35 raises the demand for an information service that is able to expose the location of a computational service, including its major underlying resource characteristics (e.g. amount of cores, memory, etc.).

The relationship between the Information entity (cf. Definition 34) and the Data management entity (cf. Definition 32) is defined as follows:

Definition 36 (Information and Data Management Entity Relationship) *The information and data management entity relationship is the exposure of static and dynamic information about the corresponding available data management entity, including its underlying resources and capabilities.*

In contrast to Definition 35, Definition 36 requires an information entity that exposes information about the storage managed by data management entities. It can be n:m meaning that several information entities can describe several data management entities. A semantic rich information model for the information entity is required that is capable of describing data management entities. To provide an example of technical realisation, Definition 36 raises the demand for an information service that is able to expose the location of a data management service, including its major underlying storage resource characteristics (e.g. amount of free data storage space, etc.).

Another relationship is between the Security entity (cf. Definition 33) and the Computational entity (cf. Definition 31). It can be seen as being *'orthogonal'* [267] to the previous ones enabling a secure setup of production e-Science infrastructures that offer execution management entities. The relationship is defined as follows:

Definition 37 (Security and Execution Management Entity Relationship) *The security and execution managment entity relationship is the support of authentication and attribute-based authorisation in the computational entity, which works with security attributes and credentials released by a security entity.*

Definition 37 requires that the attributes that are released from the security entity are correctly interpreted by the execution management entity that receives the credentials and attributes with end-user information. It is n:m meaning that there can be a wide variety of security entities that are supported by several execution management entities. The encoding of the attributes must follow a common syntax and semantics in order to ensure interoperability between different incarnations of the reference model entities. The common format refers to the encoding type of attributes as well as the transportation format. One practical example of Definition 37 is that an end-user receives a credential from a security entity defining the attribute in a manner that the end-user is part of the Atlas VO. The execution management entity needs to analyse the presented credential (with attributes) from this end-user and grant access

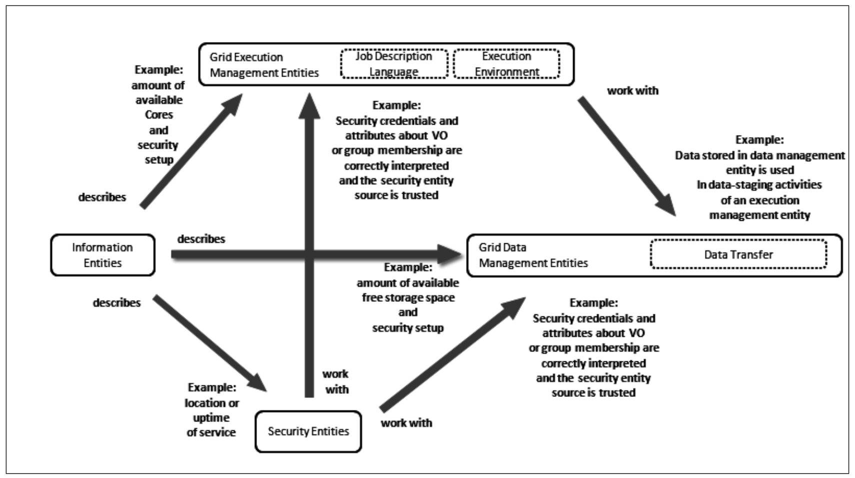


Figure 4.2: Relationship requirements between the different entities of the reference model.

or not depending on the security policy setup indicating whether or not Atlas VO members are allowed to use this entity.

The relationship between the security entity (cf. Definition 33) and the data management entity (cf. Definition 32) are defined as follows:

Definition 38 (Security and Data Management Entity Relationship) *The security and data management entity relationship is the support of authentication and attribute-based authorisation in the data management entity, which works with security attributes and credentials released by a security entity.*

Definition 38 requires that attributes that are released from the security entity are correctly interpreted by the data management entity. It is n:m meaning that there can be a wide variety of security entities that are supported by several data management entities. The encoding of the attributes must follow one particular style to ensure the interoperability between different data management entities used by end-users. The encoding style of attributes as well as the transportation format largely depend on the security entity used in production e-Science infrastructures today. One practical example of Definition 38 is that an end-user receives a credential from a security entity defining the attribute in a manner that the end-user is part of the CMS VO. The data management entity needs to analyse the presented credential (with attributes) from this end-user and grant access or not depending on the security policy setup indicating whether or not CMS VO members are allowed to access the storage. In the same way, each end-user request for file transfers managed by the data management entity is required to support the relationship as defined in Definition 38.

While the two latter requirements require that the security entity works together with the computational and data management entities, the important relationship between the information entity (cf. Definition 34) and the security entity (cf. Definition 33) must be also defined as follows:

Definition 39 (Information and Security Entity Relationship) *The information and security relationship entity relationship is the exposure of information about the corresponding available security service and its capabilities. This relationship also includes that security information is available for computational and data management entities as well.*

The scope of Definition 39 is twofold, but as they are tightly related they are defined in one requirement. First, it raises the requirement that the information entity is capable of exposing information about a security entity. This relationship is n:m referring to the fact that several information entities can describe several security entities. In the second case, Definition 39 mandates that the security setup information is exposed, not only in each of the security entities, but also in the related execution management and data management entities.

The latter indicates which security model one needs to support in an entity of the reference model in order to contact an entity. Interoperability setups between e-Science infrastructures require information beyond resource-orientated information meaning information about the security setups of the execution management and data management entities.

To provide an example, the first element of Definition 39 can be the exposure of the location or uptime of a particular security entity. One example for the second part of Definition 39 can be one particular data management entity that is only accessible via full X.509 certificates without any X.509 proxy support. This information enables clients to pick the right security setup where possible or to avoid any contact with the service when the particular security setup of the described execution management and data management entity is not supported.

The last requirement is the relationship between the execution management entity (cf. Definition 31) and the data management entity (cf. Definition 32). It is defined as follows:

No.	Requirement Definition Title
35	Information and Execution Management Entity Relationship
36	Information and Data Management Entity Relationship
37	Security and Execution Management Entity Relationship
38	Security and Data Management Entity Relationship
39	Information and Security Entity Relationship
40	Execution Management and Data Management Entity Relationship

Table 4.2: Reference model entity relationships requirements (reference model level).

Definition 40 (Execution Management and Data Management Entity Relationship) *The execution management and data management entity relationship is the functionality to perform file transfers (i.e. data-staging activities) as part of the computational services using different data management entities.*

Definition 40 points to the computational activities performed by the execution management entity as defined in Definition 31. The data-staging mechanism [115], must be provided by the execution management entity in order to perform data transfers with the help of data management entities. More specifically, this mechanism transfers data to and from the working directory of the corresponding compute resource. The execution management entity needs to work with the data management entity in such a manner that data from the latter entities can be easily used by the execution management entities. It is n:m since several execution management entities can take advantage of multiple data management entities in order to receive input data for computation or to store results after computational runs. Examples are end-users that want to analyse data of measurements from one experiment at the LHC where this data is stored in one particular data management entity. Definition 40 raises the requirement that the execution management entity used for the computational analysis is able to retrieve the measurement data from the corresponding data management entity before the analysis steps actually begin.

Figure 4.2 provides a short overview of the entities and their complex relationships that are an important factor to ensure that the important interactions in derived concrete architectures are actually working with one and another. The interactions between these entities are non-trivial. Especially when security and information aspects are included it becomes complex while at the same time this is important in order to ensure production e-Science infrastructure requirements. *'Orthogonal'* [267] addresses the lessons learned that many approaches in Chapter 3 left out such as important security and information details for simplicity. The reaons have been mostly that security (and semantic identical information in-line with other entities) is out of scope and can be added later. But security and information must be considered in the initial reference model and architectural design as any other entity. Finally, Table 4.2 summarizes the aforementioned requirements that are part of the reference model level.

4.1.3 General Technical Reference Architecture Requirements

The previous sections defined requirements for a general reference model blueprint that guides its associated architecture elements. Requirements for abstract entities and their relationships have been defined. While they provide a good frame of reference, a definition of more concrete architecture elements is needed. This section will provide the requirements for an associated reference architecture that is one level more concrete as illustrated in Figure 4.1. The general requirements defined on the reference architecture level in this section are summarized as part of Table 4.3.

The survey of related work in Chapter 3 identified that the minority of reference models provide an associated reference architecture that gives a reasonable level of detail for its implementation. The following requirement is thus defined that significantly influences the major design of a reference architecture for the given problem space:

Definition 41 (Web Services-based Reference Architecture) *A Web service-based reference architecture implements the general SOA design principles of a reference model as defined in Definition 29 with the use of Web service message exchanges that leverage the HTTP(S) and SOAP protocols.*

Definition 41 raises the demand that the general reference model implements SOA principles with the Web services technology [296] given its e-Science community relevance. The associated reference architecture should be implemented with HTTP(S) [43] and SOAP-based request and response messages [190] leveraging the potential of a wide variety of WS-based specifications available today (e.g. WS-Security [131]). One practical example of Definition 41 is the access of a service-based data management entity as defined in Definition 32 realised with WS message exchanges. The requirement raises the demand that services of the data management entities in the reference architectures, offer WS interfaces that are described with WSDL [150] and accessed using the SOAP protocol [190] via HTTP(S).

The next requirement defines which elements of the Web service-based interfaces are needed:

Definition 42 (Concrete Specifications for a Reference Architecture) *Concrete specifications for a reference architecture are present in two types. Firstly those that offer well-formed XML-based schemas for keeping information consistently. Secondly, XML-based schemas that include definitions of operations that can be remotely invoked via WS message exchanges as in Definition 41.*

Definition 42 defines which Web service interface elements (i.e. portTypes [150] with operations) must be in place in the reference architecture. This is needed for all the four major reference model entities as defined in Definition 31, 32, 33, and 34. The operations are basically functions (grouped in portTypes) that can be invoked remotely by implementing the idea of an XML-based RPC [296]. One practical example is the operation `GetResourceProperties` of the Web Service Resource Properties specification [188] being part of the larger WS-RF framework [123]. Remote clients can invoke this operation via Web services that adopt this specification without knowing the server implementation, since the aforementioned specification is accurately defined and thus realistically implementable. Definition 42 also includes well-formed schemas that basically specify XML-based data structures. As published in [264], the survey of related work reveals in Chapter 3 that not many architectures refer to real existing WS-based specifications.

The two latter requirements define invocations of remote operations in distributed systems such as production e-Science infrastructures as defined in Definition 5. The next requirement demands the restriction of certain amounts of transferred information during these invocations. The academic analysis in this thesis revealed that information and security information are special and as such all data related to it need precise constraints and invariants in order to understand where in the reference architecture the information or security information is transferred (i.e. because of the *'orthogonal'* approach in [267]). These contraint requirements are defined as follows:

Definition 43 (Information and Security Constraints for a Reference Architecture) *Information and security constraints for a reference architecture are invariants that clearly define which information and security data exist at well-defined locations within the architecture.*

Definition 43 is required for the reference architecture since it not only uses XML-based operations, but also relatively large XML-based schemas that define data structures. This is the case in the context of the defined information entity (cf. Definition 34) and its information model that can be realised with XML-based schema (e.g. GLUE2 [113]). These schemas are also relevant in context of the defined security entities (cf. Definition 33), for instance when realizing parts of it with SAML assertions that consist of XML-based security attribute statements [142]. Also the execution management entity (cf. Definition 31) can take advantage of XML-based schemas, for instance, when realizing job description languages (e.g. JSDL [115]). Information and security data is thus relevant on many levels within the architecture and requires precise invariants and constraints during the *'data flow'*. An XML-based job description data, for example, is just an input to the corresponding job management interface and thus its flow through the architecture is already well defined. But to provide a practical example for Definition 33 in the context of security, it is essential to define an invariant in the architecture that at the level of authorisation decisions, the encoding for attributes in different reference architecture implementations must be the same. With this definition two e-Science infrastructures are interoperable in the sense of Definition 21.

The aforementioned requirements already define several important cornerstones of the reference architecture that lay a foundation for its implementation. More lessons learned from Chapter 3 are taken into account in the subsequent requirements of the reference architectural design to make it more realistically implementable. As published in [264], the success principle of the TCP/IP model compared to the more theoretical ISO/OSI model is a major design aspect in this thesis. The reference model and its architecture must follow a more compact design that is defined as follows:

Definition 44 (Slim Reference Architecture) *A slim reference architecture is provided by a compact architectural design that only implements the four core reference model entities (i.e. Definitions 31,32, 33, and 34) with the given reference architecture technologies.*

Definition 44 is based the assumption that the compact reference models and slim architectures, such as TCP/IP, are more successful in practice in comparison with the rather theoretical ISO/OSI model and architecture. The precise number of lower than five is a requirement that is derived from the findings of the OGSA analysis in Chapter 3. It sets a valid boundary to the key four previously defined entities of the abstract reference model blueprint in Definitions 31, 32, 33, and 34. Definition 44 aims to support the realistic use of a reference architecture implementation within production e-Science infrastructures (cf. Definition 5) covering their most crucial functionality. Other entities and functionalities (e.g. monitoring) are still very important, but are not as crucial for technical interoperability as the aforementioned core four.

In this context, another interesting result of the academic analysis of related work is the absence of definitions which core entities are necessary to form an infrastructure. A list of core entities must be in place in order to create a useful e-Science infrastructure based on reference architecture implementations. This requirement is defined as follows:

Definition 45 (Core Reference Architecture Elements) *The core reference architecture elements are those that are required to form a working e-Science infrastructure and that are derived from the four general reference model blueprint entities covering the mandatory functionality.*

Definition 45 raises the requirement that the architecture elements must be a non-empty list of core elements that must be in place to form an e-Science infrastructure. A reference architecture implementation that is guided by the general reference model blueprint but that do not adopt the core reference architecture elements is not able to satisfy the basic needs of e-Scientists (cf. Definition 10).

Another requirement of the reference architecture that increases its chance of being implemented is referencing well-specified normative specifications:

Definition 46 (Normative Specifications for a Reference Architecture) *Normative specifications for a reference architecture are those well-defined and available specifications that fully specify each operation and related data structures thus paving the way for its implementation.*

Definition 46 requires that the reference architecture needs to explicitly list existing normative specifications that are formal enough to be implemented by stakeholders and available to the wider community. In the context of Definition 46 and in order to be realistically implementable, it is not important whether the specification is an open standard or not. Much more important is that a detailed specification is publicly available so that the reference architecture elements guided by the overall reference model design can be implemented.

Lessons learned from Chapter 3 also revealed that open standards as defined in Definition 14 also play a crucial role in the reference architecture requirements. A reference model and its associated elements such as a reference architecture is only interoperable with another if the use of transformation logic (cf. Definition 27) is avoided thus leading to the following requirement:

Definition 47 (Open Standards-based Reference Architecture) *A reference architecture is based on common open standards in the sense of Definition 14 when it lists normative standard specifications as implementation of the reference model blueprint entities. Such well-defined specifications are released from real SDOs, are publicly available, and avoid breaking existing standard specifications.*

Definition 47 is one key requirement, since the survey of releated work in Chapter 3 revealed that a reference architecture should be standard-based, including referencing concrete open standard specifications. The solution in this thesis must be thus standard-based in the complete sense of Definition 47. Entities defined in Definition 31, 32, 33, and 34, require an open standard-based implementation on the reference architecture level in order to overcome limitations of non-interoperable infrastructures (cf. Definition 19). The next definition addresses several necessary improvements for standards over time that are defined as follows:

Definition 48 (Reference Architecture Standard Refinements) *Reference architecture standard refinements are improvements of open standards in the sense of Definition 47 most significantly preserving backwards compatability with existing open standards. Such refinements must be fed back to the process within the SDO originally releasing the standard leading to evolutions of it.*

Definition 48 raises the demand for open standards that are improved with functionality that do not break backwards compatibility with the existing standard. The core approach and functionality of the open standard need to remain valid, while certain additions to it that do not break the standard can be useful in increasing performance and efficiency as well as the effectiveness of some operations.

Finally, Table 4.3 summarizes the aforementioned general requirements that are part of the reference architecture level.

No.	Requirement Definition Title
41	Web Services-based Reference Architecture
42	Concrete Specifications for a Reference Architecture
43	Information and Security Constraints for a Reference Architecture
44	Slim Reference Architecture
45	Core Reference Architecture Elements
46	Normative Specifications for a Reference Architecture
47	Open Standards-based Reference Architecture
48	Reference Architecture Standard Refinements

Table 4.3: Reference architecture general requirements (reference architecture level).

4.2 Functional Requirements

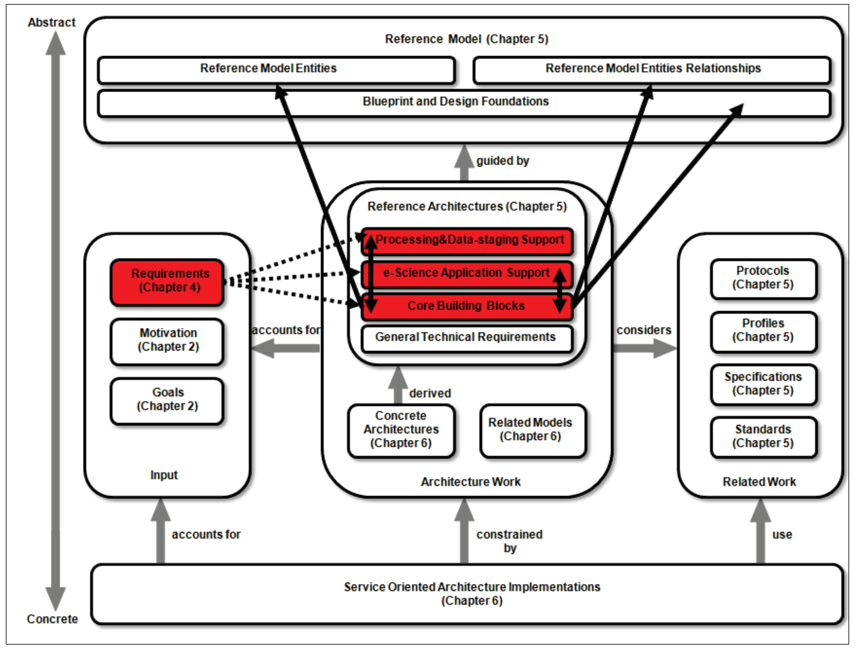


Figure 4.3: Functional requirements of the reference architecture design guided by the reference model.

This section defines detailed functional requirements for the reference architecture in guidance of the overall core reference model as shown in Figure 4.3. The figure illustrates that the requirements are defined on the architectural level. The goal of these requirements is to *'capture the basic functionality'* with core building blocks and also the various lessons learned obtained from field studies and the academic analysis results of Chapter 3. The main aspects is thus to define the very basic functionality requirements related to the thesis scope meaning secure computational job management with support of data-staging and improved processing for e-Science applications (cf. Section 1.2). Figure 4.3 illustrates that core building block requirements of the reference architecture are refined with respect to their detailed functionality in the areas of *'e-Science application support'* as well as *'processing and data-staging support'*. It also provides a hint that the building blocks need to be in-line with the more abstractly defined reference model entities as well as the general high-level reference architecture design requirements (e.g. slim design, entities with relationships, etc.).

4.2.1 Reference Architecture Core Building Blocks

The refinement of the abstract reference model entities are necessary to set more specific requirements for a reference architecture and to explore where technologies need to adopt its necessary building blocks. This section thus consists of more specific *'core building blocks'* and their inter-dependencies as part of the reference architecture as illustrated in Figure 4.3. The core building blocks requirements defined on the reference architecture level in this section are summarized as part of Table 4.4.

All requirements that are listed address the overall reference model requirement of Definition 45, which means that all of them must be satisfied in order to achieve the *'mandatory functionality'*.

A fine-grained definition of the abstract reference model entity Grid Execution Management as defined in Definition 31 is needed. This entity requirement refers to functionality that needs to be provided by middleware technologies in the sense of Definition 12. This requirement can be thus further refined as follows:

Definition 49 (Grid Execution Management Service) *A Grid execution management service of the reference architecture provides the functionality to submit and manage Grid jobs via Grid middleware as defined in Definition 12, which in turn execute and control e-Science applications as defined in Definition 9 on computational infrastructure resources as defined in Definition 6 by using a well-defined job description language and execution environment.*

Definition 49 requires a specification that defines numerous portType operations [150] that can be used to define a WS-based interface within hosting environments of Grid middleware systems. Examples are OGSA-BES [169], or proprietary WS-based job management interfaces in ARC [160], UNICORE [293], or gLite [212].

Definition 49 also points to a job description language. This can be realized via an XML-based schema that is used to define the properties of a Grid job that is controlled by a service that provides access to computational resources. Examples are JSDL [115] or the proprietary job description languages of ARC [160] or gLite [212].

Apart from enabling the running of Grid jobs on HTC resources, Definition 49 also raises the requirement for an execution environment. In more detail such an environment should be able to submit and manage parallel Grid jobs on HPC resources working nicely together with site-specific installed software. Such a framework is beneficial to use particular HPC resource types in the most efficient manner, including specific machine properties and environments. Examples are the GIN Execution Environment [38] and the open source module concept [20].

The Grid Data Management Entity as defined in Definition 32 is another requirement to be refined. It consists of two major parts that can be more precisely defined as follows:

Definition 50 (Grid Data Management Service) *A Grid data management service of the reference architecture provides the functionality to manage data including the access to remote storage resources in the sense of Definition 6 that are capable of efficiently storing and retrieving data using inherently data transfer protocols some of which can be used for data-staging activities initiated by the Grid Execution Management Service as defined in Definition 49.*

Definition 50 raises the requirement to define specific WS-based interfaces that offer portType operations with the functionality to manage, store or retrieve remotely stored data. This data can be stored in different types meaning plain data storage under specific *named entries* or data storage represented by a relational database enabling Structured Query Language (SQL)-based queries. Examples are SRM [286], WS-DAIS [118] or proprietary storage management protocols like the Storage Management Service (SMS) of UNICORE [293]. But these specifications are known to have some disambiguities and design issues hindering scalability that will be revealed and tackled in Chapter 5.

The second part of Definition 50 refers to different types of data transfers. It clarifies that data-staging and storage technologies, which offer a WS-based management interface, need to work together with data transfer protocols. In order to be efficient, the reference architecture

No.	Requirement Definition Title
49	Grid Execution Management Service
50	Grid Data Management Service
51	Grid Authentication Service Functionality
52	Grid Attribute Authority Service
53	Grid Attribute-based Authorisation Functionality
54	Grid Security Attributes Transport
55	Grid Information Model Schema and Service
56	Grid Usage Record Format Schema

Table 4.4: Reference architecture core building blocks requirements (reference architecture level).

needs to support protocols for different types of data transfer. Large-scale data transfers require high throughput, but a standard-based access using POSIX mechanisms makes also sense for smaller data transfers. Examples of large-scale data transfers are GridFTP [109] or HTTPS [43] while POSIX-based access can be realised using the ByteIO specification [230].

The complex Grid security entity as defined in Definition 33 needs also a refinement. The first part of this requirements is related to the use of authentication that we define as follows:

Definition 51 (Grid Authentication Service Functionality) *Grid authentication service functionality are mechanisms that enable each reference architecture service to check whether an end-user access is used in conjunction with a security credential that is valid, not revoked and trusted.*

Definition 51 requires security credentials for authentication. Examples for standards in this area include the use of X.509-based Secure Socket Layer (SSL) connections [195] with a mutual handshake for each of the available services. Technology examples in e-Science environments for authentication of end-users are Shibboleth [232] or the UNICORE Gateway [293] or simply authentication modules in modern Web hosting frameworks wherein Web services are hosted (e.g. Apache Tomcat [149]).

Examples that need to offer authentication functionality are those services defined in Definition 49 and Definition 50. The X.509 certificate used must be released from a trusted CA and not be revoked.

The second part is a service that releases security attributes about end-users defined as follows:

Definition 52 (Grid Attribute Authority Service) *A Grid attribute authority service releases, in a standard format, signed security attributes about users that are used for attribute-based authorisation decisions in reference architecture services such as defined in Definition 49 and Definition 50.*

Definition 52 implies that services provide portType operations that enable the request and release of signed security attributes. The attributes then can be used in conjunction with other service requests (e.g. Grid job submits using the service defined in Definition 49). Definition 52 refers to a standard-based usage of security attributes from a trusted central attribute authority. An example of a standard-based encoding of security attributes is SAML assertions [142], and examples of attribute authorities are VOMS [108] and Shibboleth [232].

The third part is the counterpart to Definition 52, meaning that there is one specific functionality that works with the aforementioned security attributes as follows:

Definition 53 (Grid Attribute-based Authorisation Functionality) *A Grid attribute-based authorisation functionality is able to interpret and use security attributes released from an attribute authority defined in Definition 52. The information extracted from those attributes must be used by the Grid attribute-based authorisation functionality in order to enforce authorisation decisions within services defined in Definitions 49 and 50.*

Definition 53 requires that attribute-based authorisation decisions must be based on functionality that is able to work with security attributes released from an attribute authority (cf. Definition 52). This functionality must be internally supplied to provide local access control, while the use of remote authorisation services with standard protocols is also additionally possible. Examples of such a protocol is XACML [234], and Argus [303] is an example of a central authorisation service.

A final part of Definition 33 is refined as follows:

Definition 54 (Grid Security Attributes Transport) *A Grid security attributes transport is a well specified standardised way of exchanging security attributes between the different services of the reference architecture such as attribute authority (cf. Definition 52), Grid execution management service (cf. Definition 49), and Grid data management service (cf. Definition 50).*

Definition 54 requires that the exchange of security attributes must be performed in a standardized way between the services of the reference architecture. Examples of mechanisms that enable the standardized transport of security credentials in general is the WS-Security set of functionality [131].

The Grid Information Entity as defined in Definition 34 needs to be refined with two definitions, the first of which is:

Definition 55 (Grid Information Model Schema and Service) *A Grid information model is an XML-based schema that provides details about the capabilities of the reference architecture services such as the attribute authority (cf. Definition 52), Grid execution management service (cf. Definition 49), and Grid data management service (cf. Definition 50), and other Grid services (cf. Definition 13). The Grid information model is exposed by the services its describes via a particular interface or via a dedicated information service.*

Definition 55 requires a standard information schema that is able to describe the properties of the reference architecture services and their underlying resources. Examples are GLUE2 [113] and the Common Information Schema (CIM) [5]. The definition also includes the use of information services to expose information based on an information model. There is a wide variety of used information services within production e-Science infrastructures. Examples are systems based on the LDAP [196] standard such as BDII [212] or based on XML [223]. Definition 34 refers to publishing of information with the use of an common information model that is much more important. As a consequence, there is no definition on a precise 'information service' with dedicated query interfaces as core building block being out of the scope and thus this thesis only refers to information services (e.g. based on LDAP [196]) mechanisms where needed.

Definition 34 also consists of a part that is related to the tracking of resource usage that can be refined as follows:

Definition 56 (Grid Usage Record Format Schema) *The Grid usage record format is an XML-based schema that provides standardized details about the resource usage reference architecture services as defined in a Grid execution management service (cf. Definition 49) and Grid data management service (cf. Definition 50) augmented with end-user identities (i.e. accounts).*

Definition 56 is essential to track resource usage within the reference architecture adoptions and an example of such a format is UR [216]. In production infrastructures it is important to track the resource usage of compute resources, but also for storage resources. For the later part, there are currently no direct standards available while there is much potential to extend the UR format to this need.

Finally, Table 4.4 summarizes the aforementioned core building blocks requirements that are part of the reference architecture level.

4.2.2 Improved e-Science Applications Executions

The requirements in the last section defined the key functionality of the reference architecture core building blocks. This section defines specific architectural design requirements that are based on an academic analysis of lessons learned from many Grid interoperability studies surveyed in Chapter 3. Significant limitations in functionality when e-Science applications as defined in Definition 9 have been used in practice on production e-Science infrastructures have been revealed. A reference architecture that overcomes these limitations needs to satisfy requirements listed in this section. Requirements are defined for the reference architecture level as illustrated in Figure 4.3 and related to the aforedefined core building blocks. The concept requirements defined on the reference architecture level in this section are summarized as part of Table 4.5.

Grid technology providers need to provide concrete *'application types'* as follows:

Definition 57 (Application Type Support) *The application type of an e-Science application according to Definition 9 indicates its manner of processing within the Grid middleware (cf. Definition 12) and thus needs to be part of the job description in order to enable more efficient processing.*

Definition 57 requires that each job description document that is submitted to a Grid middleware must have an element that describes which type of e-Science application is submitted. The type refers to the many possibilities of e-Science application submissions in production e-Science infrastructures starting from serial and parallel to pre-installed executions, or even job submissions as part of a larger workflow. One example is the use of a *'benchmark type'* in order to enable better processing within the technologies, since benchmarks are in several cases fundamentally different to production executions (e.g. include measurements) and are performed in specialized execution environments (e.g. to enable fair comparisons).

The exact specification of the e-Science application executable is another requirement that is defined as follows:

Definition 58 (Precise Application Executable Specification) *The precise application executable specification is the need to specify an e-Science application according to Definition 9 with three distinct executable parts in the job description that are the name, path, and arguments of the executable.*

Definition 58 means that job descriptions that are submitted to technologies need to provide the name, path, and arguments of the executable of the corresponding application. This addresses the shortcomings of many proprietary job descriptions obversed in the related work, including the well-known JSDL standard [115]. In many interoperability case studies it is a challenge for middleware to distinguish whether the given information is a relative executable path or a fixed path thus leading to confusion in application executions across Grids. To provide an example, the information about the executable `'bin\pepc n400'` needs to be encoded in the job description in such a way that three elements are part of it. Within this example, the path element is `bin`, the name is `pepc` and one of the arguments is `n400`.

In order to address challenges with respect to the specification of pre-installed application software the following requirement is needed:

Definition 59 (Application Software Mechanism) *The application software mechanism is a functionality that exposes pre-installed software applications (including software libraries, executables, etc.) at a resource in a common manner to be re-used for job description with Grid middleware as defined in Definition 12 using the same consistent syntax and semantics.*

Definition 59 requires a mechanism to seamlessly execute a pre-installed software application at a given resource. The available software is exposed in such a way that it can be re-used within job descriptions in order to ensure consistency with the submission. For example, a Grid resource offers a pre-installed sientific application. It exposes the application via an information service and the information is used by a GUI client functionality that creates the job description on behalf of the end-user and with GUI actions the need particular parameters can be specified.

Another requirement is related to the *'application output joins'*:

Definition 60 (Application Output Joins) *The application output joins specify the need to have a common format and location of all relevant output (std output, std error, etc.) of an e-Science application according to Definition 9 used with Grid middleware as defined in Definition 12.*

Definition 60 raises the demand to join the outputs of executed applications, since in many interoperability setups the join of all output of a particular application makes sense. It enables a more straightforward job execution analysis even by end-users from time to time. For example, several scientific packages (e.g. AMBER [242]) make intensive use of the `stderr` output of an application that in turn is required by scientists to understand whether a particular job run (e.g. molecular dynamics simulation) was successful from a scientific perspective. In this example, the scientists benefit from one joined application output file in order to have a much simpler way of understanding the job execution process. This requirement is also helpful for Grid resource administrators when they seek to understand why a given application was not successfully executed (e.g. when a technical problem exists).

The use of *'common environment variables'* is another requirements that is defined as follows:

Definition 61 (Common Environment Variables) *Common environment variables are a common set of variables that are available at each computational resource in the sense of Definition 6 enabling their use within e-Science applications according to Definition 9.*

Definition 61 addresses a problem around application portability that seems to be trivial, while experience from working with Grid applications across infrastructures reveals the opposite [38]. Real production applications make use of environment variables (e.g. number of cores, memory, etc.) that are all differently encoded at Grid resource levels today. For example, one particular e-Science application that use environment variables is able to use these variables during execution, because these have the same syntax and also semantics at any available resource being part of an individual infrastructure as defined in Definition 20.

Another requirement is related to pre-installed software, but those software applications that require an enormous amount of configuration leading to the following requirement:

Definition 62 (Common Execution Modules) *Common execution modules are pre-configured execution environments that include path settings, environment variable definition, and other configurations for a specific software application available at a resource (cf. Definition 6). A common execution module must be exposed by a Grid resource in order to enable their use in job descriptions.*

Definition 62 raise the demand for a pre-configured execution environment since several scientific packages (e.g. AMBER [242]) require the setup of different paths, environmental variables and other configurations for their executables. The sourceforge module concept [20] is one option for its realization. For example, a specific library such as the Visualisation Interface Toolkit (VISIT) [139] steering library is needed that is available at a resource and exposed in a manner that it can be re-used in the job description for an e-Science application that uses it.

Finally, Table 4.5 summarizes the aforementioned concept requirements that are part of the reference architecture level.

No.	Requirement Definition Title
57	Application Type Support
58	Precise Application Executable Specification
59	Application Software Mechanism
60	Application Output Joins
61	Common Environment Variables
62	Common Execution Modules

Table 4.5: Reference architecture e-Science applications requirements (reference architecture level).

4.2.3 Improved Processing and Data-staging Capabilities

This section defines further architectural design requirements that are based on lessons learned from Grid interoperability field studies and their associated academic analysis. But the requirements in this section are more related to Grid middleware processing and data-staging capabilities. The e-Science applications also take advantage of these requirements, but the requirements have a much greater impact on the reference architecture adoptions in Grid middleware. Therefore, the requirements in this section are defined for the reference architecture level as illustrated in Figure 4.3, but are also related to the aforedefined core building blocks. The concepts requirements defined on the reference architecture level in this section are summarized as part of Table 4.6.

In many specifications (e.g. JSDL [115]) the relevance of HPC is often lower than the traditional Grid methods, often being referred to as rather HTC-driven. Several gaps have been identified with Grid applications executed across infrastructures where more recent features of large-scale HPC resources had been not supported. The requirement is thus as follows:

Definition 63 (State-of-the-art HPC Support) *State-of-the-art HPC support stands for a wide variety of features that modern large-scale HPC resources provide (e.g. different network topologies, shape setups, or task/core mappings). These features need to be exposed by resources (cf. Definition 6) so that they can be easily re-used within job descriptions of e-Science applications according to Definition 9.*

Definition 63 addresses the growing complexity of HPC machines in the last couple of years since they support more and more features that have been not addressed in existing job description languages such as JSDL. The basic HPC usage can be described, but additions are needed to increaese the performance of e-Science applications which is particularly important for the use in HPC-driven e-Science infrastructures as defined in Definition 16. For example, a large-scale HPC resource needs to expose its characteristics such as supported network topologies [290] or available shapes [290] and task/core mappings [290]. In turn, e-Scientists use this information to identify a suitable system for their e-Science application and a system that creates the job description on behalf of the end-user is able to re-use this information too.

Another important lesson learned from Grid interoperability work is that the access of key information about a Grid resource is exposed in a standardised manner. This leads to the following requirement:

Definition 64 (High Message Exposure) *High message exposure refers to a standardised way of providing key information about a resource (cf. Definition 6) with important messages (i.e. high messages) to end-users, Grid application developers, or even administrators.*

Definition 64 raises the demand for a standardised way of exposing key information by Grid resources. In HPC-driven interoperability studies, in particular, many Grid resources have their own ways of informing end-users about scheduled resource maintenance and other important pieces of information (e.g. dashboards). In some cases different Web sites have this information about a Grid resource and its news about status changes. For example, an e-Scientist plans to migrate from one HPC machine to another since one machine is going to be offline. In order to submit jobs and to retrieve the key information about the status with *'high messages'* on this system, the end-user just needs to use a usual Grid client instead of looking on several Web sites. In the reference architecture, there must be a standardised way of exposing this information that points to the use of an information system using a standardised information model.

The next requirement is based on an academic analysis of application usage that point to the limitations in approaches with a lack of supporting *'sequences'*. The difference from sequence to workflows is that each sequence step needs to be executed at the same Grid resource while workflow steps can be potentially executed in different locations. The requirement is as follows:

Definition 65 (Computational Job Sequences) *Computational job sequences is a mechanism that enables the execution of pre- and post-processing applications before or after the main e-Science application as defined in Definition 9 using Grid middleware (cf. Definition 12).*

Definition 65 addresses several requirements when working with scientific applications in production e-Science infrastructures. In many cases, the existing job description languages is unable to support pre-job sequences enabling pre-processing activities or compilations before the main executable execution. Post-processing is also not available in many description languages such as JSDL [115]. This is a limitation, for example, for e-Scientists that like to analyse several application outputs directly after its running period with a small program. One practical example of Definition 65 is the transformation of input data before the main Grid job execution that can be done using the defined pre-processing functionality.

A closer academic analysis of current approaches by e-Scientists identified another limitation of middleware providing no *manual data-staging*. The requirement is as follows:

Definition 66 (Manual Data-staging Mechanism) *A manual data-staging mechanism is functionality within a Grid middleware (cf. Definition 12) that enables end-users to manually stage data into the Grid job sandbox in order to be used with the main e-Science application according to Definition 9.*

Definition 66 addresses an important identified limitation for e-Scientists, especially in those areas of work where manual data checking is needed. One example of this requirement is that e-Scientists often analyse data sets before they are part of inputs to e-Science application runs. The expertise of e-Scientists is needed in order to choose the input data or select important fractions of it to be computed. Automated data-staging, as defined in JSDL [115], does not have access to this specific expertise and as such only provides the possibility to transfer all or nothing. In several interoperability case studies (e.g. Chapter 6.3), these steps are often combined, meaning that at first automated data-staging is used to transfer all the data before several parts of it are dropped before the main application is run in order to reduce it to a meaningful input.

More flexibility in the Grid job execution control is another identified requirement that is defined as follows:

Definition 67 (Grid Job Manipulation Functionality) *A Grid job manipulation functionality within a Grid middleware (see Definition 12) enables end-users to manipulate the e-Science application (cf. Definition 9) executions after their submission to a Grid resource. This includes pausing and resuming them, canceling and completely removing them as well as their resubmission if necessary.*

Definition 67 raises the requirement to influence the job run after its submission. This is often not covered by existing specifications such as the OGSA-BES standard [169], which only

No.	Requirement Definition Title
63	State-of-the-art HPC Support
64	High Message Exposure
65	Computational Job Sequences
66	Manual Data-staging Mechanism
67	Grid Job Manipulation Functionality

Table 4.6: Reference architecture processing and data-staging requirements (reference architecture level).

provides the `TerminateActivities` operation. Although Definition 67 sounds trivial, its implementation often depend on the functionality of different resource management systems (e.g. pause) such as defined in Definition 11. Definition 67 implies impacts on the state model [169] of Grid middleware defining the transitions of states (e.g. pause to resume). One example is that the e-Scientists would like to check whether data that was staged to the computational resource is correct before a long full-blown production run begins on a large-scale HPC resource. Time on these resources is rare and as such many e-Scientists would like to double check whether their execution really takes the right input data. In this context the job will pause after the staging until the e-Scientist explicitly resumes the job.

Finally, Table 4.6 summarizes the aforementioned concept requirements that are part of the reference architecture level.

4.3 Non-functional Requirements

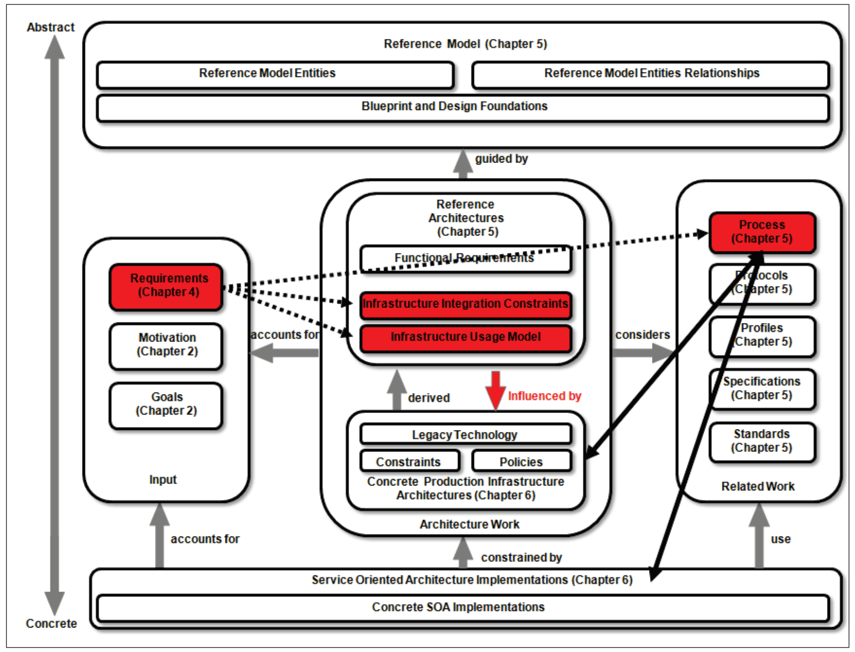


Figure 4.4: Non-Functional requirements of the architecture work with production infrastructures influence.

This section raises several non-functional requirements that need to be supported by reference architecture adoptions in existing production e-Science infrastructure environments as shown in Figure 4.4. It shows that requirements are defined on the reference architecture level with *'infrastructure integration constraints'* and *'infrastructure usage models'*. But the figure also illustrates how specific operational influences affect the requirements in this section arising from production e-Science infrastructure deployments to achieve production-oriented architectures. In addition, requirements are defined for a whole process to sustain interoperability between reference model adoptions that are not directly mapped to any particular level as shown in Figure 4.4. Many requirements in this section are thus influenced by production technology implementations (cf. Figure 4.4) in order to ensure that the requirements have a practical impact on existing production e-Science infrastructures.

Training as a whole is one element of production e-Science infrastructure and represents another non-functional requirement. But this topic is kept out of this thesis and tackled by the e-IRG ETTF [206].

4.3.1 e-Science Production Infrastructure Integration Constraints

This section sets several constraints to make concrete implementations and deployments of the reference model and its associated architecture work relevant in the context of production e-Science infrastructures as defined in Definition 5. They address certain operational requirements that are necessary to achieve smooth infrastructure integration and to be considered at all for deployments in production. Figure 4.3 reveals that the requirements in this section are on the concrete architecture level but with significant input from the real implementations in production e-Science infrastructures. The requirements defined on the reference architecture level in this section are summarized as part of Table 4.7.

The first requirement concerns the requirement of having adoptions with real e-Science production technologies that is as follows:

Definition 68 (e-Science Production Technology Adoption Constraint) *e-Science production technology adoption sets the constraint for reference architecture adoptions in that they need to be integrated by preserving production stability to address slow migration and update cycles. Adoptions need to be relevant in the context of existing production e-Science infrastructures according to Definition 5 and their deployed production Grid middleware as defined in Definition 12.*

Defnition 68 sets a strict requirement for implementation, but on the other hand ensures that reference architecture adoptions have a real production impact. This is only assured when adoptions are done in technologies that are likely to be deployed in practice. Examples in the field of e-Science infrastructures would be EGEE/EGI, DEISA/PRACE, TeraGrid/XSEDE or its production middleware systems ARC, gLite, Globus, GridSAM, GENESIS, or UNICORE.

Definition 43 implies the requirement for information and security constraints for the reference architecture. Parts of it are also non-functional requirements that are described here. The requirement towards the realisation of an *'overall information ecosystem'* between the entities of the reference model is required as follows:

Definition 69 (e-Science Infrastructure Information Ecosystem) *An e-Science infrastructure information ecosystem is based on Grid information elements as defined in Definition 34 and sets the constraint that each reference model entity in e-Science infrastructure deployments must be described via a common information model according to a common information exchange policy at any time.*

Definition 69 ensures the adoption of a common information model and information exchange policy across the entities of the reference model and as such across the core building blocks of the associated reference architecture. At any given time it must be possible to obtain information in a common format about any given core building block within the concrete reference architecture adoptions. This also includes the way in which information is obtained (i.e. information exchange policy). This clarifies the way of how information is obtained (e.g. through a well-defined information service) and prevents semantic loss when transforming key information from one model to another (i.e. transformation logic). One example is that end-users can mostly assume that relevant information is present about a particular execution management service (i.e. core building block) as defined in Definition 31. Further, the end-user assumes that this service is described using a common information model exposing consistent properties (e.g. amount of CPUs/cores, memory, etc.).

The second constraint raises the requirement of the realisation of an *'overall resource tracking ecosystem'* between the entities of the reference model that is defined as follows:

Definition 70 (e-Science Infrastructure Resource Tracking Ecosystem) *An e-Science infrastructure resource tracking ecosystem is based on Grid information elements as defined in Definition 34 and sets the constraint that each resource usage through a reference model entity in e-Science infrastructure deployments must be tracked via a common resource usage model according to a common resource tracking exchange policy.*

No.	Requirement Definition Title
68	e-Science Production Technology Adoption Constraint
69	e-Science Infrastructure Information Ecosystem
70	e-Science Infrastructure Resource Tracking Ecosystem
71	e-Science Infrastructure Attribute-based Authorisation Ecosystem

Table 4.7: Reference architecture infrastructure integration requirements (reference architecture level).

Definition 70 ensures the adoption of a common resource tracking model and exchange policy across the entities of the reference model, and as such across the core building blocks of the associated reference architecture. At any time that a specific data or computational resource is used, it must be appropriately tracked in a common format for each corresponding core building block within concrete reference architecture adoptions. The way in which this resource usage information is obtained must be known (i.e. resource tracking policy). This clarifies the way in which resource usage information is obtained (e.g. through a well-defined specific information service) and prevents semantic loss when transforming key information from one resource tracking model to another (i.e. transformation logic). One example is that administrators assume that resource usage information is present about an execution management element as defined in Definition 31 or about a data management element as defined in Definition 32. These service elements have to use a common resource tracking model that logs end-user resource usage data (e.g. amount of used CPUs/cores or storage capacity, etc.).

The third constraint raises an operational requirement towards the realisation of an *'overall attribute-based authorisation ecosystem'* across the entities of the reference model as follows:

Definition 71 (e-Science Infrastructure Attribute-based Authorisation Ecosystem) *An e-Science infrastructure attribute-based authorization ecosystem is based on Grid security elements as defined in Definition 33, and sets the constraint that each reference model entity in the e-Science infrastructure deployment must be protected by a common attribute-based security model and be according to a common authorisation policy.*

Definition 71 ensures the adoption of a common attribute-based security model and fine-grained authorisation policy across the entities of the reference model and across the core building blocks of the associated reference architecture. Dedicated mechanisms that work with a common format to encode security attributes are needed for each corresponding core building block within the concrete reference architecture adoptions that offer resource access. The way in which such security attributes are obtained must be known (i.e. common authorization policy). This clarifies the way of how such security credentials are obtained (e.g. through a well-defined attribute authority service) and prevents the loss of identity information when transforming key information from one security attribute model to another (i.e. transformation logic). One example is that end-users assume that their security attributes are working with execution management elements, such as those defined in Definition 31, or data management elements, such as those defined in Definition 32 whilst in different e-Science infrastructures. These service elements have to use a common security attributes model with a source of trust that releases such security attributes about end-users and several services that are able to correctly interpret the attributes to enforce authorisation decisions based on fine-granular security policies.

Finally, Table 4.7 summarizes the aforementioned requirements that are part of the reference architecture.

4.3.2 Requirements for Interoperable Infrastructure Usage Model

This section defines non-functional characteristics that a concrete architecture needs to provide to offer added-value as defined in Definition 3. It should not be underestimated that those architecture implementations that do not take these non-functional requirements into account often have less chances to be deployed in production e-Science infrastructures. Figure 4.3 reveals that the requirements in this section are also on the concrete architecture level but with having also significant input from the real implementations in production e-Science infrastructures. These requirements are crucial for the infrastructure provider to ensure a consistent provisioning of production e-Science infrastructures that is only possible with those technologies that take the following requirements into account. These are crucial to establish individual infrastructures (cf. Definition 20). A general concept that represents a fundamental non-functional requirement, especially in Grid interoperability setups, the usability of solutions. In the context of using multiple production e-Science infrastructures, the usability is often fundamentally reduced in Grid interoperability application use cases. Different client technologies are often used independently of each other and these require different security credential setups whilst using the same end-user certificate. For example, a command-line client (e.g. the gLite-UI [212]) is used, and in addition the graphical UNICORE Rich Client [152], is used in parallel. The important approach of Grids referring to *single sign-on* [131] is thus broken. The following non-functional requirement ensures a *transparent infrastructure usability* by concrete reference architecture implementations:

Definition 72 (Transparent Infrastructure Usability) *Transparent infrastructure usability is a concept that enables end-users to use different e-Science infrastructures with the same technology setup, ensuring the use of the same credentials in general and preserving single sign-on in particular.*

Another non-functional requirement is the flexibility for end-users to choose the Grid resources they require without being bound to one specific production e-Science infrastructure. Quite the opposite is often observed in infrastructures today. End-users have to use the technology of choice of one specific infrastructure, such as UNICORE in DEISA/PRACE or gLite in EGEE/EGI. End-users only choose Grid resources within the boundaries of one specific infrastructure that is not interoperable with another (cf. Definition 19), because of the technology. This leads to the following requirement:

Definition 73 (Flexibility to Choose Resources) *Flexibility to choose resources stands for a concept that enables end-users to choose the technologies they desire through mechanisms to prevent vendor-locks. This concept also implies that end-users are able to choose infrastructure resources (cf. Definition 6) without being bound to one production e-Science infrastructure.*

Definition 73 is similar to the next non-functional requirement *'Multiple Infrastructures Usage Performance'*. The 'multiple infrastructures' here stand for interactions across two or more e-Science infrastructures. Chapter 3 reveals that in many cases the transformation logic (cf. Definition 27) is the reason why performance decreases when using multiple infrastructures with one specific client technology. The transformation process from one protocol/schema to another often takes more time than the use of one protocol or schema. This leads to the following requirement about avoiding the use of transformation logic to ensure good performance:

Definition 74 (Multiple Infrastructures Usage Performance) *Multiple infrastructures usage performance is a concept that stands for the requirement that job execution and response times are not increased with transformation logic (cf. Definition 27) when multiple e-Science infrastructures are used.*

Another requirement for reference architecture implementations is *'Infrastructure Resource Usage Efficiency'*. One fundamental drawback of most of the approaches in Chapter 3 is that in many cases the technologies have been implemented towards the use of multiple infrastructures, whilst partly losing important low-level functional capabilities of infrastructure resources. Resource-specific capabilities that would preserve the efficient use of infrastructure resources have been neglected to enable at least minimum interoperability.

One example is the use of HPC-based infrastructures as defined in Definition 16. The HPC-BP [154] specification supports HPC, but deeper investigation reveals that an efficient use of parallel resources is not possible without several additions (e.g. network topologies [290]). This leads to the following requirement for reference architecture implementations:

Definition 75 (Infrastructure Resource Usage Efficiency) *Infrastructure resource usage efficiency is a concept that enables the low-level use of resource capabilities and access to necessary functionalities in order to make possible the most efficient use of an infrastructure resource (cf. Definition 6).*

Architectures that satisfy Definition 75 will enable better resource consumption, job throughput, and thus an overall lower-time-to-solution. HPC resources in particular are often oversubscribed and thus their efficient use is a must of any architecture that takes advantage of this highly precious infrastructure resources. Definition 75 is another example of how non-functional requirements can essentially make a difference in whether architecture implementations have a realistic choice of being deployed and used in production.

A huge set of non-functional requirements are also summarised under the broad term supportability. In this thesis, this term stands for maintainability, extensibility, installability, and testability of any reference architecture implementation. All these aforementioned aspects play a significant role and many approaches in Chapter 3 revealed that these non-functional requirements are often neglected. Several implementations are thus not deployed in production e-Science infrastructures being essentially only available in testbeds.

One example is the provisioning of a TSI adapter within UNICORE to bridge to gLite installations [256]. This non standard-based adapter was never part of the main UNICORE release and thus its maintainability was decreased, and only the developer was able to install this workaround. This not directly maintained adapter significantly decreased the installability of the solution by infrastructure providers. It is very specific, providing no room for further extensibility. Reference architecture implementations therefore need to satisfy the following requirements:

Definition 76 (Supportability) *Supportability is a concept that ensures that reference architecture implementations are released as part of the official technology releases in order to provide a sophisticated level of supportability (i.e. maintainability, extensibility, installability, testability).*

Finally, architecture adoptions have to implement certain *'patterns'* (cf. Figure 4.1) that enables trust and security as well as an overall run-time concept. These patterns describe how the reference architecture is used to overcome limitations mentioned in Chapter 2 (e.g. joint HPC and HTC usage). These architecture work patterns are defined as follows:

Definition 77 (Architecture Work Patterns) *Architecture work patterns are architectural and run-time models that take advantages of key functionality of the reference model and its associated architecture elements in order to give insights how interoperability is promoted with it in a secure manner giving end-users feedback about the state of the computational jobs.*

Definition 77 raises the demand that there must be dynamic run-time information beyond static descriptions of the reference model guided architecture work. This also includes 'a way

No.	Requirement Definition Title
72	Transparent Infrastructure Usability
73	Flexibility to Choose Resources
74	Multiple Infrastructures Usage Performance
75	Infrastructure Resource Usage Efficiency
76	Supportability
77	Architecture Work Patterns

Table 4.8: Reference architecture interoperable usage requirements (reference architecture level).

of using the infrastructures' that can be defined as an algorithm to better understand how users take advantage of reference architecture adoptions. Another example is a state-model [115] that provides run-time information about the architecture in terms of which status a particularly executed e-Science application has giving end-users a certain amount of feedback. Both the aforementioned aspects are on the border-line to be also functional requirements, but here the *'end-user feedback'* and *'way of using'* is much more important than its technical realization with patterns underneath.

Finally, Table 4.8 summarizes the aforementioned requirements that are part of the reference architecture.

4.3.3 Process Requirements for Sustained Infrastructure Interoperability

This section provides definitions for an associated process to the reference model and its architecture as shown in Figure 4.4. The requirements defined on the reference architecture level in this section are summarized as part of Table 4.9.

One conclusion of Chapter 3 was that a reference model and its standard-based associated architecture work bears the potential of significantly increasing the interoperability of e-Science infrastructures (cf. Definition 21). But it was also concluded that a whole process is required that preserves the interoperability between infrastructures once their boundaries are lowered. This process is defined as follows:

Definition 78 (Process for Sustained Infrastructure Interoperability) *The process for sustained infrastructure interoperability is a multi-segmented process that is defined around a reference model and its associated architecture. It defines rather non-functional segments addressing sustainability and the evolution of standards, and the synergy of collaboration between involved stakeholders.*

As Definition 78 reveals, there is a need for a process that tackles non-functional aspects and that is complementary to the technically-driven reference model aspects. This includes issues around collaboration and fundings while more detailed requirements are given in the subsequent paragraphs. As a consequence, Figure 4.3 illustrates that the particular requirements in this section are complementary to the architecture work but not directly defined on one particular level but is related to production e-Science infrastructures and their different technologies that are deployed on them.

One key segment of the process is the creation of a reference model that is defined as follows:

Definition 79 (Common Reference Model Creation) *The common reference model creation stands for a process where technology providers that contribute to interoperable e-Science infrastructures (cf. Definition 21) work together on an agreed reference model including associated architecture elements. The entities of such a reference model formed as concrete building blocks within the reference architecture must be based on open standards (cf. Definition 14).*

The requirement in the sense of Definition 79 is an important process aspect that bears the potential to increase the interoperability between production e-Science infrastructures.

In order to have a significant impact of the aforementioned definitions in the process, the following requirement is also very important:

Definition 80 (Key Technology Providers Collaboration) *The key technology providers collaboration is a process where those technology providers that have a significant impact with their technologies on production e-Science infrastructures, in the sense of Definition 5, commonly work together.*

Definition 80 prevents that interoperability and interoperation of e-Science infrastructures is hindered by having not all the key players involved in the collaboration that *'can actually make a difference'*. This means the change of a setup of technologies deployed on production e-Science infrastructures, because this is crucial to enable individual infrastructures (cf. Definition 20).

Over the last few years, work in SDOs around GIN has shown that theoretical specification work should be augmented with a complementary reference implementation defined as follows:

Definition 81 (Reference Implementation Developments) *The Reference implementation developments is work that takes early versions of emerging open standard specifications and reference architecture core building blocks as input and provides practical development and implementation feedback on its realisation during the definition phase of design, architecture, or specification.*

Definition 81 raises the demand to have prototypes for emerging specifications, standards, and for the architecture elements of the reference model. It enables practical feedback to theoretical work and is in many cases also a contribution to interoperable solutions since the early adoption of standards provides significant insight into its practical realisation. The latter is often neglected and as such several standards are less adopted by technology providers since they are practically non-relevant or considered to be too academically-driven to be useful.

The next requirement raises the demand for a process segment that covers the adoption of open standards addressing their evolution:

Definition 82 (Open Standard Evolution Process) *The open standard evolution process refers to the improvements of open standards based on their use in production e-Science infrastructures in the sense of Definition 5, that provides feedback, which must be fed back into the standardisation process.*

Defnition 82 acknowledges that no standard specification is perfect, and that they evolve over time as their usage increases within infrastructures. But a systematic approach is required to take the amount of experience in using standards back into the corresponding SDOs. Often specific SDO working group are not longer in existence or the combined use of standards makes it extremely complicated to decide which group needs to tackle which lessons learned.

One of the problems is also the wide variety of technology providers that are relevant for production infrastructures that have different roadmaps and are not aligned to future strategies. Interoperability of e-Science infrastructures can be significantly increased with the following process requirement:

Definition 83 (Future Strategy Sharing and Common Roadmap Definition) *Future strategy sharing and common roadmap definition are process elements that are required to be performed by technology providers in collaboration with e-Science infrastructure stakeholders in order to increase the interoperability of deployed technologies in the long-term perspective.*

Definition 83 is not easy to achieve, while at the same time significant interoperability progress can be achieved when technology providers align their roadmaps and work together under a common project umbrella (e.g. OMII-Europe [69]).

No.	Requirement Definition Title	Reference Model Context
78	Process for Sustained Infrastructure Interoperability	Overall Associated Process
79	Common Reference Model Creation	Associated Process Element
80	Key Technology Providers Collaboration	Associated Process Element
81	Reference Implementation Developments	Associated Process Element
82	Open Standard Evolution Process	Associated Process Element
83	Future Strategy Sharing and Common Roadmap Definition	Associated Process Element
84	Operation Policy Harmonisation	Associated Process Element
85	Funding and Cross-Project Coordination	Associated Process Element

Table 4.9: Reference architecture associated process requirements summary.

Interoperability studies in Chapter 3 with applications have worked on increasing the interoperability between technologies, for instance, by adopting common open standards. But this is only the technology foundation since it is not guaranteed that the policies of the corresponding infrastructures actually allow for the use of such technologies. The following requirement is thus needed:

Definition 84 (Operation Policy Harmonisation) *The operation policy harmonisation is a process where different usage models of different production e-Science infrastructures in the sense of Definition 5 that are governed by various policies are harmonised in order to enable true interoperations.*

Definition 84 addresses a major showstopper for current Grid interoperability applications that rely on different policy definitions. Examples are areas of resource allocation (e.g. VO vs. peer-reviewed grant-based access) or security (e.g. proxy-based setups or full certificates).

Another requirement has the potential to increase interoperability today and in the future:

Definition 85 (Funding and Cross-Project Coordination) *Funding and cross-project coordination is a process where different funding bodies interact in order to ensure coordinated project grants across national boundaries or to fund neutral projects that push for common community needs.*

Definition 85 refers to the requirement of having common initiatives at the highest possible level to promote interoperability in every possible way to achieve a network of interoperable services (cf. Definition 15) vision.

Finally, Table 4.9 summarizes the aforementioned requirements that are part of the associated process to increase interoperability over time.

4.4 Conclusion

The conclusion of Chapter 3 highlights the relevance of open standard adoptions that should be guided by a reference model and associated reference architecture in order to promote e-Science infrastructure interoperability. Based on academic analysis of Chapter 3 results, Chapter 4 defines concrete requirements for such a reference model with entities and relationships. More concrete associated reference architecture core building blocks are also defined. Adoptions of a reference model and its associated elements that are able to satisfy these requirements are able to overcome the limitations of not interoperable infrastructures identified in Chapter 2 and 3 respectively. Chapter 4 specifies many important elements towards answers for the major research question of this thesis. These requirements for a reference model guide the architectural design of a *'network of interoperable services for production e-Science infrastructures'* to be carried out in the next chapter.

These requirements are defined on different abstraction levels, from the very abstract level to the more concrete. High-level requirements about the reference model blueprint and its entity requirements, including their interactions have been defined. This chapter underpinned these abstract requirements with relevant concrete reference architecture requirements. These architecture requirements cover general communication behaviour among the reference model entities (e.g. Web services, SOAP, etc.), but also the role of common open standards that have been identified as being crucial in Chapter 3.

The requirements need to be functional and non-functional, in order to address the broad set of challenges that are described in Chapter 2. Lessons learned from the academic analysis in Chapter 3 is taken into account in the requirements so that experience from different scientific application case studies is used. One of the conclusions of this chapter is also that existing work in the field needs more functionality for recent HPC systems and better application support mechanisms. E-Scientists need more control when interacting with middleware reaching from the change of the job run after its submission to performing manual data-stagings.

The non-functional requirements focus on production e-Science infrastructure integration constraints and process requirements complementing theoretical work with practical insights. One conclusion so far is that only technologies that already play a major role in production e-Science infrastructures (e.g. UNICORE, gLite, etc.) have a realistic chance of being deployed with reference model and architecture adoptions. New technologies that adopt the reference model are an option, but still need to convince infrastructure providers about their reliability in terms of functionality and sustainability. This is a process that take many years and this will not change over night. Instead, changes to already deployed technologies are easier to get into daily e-Science infrastructure production. A reference architecture with open standards is a fundamental step in the right direction, but it must be complemented with a whole process that takes the dynamics of change and policy aspects of infrastructures into account. Production e-Science infrastructures are governed by various policies within given boundary conditions that not only consist of the continuation of production quality, but also of complex governance structures and technology setups that have been established for many years. Requirements for several process segments that are able to engage in the challenges of change as a long-term activity have been defined. Another conclusion is surely the requirements of a concrete process that is necessary to preserve the success of a technical reference model and its associated reference architecture. This is needed to sustain and improve interoperability between infrastructures to establish individually formed infrastructures for end-users.

Chapter 5

Architectural Design

The reference model in this chapter embodies the basic goal of an *'interoperable network of Grid services'* to achieve production e-Science infrastructure interoperability. The goal is that this model and its associated architectural design elements can be referred to for various purposes and consist of a number of concepts that all rolled up into the *infrastructure interoperability reference model (IIRM)*. As a reference model, it is more abstract than a framework, since it also deals with non-middleware-specific aspects and infrastructure deployment issues. Guidance on how interoperability can be sustained is provided as an associated process based on seven segments. The scope is thus broader than a framework in order to address the great complexity of interoperability problems described in Chapter 2 taking into account that they cannot be solved by component-based approaches as revealed in Chapter 3.

The problem space and a model of the problem was created in Chapter 2, followed by comparisons of the problems with similiar issues and an analysis of a wide variety of existing solutions in Chapter 3. Based on this, concrete requirement definitions have been the focus of Chapter 4. The aim of this chapter is to provide insights into the second major part of the analysis efforts in this thesis that takes the lesson learned of practical field experiences conducted over years into account. These have been addressed in two major ways that are detailed *design and concepts* (for production implementations) and an aligned *process* (for sustained solutions). Chapter 5 defines what collectively is referred to under the umbrella of the *'IIRM'* including associated elements necessary for use by real applications described in the next chapter.

The first section provides an architectural blueprint for a reference model with associated elements (i.e. reference architecture, patterns, etc.) that is specifically designed to enable the interoperability between production e-Science infrastructures. It identifies the reference model entities and their relationships taking the requirements of Chapter 4 (e.g. TCP/IP approach) into account as well as key aspects of practical field studies (e.g. need for standards) reviewed in Chapter 3. It provides solutions to overcome infrastructure integration issues when deploying IIRM solutions on production e-Science infrastructures, including the definition of necessary invariants. Complementary to the fundamental architectural design, the second section reveals results of academic analysis of practical field tests as a set of standard-based concepts aiming to improve e-Science infrastructure interoperability and the efficiency of e-Science applications. In the third section, further analysis and proposed solutions culminate in the definition of a medium to long-term orientated process providing concrete segments that cover policy, coordination, and cooperation issues. The reference model is thus aligned with a process that has the potential to enable and sustain infrastructure interoperability.

5.1 Reference Model Design and Associated Architecture Work

Based on the lessons learned from the survey of related work in Chapter 3 and the definition of relevant requirements in Chapter 4, this section reveals an architectural blueprint of the proposed reference model and its associated architecture elements. The subsequent sections provide several pieces of architecture work (e.g. reference architecture, patterns, standards, etc.) that are all specifically designed to enable interoperability between production e-Science infrastructures as motivated in Chapter 2.

5.1.1 The Infrastructure Interoperability Reference Model Design

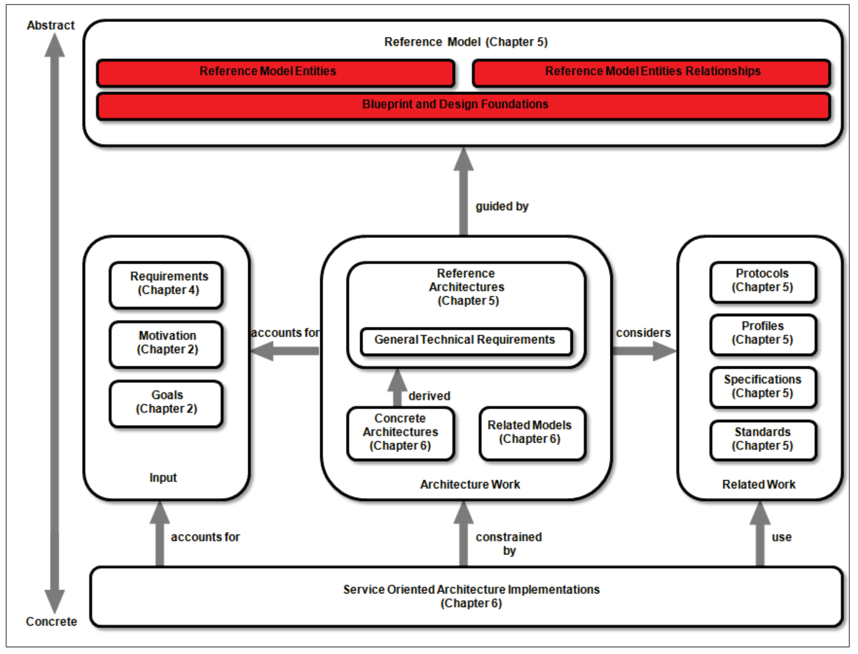


Figure 5.1: The reference model guides many other associated architectural design elements.

The reference model is used as an *'umbrella term'* for the contributions of this thesis, including its associated architecture elements. The definition of the abstract reference model itself is the focus of this section as shown in Figure 5.1 (marked in red). Associated architecture work elements are defined in subsequent sections. The overall concept of the abstract design takes the service-based reference model requirements according to Definition 29 into account.

Figure 5.2 illustrates the abstract reference model that is designed to be in-line with Definition 28 in the context of the given e-Science environment. It is *'abstract'*, because its entities and concepts are an abstract representation of the entities that often exist in production infrastructures. The entities follow a service-based approach and actual Grid services deployed on e-Science infrastructures may have certain performance characteristics, but the concept of a Grid service itself is here relevant and not the particular deployment. Figure 5.2 illustrates not a particular reference installation, or any concrete deployment of implemented reference architecture core building block services, and instead focus on the concept of an abstract Grid service itself.

Another applied principle is that the reference model defines *'entities'* for various functionality and *'relationships'* between them. Figure 5.2 illustrates that all the reference model entities are interconnected with each other in some form or another. But the reference model does not follow a common layered design approach like, for instance, the TCP/IP or ISO/OSI models described in [295].

The third important principle that governs the reference model architectural design is that it does not attempt to describe the whole Grid nor solve all Grid problems that exist. This reference model defines a *'clear problem space'* that is used to clarify *'a network of interoperable Grid services within production e-Science infrastructure environments with relevant storage, HPC, and HTC resources'*. Figure 5.2 illustrates the specific ecosystem (clients, infrastructures, resources, etc.) wherein the reference model as a whole is embedded and thus represents an abstract representation of the given problem space. The reference model and associated architecture elements thus 'only' provide a collection of solutions to tackle certain *'known interoperability problems'* within those particular environments with several specific approaches.

The reference model is also *'Grid technology agnostic'*. It would be not useful if it would make assumptions about specific Grid technologies such as Grid middleware systems (e.g.

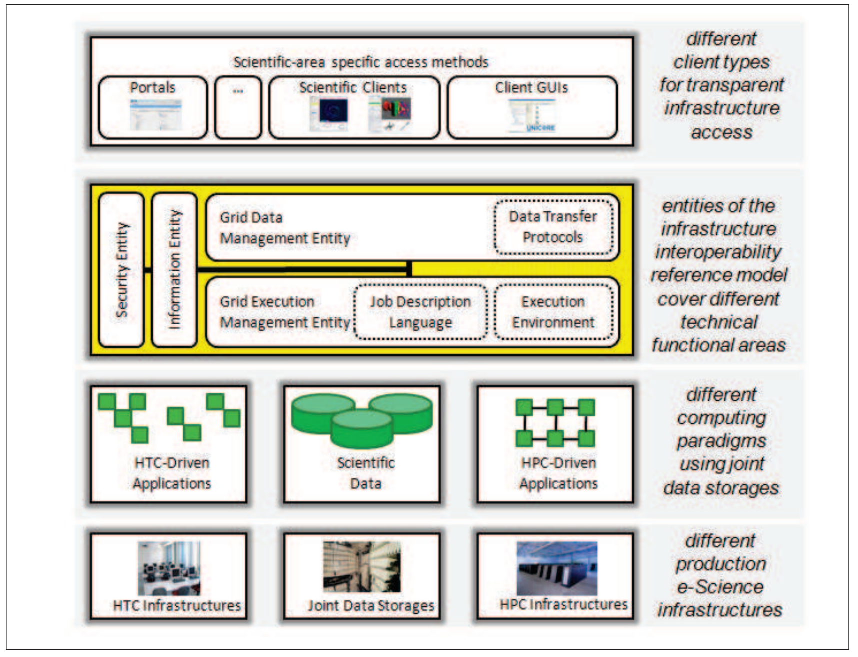


Figure 5.2: The abstract reference model defines basic entities and relationships in a clear problem space.

gLite, UNICORE, etc.) as defined in Definition 12. These technologies are often deployed in *'production e-Science infrastructure environments'* (i.e. EGEE/EGI, DEISA/PRACE, etc.) but are not part of the reference model as shown in Figure 5.2. The IIRM is a mechanism for understanding the interoperability problems faced in the specific environment of e-Science, not the particular production infrastructures or Grid middleware solutions involved. It describes the reference model and its associated elements independent of real-world Grid technologies or infrastructure deployments in order to provide value for adoptions from other e-Science domains (e.g. ESFRIs). Figure 5.2 provides an overview of the reference model, but it lacks many other related entities such as those arising from the operational infrastructure areas of monitoring or support helpdesks (e.g. ticketing) just to provide two examples. Although these areas are highly relevant for production e-Science infrastructures and their seamless operation, these entities are considered out of scope of this thesis as described in Section 1.3 and therefore not illustrated in 5.2.

But for the sake of completeness, further related entities are included, but not limited to, Table 5.1 alongside some known *'interoperability challenges'* that are topics out of the thesis scope. Also, many of these listed entities are not used by the majority of the production e-Science infrastructure end-users (e.g. advance reservation entity). In contrast, this thesis focus is described in Section 1.2 and as such the thesis focusses on those entities that directly support computational activities and related data-staging functionality for e-Science applications that also represent the most used entities by end-users in the given thesis problem space today. As a consequence, the thesis scope also influences the definition of some of the following entities. For example, the EUDAT [22] infrastructure is a similiar data-oriented e-Science infrastructure but would define a *'data management entity'* slightly different highlighting functionalities such as *'policy-based data management'* [229] or the use of PIDs [28].

The reference model is designed to be applicable to other e-Science infrastructures (cf. Definition 4) that take advantage of similiar infrastructure resources (cf. Definition 6) such as storage and computational resources illustrated in Figure 5.2. One example are the wide vari-

Out of Scope Entity	**Interoperability Challenges**
Monitoring Entity	Monitoring the status of Grid services is performed with different tools (e.g. NAGIOS [126]) and interoperability is needed to retrieve a status across infrastructures
Ticketing Entity	Ticketing and help desks for support are using different systems (e.g. [37, 10]) and interoperability is needed to exchange tickets between different infrastructures
Advance Reservation Entity	Advance reservation [285] enables the use of multiple (HPC) infrastructures at the same time and an exchange on reservation slots for interoperable infrastructures is needed to use them in parallel
Credential Transformation Entity	e-Science infrastructures use a wide variety of different security credentials (e.g. [237, 195]) and a transformation service (e.g. [17]) from one to another format support interoperability (using transformation logic)
License Entity	e-Science applications partly rely on software licenses that require license services (e.g. [90]) for distributed systems and their interoperability is crucial when using multiple infrastructures
Domain-specific Entity	e-Science infrastructures also deploy some scientific domain-specific services (e.g. [192]) and their seamless interoperability with security setups is needed

Table 5.1: Entities relevant for production e-Science infrastructures that are out of scope.

eties of research infrastructures (RIs) currently established as part of the ESFRI roadmap [274], as introduced in Section 2.1.3. Although such infrastructures are considered to use the existing production e-Science infrastructures (cf. Definition 5) in Europe, many of them will have their own centres with computational and storage resources creating community-orientated research infrastructures (RIs). In order to establish interoperability between them and interoperability with the existing e-Science infrastructures (e.g. EGEE/EGI, DEISA/PRACE) similiar challenges will be faced.

In this context, the e-IRG presents in its blue paper 2010 that *'Europe's existing e-Infrastructure is also a research infrastructure. What perhaps sets it apart from other RI is its ability to deliver services across a broad spectrum of RI user communities. Close cooperation across RI and e-Infrastructure will drive evolution to their mutual benefit'* [210]. In-line with the blue paper statement, many concepts of the reference model are expected to be of use in the next decade when the 44 projects of ESFRI will start production runs establishing a wide variety of research infrastructures while they in turn will require similiar service entities.

The reference model is not just a model that can be applied to infrastructure interoperability setups, it also represents a platform for further innovative concepts. Basic Grid functionality (i.e. compute, data, information, security) as shown in Figure 5.2, is provided that can be the basis for more advanced Grid functionality (e.g. Grid workflows, SLA methods, collaborative Grid visualisation frameworks, etc.). The reference model describes basic services in-line with Definition 29 that can be re-used by a wide variety of so-called *'higher-level'* Grid services.

Figure 5.2 illustrates the idea of having a service-based reference model that offers infrastructure (and implied resource) services to end-users. The illustrated reference model entities are services following the SOA-based design as defined in Definition 29, having thus concrete service interfaces with clear semantics.

Scientific use cases are taken into account as a priority instead of industry-related case studies. Based on this and the given problem space illustrated in Figure 5.2, the reference model is designed to be suitable for e-Science environments as defined in Definition 30.

Requirement Definition	**Addressed in which manner**
Definition 28 (General Reference Model Design Principles)	IIRM is abstract; has entities with relationships; particular problem domain e-Science; technology-agnostic;
Definition 29 (Service-based Reference Model)	Service-based entities with interfaces and semantics;
Definition 30 (e-Science-Driven Reference Model)	IIRM designed for e-Science environments and applications;
Definition 31 (Grid Execution Management Entity)	Grid Execution Management Entity of the IIRM
Definition 32 (Grid Data Management Entity)	Grid Data Management Entity of the IIRM
Definition 33 (Grid Security Entity)	Grid Security Entity of the IIRM
Definition 34 (Grid Information Entity)	Grid Information Entity of the IIRM
Definition 35, Definition 36, Definition 37, Definition 38, Definition 39, Definition 40 (Relationships between Entities)	Interconnected but not layered relationships between entities;

Table 5.2: Addressed requirements on the reference model level.

Figure 5.2 illustrates the IIRM key entities as published in [265] and summarized as part of Table 5.2. A *Grid Execution Management Entity* as defined in Definition 31, including execution environments is provided. Also, a *Grid Data Management Entity* according to Definition 32, including data transfer functionality is provided. Furthermore, a *Grid Security Entity* as defined in Definition 33 is illustrated essentially meaning authentication and attribute-based authorisation methods. Finally, the IIRM also provides a *Grid Information Entity* as defined in Definition 34, including resource usage tracking.

Figure 5.2 illustrates relationships between the aforementioned entities with 'black interconnections'. But the relationships are not following a layered approach being still inter-connected as illustrated in Figure 4.2 in Section 4.1.2. Definitions 35 and 36 mandate that the reference model exposes information about computational and data entities.

The model also satisfies Definition 37 in the sense that the computational entities are enabled with authentication and attribute-based authorisation mechanisms, while the same is being true for data entities as defined in Definition 38. Definition 39 raises the demand for also having information available about security services and setups. The reference model exposes information from security entities (e.g. attribute authority [303]) and describes security information as part of the computational and data service entities. Clients are thus also to find out which security setup a particular computational, data or even security entity is using. Definition 40 mandates that the computational and data entities are interconnected in the sense that computational entities mostly require data-staging [115] methods.

Finally, Table 5.2 summarizes the aforementioned reference model elements and provides an overview how the requirements of Chapter 4 are addressed on the reference model level.

5.1.2 Associated Reference Architecture General Design

The previous section provides us with the abstract IIRM reference model design, but the design is too abstract and thus not detailed enough for implementation. It only allows for comparison between reference models while its associated elements provide much more detail for the implementation of this model. But its design decisions guides the more detailed architectural work leading to a more concrete Web service-based IIRM reference architecture as defined in Definition 41. Important associated architecture work considering specific protocols, profiles, specifications, and standards in-line with Definition 42 are part of this section and shown in Figure 5.3 (marked in red). The requirements addressed on the architecture level are summarized as part of Table 5.4.

Guided by the abstract IIRM design its entities are mapped to specific core building blocks of the reference architecture. Inspired by lessons learned around the ISO/OSI and the more successful TCP/IP model reviewed in Section 3.1.1, the design decisions lead to a compact architecture compared to OGSA. One of the reasons why ISO/OSI was not successful was that the reference model was not usable as planned and, what is interesting in the context of this thesis, *'no thought was given to internetworking'* [295]. Tanenbaum concludes this topic with *'things did not turn out that way'* [295] essentially meaning the difference between theory and practice. Section 3.1.1 already revealed that TCP/IP is much more successful and that can at be at least partly explained by the fact that the protocols came first [295]. The model was really *'just a description of the existing protocols'* [295] and its implementations perfectly fit to the model, while it basically does not fit any other protocol stack.

The aforementioned insights lead to two major approaches that is a survey of existing standard protocols in practice as well as internetworking (i.e. relationships) in real production e-Science infrastructures. While the former is handled in this particular section, the relationships between the different standards are handled in the next section of this chapter in more de-

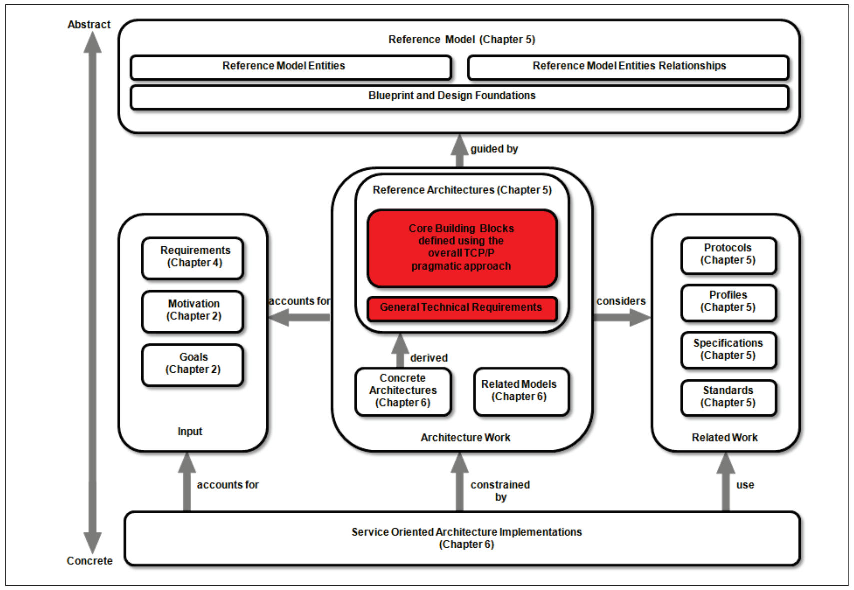


Figure 5.3: Reference architecture with protocols, profiles, specifications and standards as building blocks.

tails by exploring various different concepts where standards are closely interlinked with each other. The success of TCP/IP outweighs the drawbacks of the aforementioned approaches described in [295], and as a consequence the reference architecture is truly compact as required in Definition 44.

Inspired by the success of TCP/IP that is based on *'emerging standards'* that have been already used in practice before the reference model definition, we followed the same approach within the OGF, thus driving the activities of the GIN community group. A GIN research study culminated in a publication [256] that clearly identifies those standards that are used in practice in a wide variety of production e-Science infrastructures according to Definition 5. Based on this study, the core building blocks of the reference architecture are based on open standards as foundational approach as defined in Definition 47. The standards are either well established in production Grids or are planned to be adopted in the near future, while all provide normative specifications as required by Definition 46. Table 5.3 lists these standards and their SDOs, including mappings to the corresponding abstract entity of the IIRM design defined in the previous section (cf. Table 5.2).

Some listed standards in Table 5.3 use inherently other standards, especially when using security standards such as (g) SAML and (h) X.509. These standards and a few others are not part of the initial results (i.e. those below the triple line) since at the time of the academic analysis they were either considered as fundamental or too new. For example, SAML takes advantage of (m) WS-Security [131] that stands for several specifications used to transport XML-based SAML assertions during SOAP-based WS message exchanges.

In addition, since (o) HTTP(S) is used to transfer SOAP-based message exchanges it was overlooked that some middleware systems actually also use it for 'normal data transfers' (e.g.

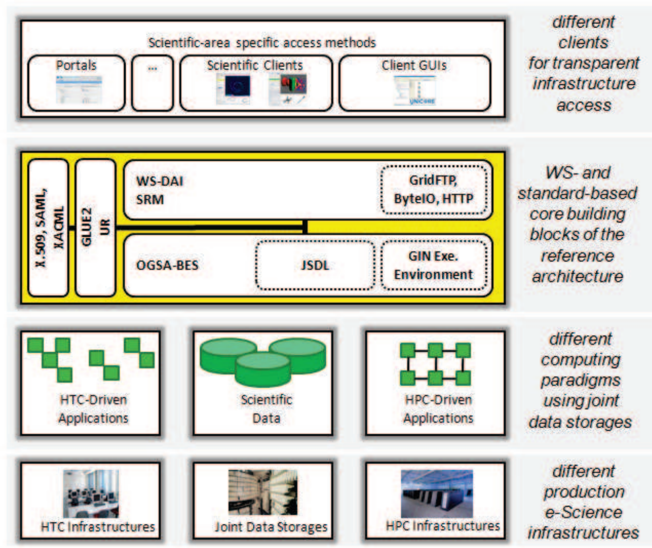


Figure 5.4: The reference architecture of the IIRM model with standards as core building blocks.

UNICORE) with quite good performance. POSIX-based access was used in several production Grids that later become the (n) ByteIO standard specification [230] being already adopted (e.g. in UNICORE and GENESIS).

Experimental work on the aforementioned standards have been performed in the GIN activities in OGF via interoperability tests between Grid middleware (cf. Definition 12) leading to application demonstrations in many events (e.g. Supercomputing conference series 2007 -

Open Standard	**IIRM Entity Mapping**	**SDO with Specification status**
(a) OGSA-BES	Grid Execution Management Entity	OGF Proposed Recommendation
(b) JSDL	Grid Execution Management Entity	OGF Recommendation
(c) DRMAA	Grid Execution Management Entity	OGF Recommendation
(d) UR	Grid Information Entity	OGF Recommendation
(e) GLUE2	Grid Information Entity	OGF Proposed Recommendation
(f) XACML	Grid Security Entity	OASIS Specification
(g) SAML	Grid Security Entity	OASIS Specification
(h) X.509	Grid Security Entity	IETF Standard
(j) GridFTP	Grid Data Management Entity	OGF Recommendation
(k) SRM	Grid Data Management Entity	OGF Recommendation
(l) WS-DAIS	Grid Data Management Entity	OGF Proposed Recommendation
(m) WS-Security	Grid Security Entity	OASIS Specification
(m) ByteIO	Grid Data Management Entity	OGF Proposed Recommendation
(o) HTTP(S)	Grid Data Management Entity	W3C Standard

Table 5.3: Common open standards as potential core building blocks of the IIRM.

2010 [94, 95, 96, 97]). Therefore, the initially not listed standards of the model are added, but below the triple line in Table 5.3.

A collaboration of GIN with the European Telecommunications Standards Institute (ETSI) [273] standardisation organisation lead to further insights about the list of these specifications. The GIN interoperability studies influenced ETSI as described in a table about Grid middleware standard adoptions in Rings et al. [273]. Apart from Table 5.3, an overview of the core building blocks of the IIRM reference architecture is provided by Figure 5.4.

The GIN Execution environment (e.g. environment variables, etc.) [38] is in the process of being standardised as part of the PGI profiles and is largely driven by GLUE2 elements as subsequent sections will reveal. A more thorough comparison between Table 5.3 and the IIRM reference architecture overview as provided in Figure 5.4 reveals its missing standard, DRMAA. The DRMAA interface in the middleware UNICORE implementation [262] was evaluated with existing RMS systems (cf. Definition 11) in HPC environments. But although the approach is promising, only a limited number of vendors have adopted it (e.g. Sun Grid Engine [183]). Other major RMS systems do not provide any adoptions of DRMAA, especially those relevant for HPC systems (e.g. Load Leveler [205]). The adoption rate of DRMAA is thus not broad enough to be part of the reference architecture.

Another missing emerging standard is OGSA-RUS [65] that can be explained by two reasons. First, although early draft implementations exist, for instance as shown in [177], OGF did not yet delivered a full specification as required as part of Definition 46. Second, the analysis of the implementation [177] revealed fundamental scaling problems in exposing a high number of URs via the usual Web service-based OGSA-RUS interface. Instead, it makes sense to adopt the URs as Grid information elements like the GLUE2 schema (cf. Table 5.3) despite its different focus (and more security constraints). Although being a traditional part of accounting, it basically exposes usage information and as such can be exposed via an information service that is traditionally scalable (e.g. LDAP-based BDII systems [212]). The OGF WS-Agreement standard [114] is related to production Grids for dynamically establishing and automatically negotiating SLAs today. But production Grids as defined in Definition 7 have not yet widely deployed adoptions of this particular concept and thus it it not relevant in this thesis that is in-line with Definition 44.

Finally, Table 5.4 summarizes the aforementioned general reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	**Addressed in which manner**
Definition 41 (WS-based Reference Architecture)	Services implemented with Web services;
Definition 42 (Concrete Specifications for a Reference Architecture)	Referenced standard specifications;
Definition 44 (Slim Reference Architecture)	Compact design using TCP/IP approach (existing protocols);
Definition 46 (Normative Specifications for a Reference Architecture)	Architecture is based on normative specifications;
Definition 47 (Open Standards-based Reference Architecture)	Architecture core building blocks are standards;

Table 5.4: Addressed general requirements on the reference architecture level.

5.1.3 Detailed Associated Reference Architecture Core Building Blocks

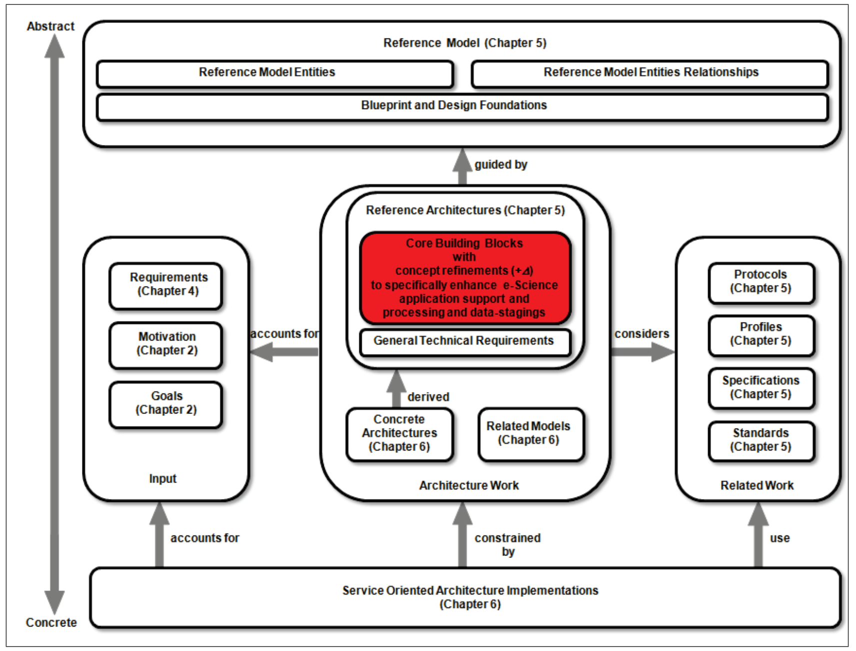


Figure 5.5: Reference architecture core building blocks with several concept refinements.

The last sections provide insights into how reference model entities guide the reference architecture design and its core building blocks. The abstract core IIRM entities guide the specific IIRM architectural design on another abstraction level as illustrated in Figure 5.5.

This section gives more details of associated architecture elements (marked in red in Figure 5.5) providing more information about the core building blocks and how exactly they address the requirements raised in Chapter 4. This includes the levels of which the core building blocks are relevant in a typical Grid architecture and how they should be implemented within Grid middleware systems as defined in Definition 12. The requirements addressed on the architecture level are summarized as part of Table 5.6.

This section also reveals where the academic analysis as well as the lessons learned from practical field studies surveyed in Chapter 3 influence the reference architecture with several identified refinement concepts. These concept refinements are specifically focused on the improvement of e-Science application support as well as job processing and data-staging functionality. A conceptual view of the reference architecture is presented in Figure 5.6 illustrating where the core building blocks and their refinements are adopted.

Figure 5.6 reveals an important outcome of the lessons learned using many of the core building blocks already in practice, that in turn revealed a certain amount of concept refinements in the sense of Definition 48. Although the core building blocks follow Definition 46, several refinements bear the potential to increase the effectiveness and efficiency in production

Core building block	Description
JSDL + ΔU	Refinement concept for job description
OGSA-BES + ΔV	Refinement concept for job management
GLUE2 + ΔW	Refinement concept for the information model
WS-DAIS + ΔX	Refinement concept for the database access
SRM + ΔY	Refinement concept for the storage management
UR + ΔZ	Refinement concept for the resource usage tracking

Table 5.5: Refinements list of the IIRM core building blocks.

setups with corresponding standards that are used as core building blocks or improve specifically interoperability. These refinements are an evolution of open standards, such as the move from the IPv4 to the IPv6 in the TCP/IP model that has gained considerable profile over the last couple of years. As shown in Figure 5.6, the refinements are marked with Δ in the context of the core building blocks referring to additions to the corresponding normative specifications. A summary list of these refinements is shown in Table 5.5.

The comparison of Table 5.5 with the overall reference architecture core building blocks listed in Table 5.3 shows that not all core building blocks of the reference architecture need refinements.

Figure 5.6 illustrates the core building blocks of the reference architecture in the given context of production e-Science infrastructures (cf. Definition 5), including their infrastructure resources (cf. Definition 6). The conceptual view reveals the non-empty list of core services

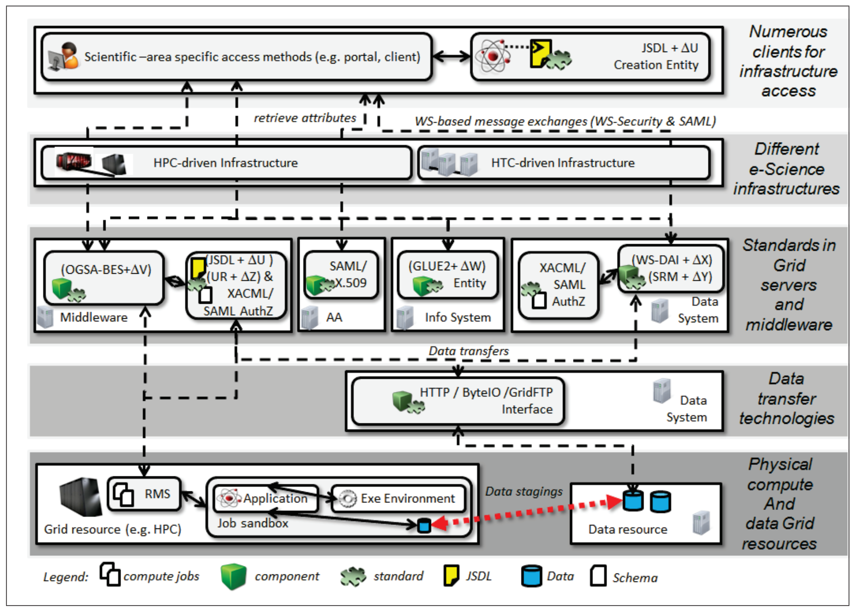


Figure 5.6: Conceptual view of the reference architecture with refined open standards.

that must be in place to form an infrastructure as defined in Definition 45. The services that are part of the core four key entities of the reference model are described in the following paragraphs, by explaining how they address the functional requirements raised in Section 4.2. All those standards that are considered to be core building blocks have been part of the studies in the context of GIN since many years [256].

The *Grid Execution Management Service* as required by Definition 49 is represented by the OGSA-BES version 1.0 specification [169]. It is able to satisfy the demands for scientific use cases, and several refinements of the specifications are proposed in later sections in this chapter. The OGSA-BES specification relies on the JSDL specification [115] that satisfy another requirement part in Definition 49. This includes its various extensions such as the SPMD [281], the HPC application extension [197], or the parameter sweep job extension [155] that are in-line with the basic JSDL framework. These core building blocks are related to another core building block that is defined as part of Definition 49 and that is currently in the process of being standardized in PGI based on a draft GIN specification [38].

In terms of access and management of storage resources as defined in Definition 50, the reference architecture is making use of the SRM [286] and WS-DAIS [118] standards. The latter is considered to access relational databases and the former makes use of data transfer mechanisms defined as part of Definition 50. The GridFTP [109] specification, HTTPS [43], and the ByteIO specification [230] are also considered as core building blocks as shown in Figure 5.6. These data transfer specifications not only play a role in storage management activities, but also for data-staging activities as part of the execution of Grid jobs.

Figure 5.6 also illustrates the existence of an information system that uses the GLUE2 specification [113] and its different renderings (e.g. LDAP, XML, etc.) as another core building block of the architecture. It addresses the requirement defined in Definition 55 contributing to the common information ecosystem throughout the architecture. Key IIRM design elements rely significantly on this information ecosystem, as subsequent sections in this chapter will reveal. Apart from the information about capabilities of services and the properties of their resources, resource usage tracking is also an important aspect. The UR specification [216] is used addressing the requirement defined in Definition 56. As Figure 5.6 reveals, both GLUE2 and UR can be refined that is described in subsequent sections in this chapter.

Figure 5.6 also reveals security aspects, and as a crucial and important topic for interoperability, subsequent sections will go into much more detail. In this section, the focus is only on aspects describing how the architecture satisfies requirements raised in Section 4.2. X.509 based SSL connections [153] for authentication as defined in Definition 51 are used as authentication mechanism for each Grid service (cf. Definition 13) in the architecture. While the majority of systems provide authentication directly at the functionality-specific Grid services, some mid-

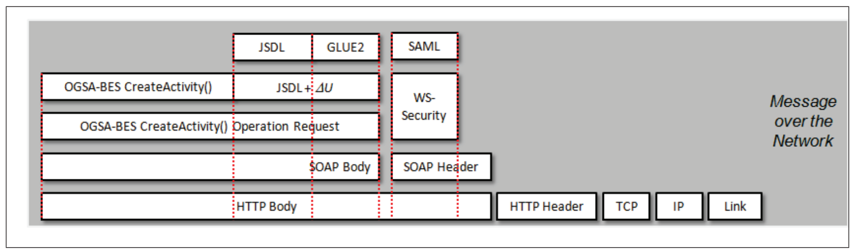


Figure 5.7: Communication baseline mechanism that stands for a couple of interlinked standard protocols.

dleware systems also offer *'forwarding systems'* for authentication like UNICORE with the UNICORE Gateway [293]. The latter ones are for simplicity not illustrated, but are automatically part of the corresponding core building block service (e.g. OGSA-BES, SRM, etc.). Another key security feature is the support for attribute-based authorisation that is evident on many levels within Figure 5.6. The existence of an Attribute Authority (AA) service that can be used with the SAML protocol [142] in order to retrieve security attributes as required by Definition 52. Also, WS-Security [131] is used in order to transport SAML tokens [142] as defined in Definition 54. Each of the reference architecture services provides an attribute-based authorisation functionality, as defined in Definition 53, that works with SAML using well-defined XACML policies [234].

Another element of the reference architecture is the *'baseline communication mechanism'* that addresses the requirement raised in Definition 41. The communication between the core building blocks uses WS message exchanges [296] based on the known SOAP protocol [190] and is thus standard-based (cf. Definition 47). Grid technologies mostly adopt WS-based specifications and *WS-* standards* stands for the wide variety of WS-based standard specifications used today. Because there are so many specifications, a reference model approach is needed to reach common adoption enabling interoperable e-Science infrastructures (cf. Definition 21).

These baseline communication is omitted from Figure 5.6 to preserve the focus on the core building blocks. These underlying protocols are not considered to be core building blocks of the reference architecture and are collectively described here in this section as the baseline communication mechanism instead. Apart from SOAP and HTTPS, the use of WS-Addressing [137]

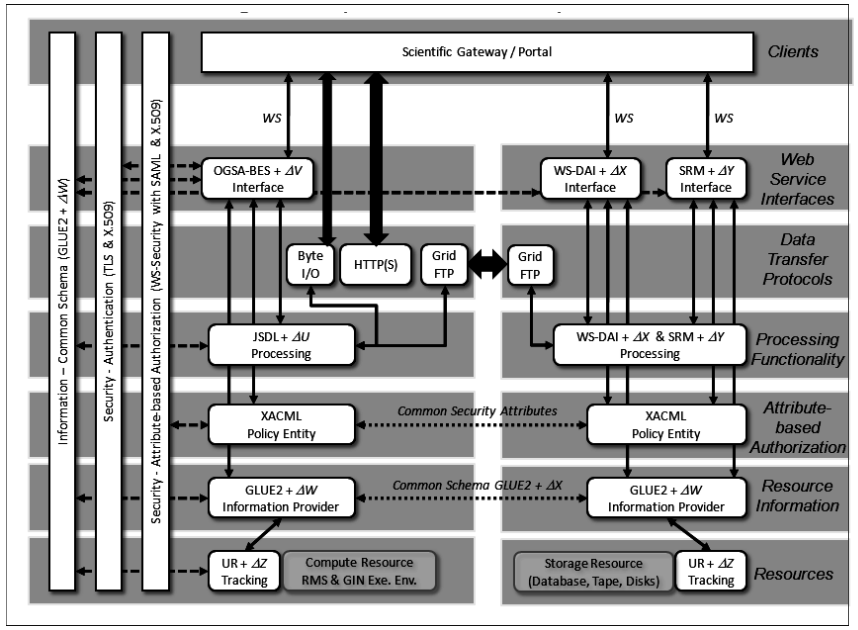


Figure 5.8: Non-layered reference architecture in context of a compute and storage resource.

is another part of the baseline mechanism enabling the standard-based addressing of specific Web services. X.509 is used to underline its importance for end-user certificates that are used to enable SSL connections [153] with the HTTP(S) protocol used for SOAP-based WS message exchanges as illustrated in Figure 5.7. It illustrates the baseline communication mechanisms relates to the different types of reference architecture core building blocks (e.g. OGSA-BES and JSDL) and their refinements (e.g. ΔU = improvements of JSDL elements or additional elements). It also illustrates how SAML is used via WS-Security in the SOAP header.

Figure 5.7 illustrates one example of using this baseline communication mechanism with core building blocks (and their refinements) that can be used with WS-based specifications. Like OGSA-BES, several core building blocks are specifications that offer service interfaces via portTypes [296] and WS operations [296] described via a WSDL [150] per Grid service (cf. Definition 13). In the reference architecture, those are OGSA-BES for computational functionality, and SRM as well as WS-DAIS, that both offer storage and data related functionality. In contrast, other types of core building blocks are XML-based schemas such as JSDL, UR, or GLUE2. As shown in Figure 5.7, JSDL is used as part of the OGSA-BES specification and as such it is part of the WS-based message exchanges between OGSA-BES clients and servers. This includes its extensions that are defined in various profiles (e.g. SPMD [281]) that can be optionally used without breaking the core building block.

As indicated in Figure 5.7, JSDL and GLUE2 is used together as the job description as part of the OGSA-BES specification to increase the efficiency of e-Science applications (cf. Definition 9) executions as later parts of this chapter reveal. Information systems such as the CIS [223] expose GLUE2 as part of information queries. Although SAML also offers protocols [142] that are adopted by AA services, SAML assertions [142] are just XML-based schemas that are shipped in SOAP headers [190] using WS-Security methods as described in more detail in [302] and illustrated in Figure 5.7.

The following core building block is used in the lower areas of Grid middleware and storage technologies that is traditionally in the area of accounting but here rather considered as information. For tracking resource usage, the UR schema [216] is often adopted, but it is not transported via WS message exchanges such as JSDL as part of OGSA-BES. In terms of transport, we refer to accounting and billing systems such as the Distributed Grid Accounting System (DGAS) [245] or the Swedish Grid Accounting System (SGAS) [279] that are not in the scope of this reference model.

XACML is adopted in Grid technologies for policy-based authorisation decisions and is as such not directly affected by WS-based messages as well. But the use of XACML is optionally possible as protocol for XACML-compliant centralized authorisation systems (e.g. Argus [303]) in order to request remote authorization decisions in addition to local security policies. Figure 5.8 presents an overview of the overall reference architecture initially published in [272] illustrating core building blocks (with refinements Δ) and their interrelationships. It also illustrates the so-called *'plumbing'* concepts part of the associated security pattern that is revealed later in this thesis.

The reference architecture presented in Figure 5.8 addresses several non-functional requirements (cf. Section 4.3). Although this will be described in more detail in Chapter 6 based on specific concrete architecture adoptions, several insights are presented in this section. Assuming that multiple middleware adoptions will exist, the design is in-line with Definition 74 since we avoid decreased performance when using multiple infrastructures by avoiding the need of transformation logic (cf. Definition 27). The refinements of core building blocks that follow in subsequent sections are aimed to lay the foundation for Definition 75. It can be supported in the sense of Definition 76 since major middleware provider adopt the reference architecture concepts as part of their architecture already as Chapter 6 reveals.

Finally, Table 5.6 summarizes the aforementioned detailed reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level. It also shows that the requirements are mapped to either one (e.g. execution management) or several standards (e.g. data transfer protocols) and as such can be considered as different areas where existing standards are applicable. This is part of the idea of following the TCP/IP concept mentioned in Chapter 3 and 4 with a bottom-up approach of focussing on those existing standards that proved to be useful in production. Tanenbaum describes this approach as the protocols came first (like in the IIRM case), and *'the model was essentially just a description of the existing protocols'* [295]. Table 5.6 shows nicely that the IIRM does not invent new protocols, but instead puts existing protocols (and schemas) together to a model following the TCP/IP compact design.

Requirement Definition	**Addressed in which manner**
Definition 41 (Web Services-based Reference Architecture)	Baseline communication with HTTP(S), SOAP, WS-Addressing
Definition 45 (Core Reference Architecture Elements)	Core building blocks represent core reference architecture
Definition 46 (Normative Specifications for a Reference Architecture)	Core Building blocks are normative specifications
Definition 47 (Open Standards-based Reference Architecture)	Core building blocks are open standards
Definition 48 (Reference Architecture Standard Refinements)	Architecture core building blocks with Δ;
Definition 49 (Grid Execution Management Service)	Core building blocks OGSA-BES + ΔV with implied JSDL + ΔU, GIN exe. env.
Definition 50 (Grid Data Management Service)	Core building blocks WS-DAIS + ΔX, SRM + ΔY with implied GridFTP, HTTPS, and ByteIO
Definition 51 (Grid Authentication Service Functionality)	Core building block PKI-based SSL
Definition 52 (Grid Attribute Authority Service)	Core building block SAML assertions and protocol
Definition 53 (Grid Attribute-based Authorisation Functionality)	Core building block XACML
Definition 54 (Grid Security Attributes Transport)	Core building block WS-Security and SAML assertions
Definition 55 (Grid Information Model Schema)	Core building block GLUE2 + ΔW
Definition 56 (Grid Usage Record Format Schema)	Core building block UR + ΔZ
Definition 74 (Multiple Infrastructure Usage Performance)	Reference architecture design does not use transformation logic
Definition 75 (Infrastructure Resource Usage Efficiency)	Core building blocks refinements for optimized resource usage
Definition 76 (Supportability)	Core building blocks partly adopted in official technology releases

Table 5.6: Addressed detailed requirements on the reference architecture level.

5.1.4 Associated Reference Architecture Infrastructure Integration Constraints

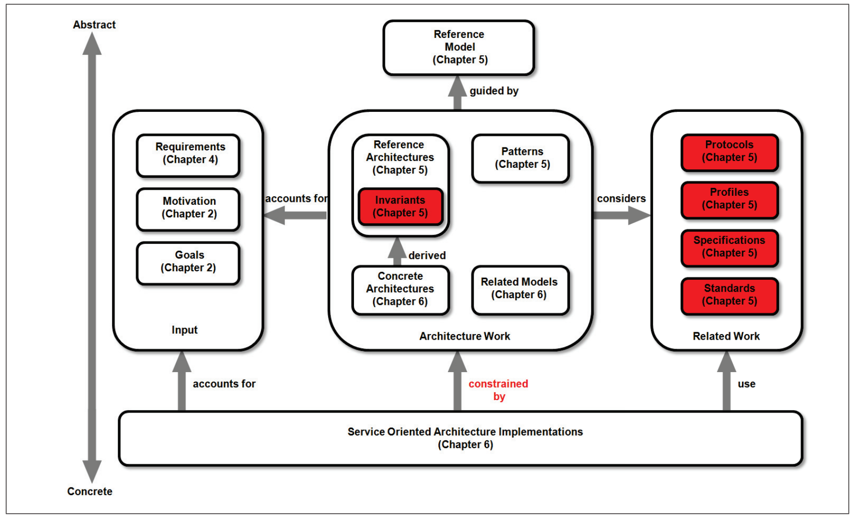


Figure 5.9: SOA-based implementations are contrained by invariants of the architecture work.

A set of invariants that have been published earlier [260] and that complement the reference model and its associated reference architecture with respect to infrastructure integration issues is presented in this section. These issues concern several operational constraints in the area of information exchange, accounting, and attribute-based authorisation as required in Definition 43. Adoptions that satisfy these invariants significantly increase interoperability between production e-Science infrastructures (cf. Definition 5). The invariants in the context of the broader reference model guideline are illustrated in Figure 5.9. The requirements addressed on the architecture level are summarized as part of Table 5.7. In [260], the first infrastructure integration element of the reference architecture is described as the *'Global Information Invariant (GII)'*. It defines an overall information ecosystem across infrastructures. The aim of this invariant is to address the requirement regarding constraints defined in Definition 69, contributing to an overall information ecosystem within concrete infrastructure deployments. The fundamental information exchange occurs when working with multiple infrastructures in terms of a specific invariant in the context of the reference architecture core building blocks. The intent is not to provide an in-depth view of the component information mechanism, but instead to give guidance for concrete architecture implementations that adopt the reference architecture. The information exchanges itself are the focus and not descriptions of low-level protocol aspects. The GII defines on the highest-level the abstract *'service and resource information property'* to be preserved by all the Grid services (cf. Definition 2.4) in the architecture. It is defined as follows:

Definition 86 (Global Information Invariant) *The global information invariant mandates the exposure of information with the common information schema GLUE2 for all reference model entities and their associated reference architecture core building blocks. At any given time, the information within this schema must be accessible via an information service that in turn is the crucial element of the information exchange policy.*

Definition 86 ensures that all parts of an interoperable network of IIRM services are constrained by the common information exchange policy that mandates the use of the common information schema GLUE2 [113], which is one of the core building blocks of the reference architecture. The aim of this invariant is to enable adopters to make certain assumptions in that information about deployed production e-Science infrastructure services at any given time. It ensures that the same information model is used in order to prevent transformation logic (cf. Definition 27) that need to transform one information model to the other which in turn often leads to semantic loss. Figure 5.10 illustrates the information exchange policy that mandates the use of GLUE2 exposed via an information service for each of the IIRM services. The red line in Figure 5.10 indicates that GLUE2 information must be seen by end-users via some dedicated user interface. Any relevant production e-Science infrastructure deployment needs to satisfy this invariant in order to be compliant with the reference architecture design to achieve *'basic semantic interoperability'* with other infrastructures.

Although such an invariant in general and the use of a common information schema like GLUE2 sounds trivial, this approach was not followed in the European production e-Science infrastructures EGEE/EGI, NorduGrid, and DEISA/PRACE. In [163], Field et al. describes the use of a NorduGrid schema for the NorduGrid infrastructure while the EGEE/EGI infrastructure is largely based on the GLUE1.3 proprietary information schema. DEISA/PRACE used UNICORE services [223] that describe resources via the so-called Common Information Model (CIM) [5] of the Distributed Management Task Force (DMTF). The findings of these studies

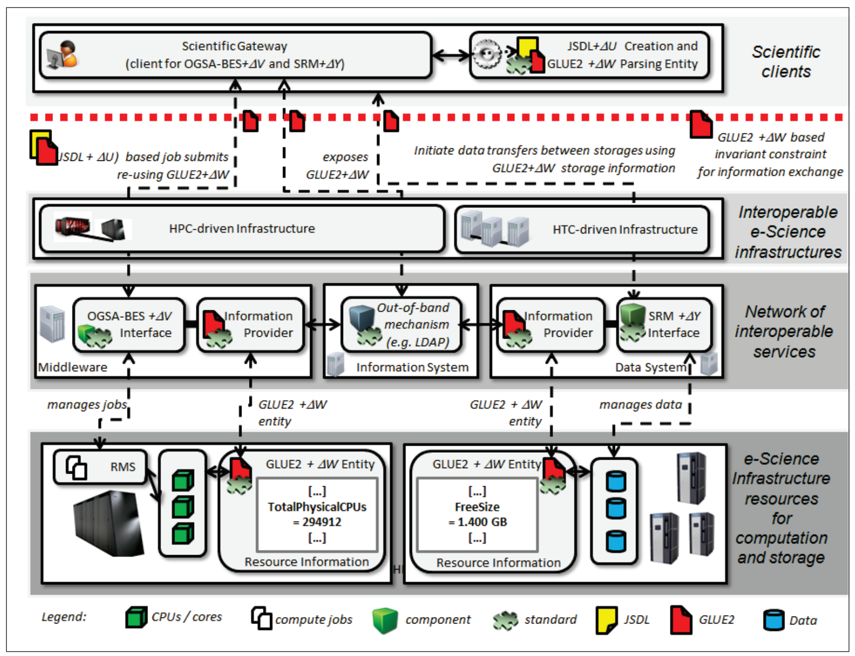


Figure 5.10: The Global Information Invariant ensures information exchange without semantic loss.

reveal the need for a commonly accepted information schema that is applicable by the wider community. The GII is defined in order to enhance interoperability between production e-Science infrastructures.

As shown in Figure 5.10, the constraint is set so that reference architecture services that enable access to infrastructure resources (cf. Definition 6) such as computational resources (e.g. OGSA-BES) expose GLUE2-compliant information. In [224], methods for using OGSA-BES with GLUE2 are presented in order to expose standard-compliant information about computational resources (i.e. number of cores, etc.) instead of exposing a limited OGSA-BES information schema [169]. The exposed GLUE2 elements of these services are re-used for the job submission as part of the JSDL job description schema. Resource requirements of the JSDL document [115] can be matched with the actual resource information published by the corresponding services, that in turn avoids semantic mismatches. Details about this approach are published in [261], while this section provides insights about its higher-level operational impacts. Figure 5.10 illustrates that the invariant also enforces this constraint of exposing GLUE2 on services that manage data storage resources (i.e. SRM). Definition 86, refers to an information system (e.g. LDAP [196]) that exposes GLUE2-based information about resources collected from information providers.

The second infrastructure integration element of the reference architecture in [260] addresses the requirements in the area of accounting. The previous paragraphs focussed on an invariant about the common information exchange. In constrast, the next paragraphs focus on resource accounting as a particularly important *'type of information'* that is relevant to production e-Science infrastructures. Resource accounting tracks the use of several services and, most notably, their underlying resources to which the services provide access to. But the handling of accounting information is different to the handling of the general service and resource information introduced in the last paragraphs.

The information encoded in GLUE2 being exposed via information services is available to all infrastructure users such as e-Scientists (cf. Definition 10). In contrast, accounting data often raises privacy issues that limit their open exposure via generally accessible information services. The information encoded in GLUE2 is mostly static with only *'a few'* changes in their dynamic parts (e.g. TotalJobs in ComputingService Entity [113]) compared to resource accounting information that is highly dynamic for infrastructures resources. Especially for modern large-scale HPC systems the services serve a high amount of users at the same time and the amount of accounting information about used resources by end-users can be much higher than in infrastructures traditionally driven by HTC. This issue becomes even more relevant in interoperable e-Science infrastructure setups with a plethora of available services.

The fundamental accounting information exchange adds another constraint to the IIRM and its architectural design. The invariant for these infrastructure operations is the *Global Accounting Invariant (GAI)* that addresses the requirements raised as part of Definition 70. It defines at the highest-level the abstract *'resource accounting property'* that must be preserved by IIRM services in particular and in the whole interoperable e-Science infrastructure ecosystem in general. It is defined as follows:

Definition 87 (Global Accounting Invariant) *The global accounting invariant mandates resource tracking with the common resource usage schema UR for all reference model entities and their associated reference architecture core building blocks. At any given time, any resource usage for computing and data management entities must be trackable via the common resource usage schema that in turn is a crucial element of the common accounting exchange policy.*

As shown in Figure 5.11, the accounting exchange policy enforces the use of URs [216] as the common resource usage tracking schema, which is also one of the core building blocks of the reference architecture. This UR-based constraint ensures that all services of the IIRM and its associated reference architecture can exchange accounting information without any semantic loss or the need for transformation logic.

Any relevant production e-Science infrastructure that adopt IIRM services need to satisfy this invariant in order to be compliant with the reference architecture design. This enables the *'basic accounting interoperability'* with other infrastructures that also adopt the IIRM. The reason is to overcome limitations faced when tracking resource usage across infrastructure boundaries as often observed during interoperability studies in the GIN studies [256]. The policy as defined in Definition 87 set the constraint for the computing services (i.e. OGSA-BES), but also for the storage services (i.e. SRM). In contrast to computing services, storage services so far did not track resource usage in a common format in production Grids while at the same time being equally important as compute. This can be partly explained by the fact that the UR specification [216] only supports compute resource tracking and not storage and thus this must be changed in order to satisfy the defined invariant. This is indicated in Figure 5.11 with the UR + ΔZ element where ΔZ indicates storage resource refinements according to Definition 48 some of which will be revealed later in this chapter.

This is essential to enable a consistent *'basic accounting interoperability'* between production e-Science infrastructures. Accounting systems out of thesis scope (e.g. SGAS [279]) work with UR-based information collected from accounting providers that are, from the design perspective, implemented close to the different IIRM services. The transfer mechanism can be realised

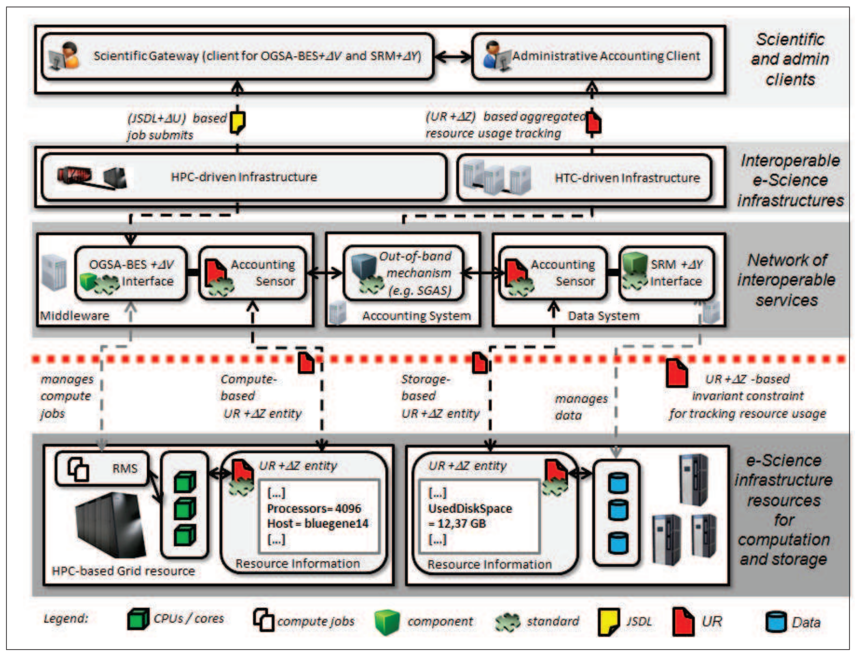


Figure 5.11: The Global Accounting Invariant ensures accounting information exchange without semantic loss.

using messaging implementations (e.g. ActiveMQ [289]), because earlier investigations [177] revealed that a WS-based OGSA-RUS [65] is not scalable enough when HPC-driven resources are used. There is also a demand to extend the UR schema with aggregated resource usage tracking concepts as shown in Figure 5.11. In that case, an administrative accounting client takes advantage of the convenient use of aggregated URs. This and other refinements are indicated with ΔZ.

The third infrastructure integration element of the reference architecture is the *'Global Authorisation Attributes Invariant (GAAI)'* that defines elements of the overall security ecosystem across infrastructures. The next paragraphs describe the need for a common set of security attributes to enable attribute-based authorization during secure production operations across infrastructures as defined in Definition 71. These security attributes convey pieces of information about project, group, or VO [172] memberships as well as role possession. The GAAI defines at the highest-level the abstract *'service authorization property'* that must be preserved by the IIRM services. It is defined as follows:

Definition 88 (Global Authorisation Attributes Invariant) *The global authorisation attributes invariant (GAAI) mandates the use of common security attributes for each end-user in the security enforcement activities for all reference model entities and their associated reference architecture core building blocks. At any given time, the identity of an end-user must have an aligned set of common security attributes that are released from a trusted attribute authority representing a crucial element of the common security attribute exchange policy.*

As shown in Figure 5.12, the common security exchange policy enforces the use of common security attributes that are exchanged using different approaches. While one approach uses the attributes encoded in SAML tokens [142], another approach ships the security attributes as part of X.509 proxy extensions (aka attribute certificates [303]). While both need to be supported in production e-Science infrastructures, SAML is established as an alternative method of choice in parallel to attribute certificates. Acknowledging the slow infrastructure migration times mentioned in Definition 68 both have to be supported. But using different approaches for transporting is not directly a problem when the attributes are still compliant to the invariant by using the common set of attributes for authorisation decisions.

Definition 88 is crucial to enable compliance with the reference architecture design to achieve the *'basic authorisation interoperability'* with other infrastructures. This is possible since the same attributes, even if differently encoded, state the same security information while their transfer method is another aspect that is not as important as the common security attributes for end-users, thus following the same semantics.

This is best explained by describing the other essential parts of attribute-based authorisation that is about the corresponding AA and security policies as illustrated in Figure 5.12. The security policy mandates that there is one central AA like the VOMS [303] system responsible for releasing signed attributes about end-users in two ways. This system achieved this with SAML assertions and also with attribute certificates as part of X.509 proxy extensions. The AA is one central source of trust and thus it is important that it satisfies the defined GAAI. While the manner in which ways the attribute get from clients to services is less important, it is more important that the sink also conforms to our invariant as the source. As the security attributes are being signed they cannot easily compromised or used by other end-users, but it is essential that the sink that enforces the authorisation trusts the source (i.e. the AA).

Figure 5.12 illustrates the aforementioned sink where authorisation policies are encoded using Gridmap files [167] or XACML [234] policies. In order to achieve *'basic authorisation interoperability'*, the constraint is set in which the sink, like the source, needs to satisfy the invariant. This raises the demand for such authorisation policies to include authorisation definitions for common security attributes. During job submissions or data transfer between storage, Definition 88 includes the constraint that the IIRM services (i.e. OGSA-BES and SRM) have to use such a sink in order to enforce attribute-based authorisation.

Security is often complex, but it is essential to achieve a secure operational level of interoperability between production e-Science infrastructures, including the finely-granular attribute-based authorisation capabilities. Some infrastructures, like EGEE/EGI for example, have used their own attribute profile based on so-called Fully Qualified Attribute Names (FQANs) [108]. Other infrastructures such as DEISA/PRACE did not perform any kind of attribute-based authorisation (e.g. through UNICORE), because the security was enforced on a level of existing unix accounts at resources. There was not even basic authorisation interoperability between European e-Science infrastructures that led to the fact that e-Scientists that would like to use both were required to obtain access to them in a different way.

More recently, attribute-based authorisation is supported by UNICORE [302] and used in DEISA/PRACE with some applications across infrastructures (e.g. EUFORIA [225]). But to achieve seamless operational interoperability without dedicated interoperability setups, those setups need to satisfy the global authorisation attributes invariant defined above. This enables the *'basic authorisation interoperability'* between middleware and storage technologies that adopt the IIRM and its associated reference architecture.

All the aforementioned constraints address the overall reference architecture requirement defined in Definition 43. Which information and security data needs to be supported is known

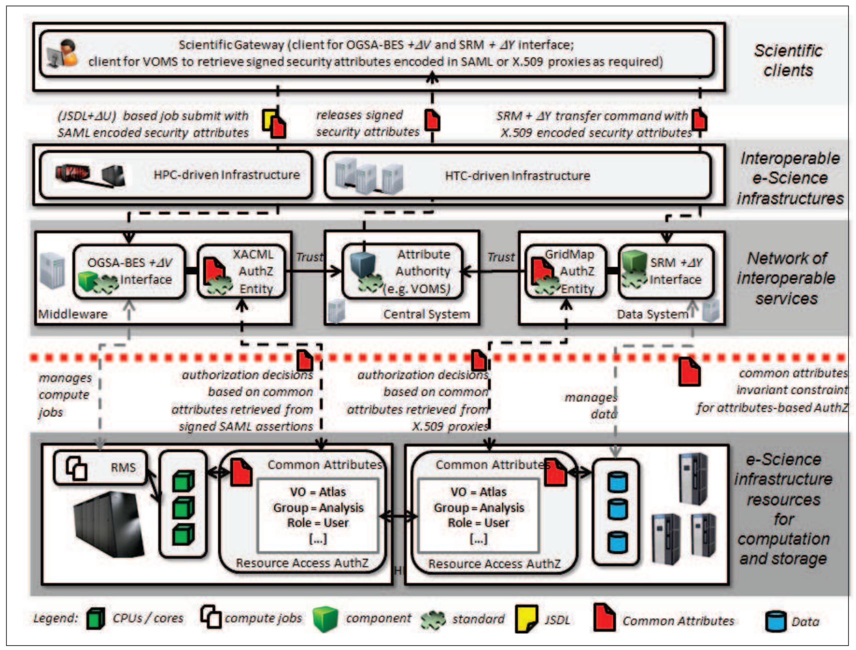


Figure 5.12: The Global Authorisation Attributes Invariant ensures the use of common security attributes.

in the reference architecture (e.g. the need for common security attributes about end-users). Locations within the reference architecture are known where one can assume that specific information exists (e.g. common usage records at the middleware level).

Figure 5.9 illustrates that concrete architectures with SOA-based implementations of our reference architecture are constrained by the whole architecture work (i.e. core building blocks, etc.) in general and the defined constraints in particular. Finally, Table 5.7 summarizes the aforementioned invariants and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	Addressed in which manner
Definition 43 (Information and Security Constraints for a Reference Architecture)	Three constraints for security and information;
Definition 68 (e-Science Production Technology Adoption Constraint)	SAML in parallel to legacy attribute transers;
Definition 69 (e-Science Infrastructure Information Ecosystem)	GII invariant for information ecosystem;
Definition 70 (e-Science Infrastructure Resource Tracking Ecosystem)	GAI invariant for resource tracking;
Definition 71 (e-Science Infrastructure Attribute-based Authorisation Ecosystem)	GAAI invariant for authorization;

Table 5.7: Addressed detailed requirements of the infrastructure integration constraints.

5.1.5 Overall Run-time Pattern for Associated Architecture Work

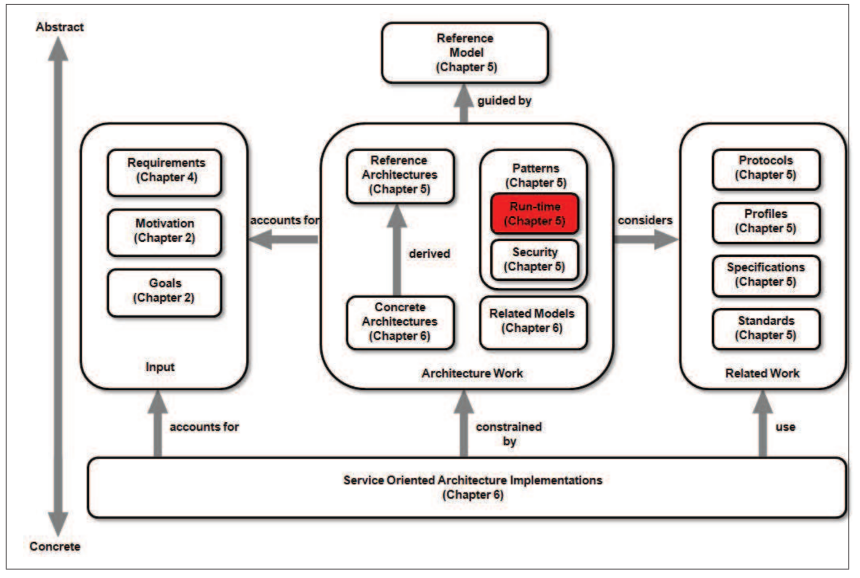


Figure 5.13: The run-time pattern provides insights into run-time perspectives of the reference architecture.

This section describes the IIRM run-time pattern that is associated to the overall reference model architectural design in order to understand approaches during run-time. The previous sections provide a rather static perspective on the IIRM architecture work and the implied support for different computational paradigms, while this section provides a more dynamic and more user-oriented perspective. As illustrated in Figure 5.13, the concrete infrastructure architectures are formed from the combination of reference architecture core building blocks as well as the run-time pattern. This partly addresses the requirement about architecture work patterns raised in Definition 77. The requirements addressed on the architecture level are summarized as part of Table 5.8.

The following paragraphs provide information for reference model adopters to better understand how scientific endeavors that require interoperability of production e-Science infrastructures (cf. Definition 5) are supported through the reference architecture design. The run-time perspective offers an algorithmic description how e-Science applications take advantage of different computational paradigms available through the combined use of HTC-driven infrastructures (cf. Definition 17) and HPC-orientated infrastructures (cf. Definition 16). It can be also used with hybrid infrastructures (cf. Definition 18). In [254], scientific workflows that can be essentially modeled as one greater algorithm are described using e-Science applications (cf. definition 9) with (a) HTC resources and (b) HPC resources with one client.

The academic analysis of the case studies identified a pattern how e-Scientists (cf. Definition 10) work with such emerging interoperable infrastructures (cf. Definition 21) using an emerging network of interoperable services (cf. Definition 15) rather than one single infrastructure. The analysis revealed a certain pattern published in [254] as one *'design pattern in e-Science'* alongside several others already existing (e.g. in [182]). This e-Science design pattern can be considered as a *'toolset'* for architecture adopters and e-Scientists when different types of computing paradigms for one larger scientific purpose are required. This requirement is often raised in scientific field-specific e-Science environments (e.g. ESFRI RIs [274]). This toolset is independent from any particular existing production e-Science infrastructure and applicable to a wide variety of e-Science endeavors but is based on the overall reference model architectural design. The associated pattern to the reference model and its reference architecture are thus useful to be looked up for various purposes such as understanding the run-time behaviour of the reference architecture at large.

It is also useful for comparisons with other reference models and associated architectures in distributed systems. The design pattern is just one way of providing an abstract notion that is not too specific but gives a reasonable frame of reference in the context of e-Science infrastructure that offer specific infrastructure resources (cf. Definition 6). It enables a better understanding of the approach taken in many scientific workflows and enables easier comparison as to whether or not emerging infrastructures (e.g. ESFRI RIs) require interoperability setups.

Chapter 2 introduced the state-of-the-art of production e-Science infrastructures and provided some pieces of information about their resources that are mainly computational resources, including storage resources to store results and data. End-users are satisfied in using one dedicated e-Science infrastructure to perform science, but also have an increasing demand for using multiple infrastructures jointly together for one larger scientific workflow.

Current non-interoperable infrastructures (cf. Definition 19) represent a hindrance for scientific work and overall support for e-Science (cf. Definition 3). The survey of current work practice in Chapter 2 and 3 reveals that e-Scientists often use one particular client technology that is commonly used in the corresponding field of e-Science (e.g. Kepler [215]), but which has been recently augmented with client libraries [225] to obtain the benefits of Grid infrastructures (cf. Definition 7). This multi-infrastructure workflow is precisely defined with the help of the following *'basic reference architecture algorithm'* illustrated in Figure 5.14.

The pseudo-code exactly defines the design pattern, which describes how the adoption of the overall IIRM and its associated reference architecture can be used by e-Scientists in scientific workflows that cross the boundaries of e-Science infrastructures. Section 2.3, revealed that interoperability is just starting to emerge on existing infrastructures and thus many of these workflows of the thesis studies are based on hacks, workarounds, and made possible via tweaks of additional standard development to support different concepts (e.g. integration of several client libraries in parallel). Many of these hacks and workarounds are published in [256] describing the thesis activities undertaken as part of the GIN community group that is a reasonable basis for further thesis contributions, including detailed case studies that are described in much more detail in Chapter 6. But these solutions are neither sustainable nor use the underlying infrastructure resources in the most efficient manner (e.g. HPC resources), but the workflow can be implemented with e-Science technologies guiding thus also the architectural design. Based on this lesson, the reference architecture and its associated algorithm aims to provide sustainable solutions. The usage of resources in a more efficient manner is possible through refined open standards that are interlinked in many ways as described in later parts of this chapter.

Figure 5.14 offers a run-time perspective to the overall reference model and its architectural design with a perspective from end-users and their usage of the core building blocks that enable the creation of individual infrastructures (cf. Definition 20).

As illustrated in Figure 5.14, the algorithm provides a high-level description of the usage of the core building blocks while this chapter will go into more technical details when the design layouts are described later. The algorithm starts in part (1) with the exposure of resource specific information for each IIRM service instance via the GLUE2 core building block of the

```
Begin
  Begin GridInformationProvisioning
    Grid Information Providers (GIPs) publish pieces of
    information about infrastructures (HPC and HTC resources)
    via out-of-band information service
  End

  scienceworkflowfinished = false

  WHILE (not scienceworkflowfinished)
    Begin Brokering
      End-user uses client technology (CT) and performs application setup
        and defines HPC or HTC requirements for next scientific workflow step
      Compute resource (CR) of corresponding HPC and HTC infrastructure is
        found based on the information exposed by GIPs
    End

    Begin JobSubmitToResource
      If CR.type is HTC then
        End-user of CT submits HTC job to a HTC resource
        using middleware MA of the corresponding infrastructure IA
      End If
      If CR.type is HPC then
        End-user of CT submits HPC job to a HPC resource
        using middleware MB of the corresponding infrastructure IB
      End If
    End

    Begin AnalysisScienceComplete
      Transfer and store (intermediate) results in storage technologies
      If end-user need no further computing then
        scienceworkflowfinished = true
      End If
    End
  End While
End
```

1 The information provider sends information using the GLUE2 elements about available resource information (e.g. Available core/CPUs, amount of memory, etc.). The Information service exposes it with a GLUE2 Instance to clients.

2 End-users provide requirements for resource description that are required for HTC and HPC jobs across interoperable e-Science infrastructures available. GLUE2 is used to obtain list of resources.

3 In re-using GLUE2 elements in JSDL we preserve the same semantics of the elements that are initially provided and exposed by the Grid resources itself leading to effective executions via OGSA-BES.

4 Execution are often part of larger scientific workflows with several executions, but after each loop in the algorithm scientists need to transfer and store (intermediate) results with ByteIO, HTTPS, GridFTP and SRM.

(i) Listing: Pseudo-code of basic reference model algorithm with reference architecture core building blocks in context

Figure 5.14: Pseudo-code of the basic reference architecture algorithm using different computational paradigms.

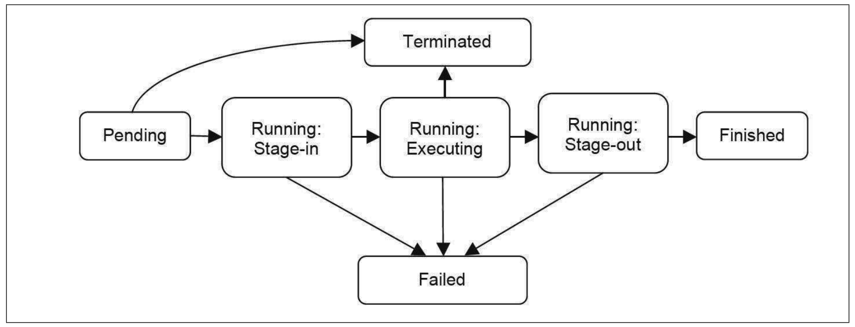


Figure 5.15: Basic reference architecture state model for a Grid job run on computational resources [305].

reference architecture. This can be done with information services that obtain computational resource (CR) (or storage resource) details from information providers, implemented as part of the other core building blocks that represent the IIRM set of services such as OGSA-BES (or SRM in terms of storage).

In part (2) of the algorithm, the e-Scientists set up their resource requirements and identify necessary services that match their specific requirements for the corresponding resource types (i.e. HPC and HTC). The GLUE2 core building block here offers a standardised way of describing resources across infrastructures, thus contributing to the fact that the process of matching requirements is more effective. Once the application setup (e.g. executable definition, etc.) is done, the job is sent to computing resources that match their specified requirements in GLUE2. This includes the supported security setup as the subsequent section about an associated security pattern will reveal in Section 5.1.6.

Part (3) of the basic reference architecture algorithm is about another aspect of the IIRM and its associated architecture, referring to the use of GLUE2 within JSDL. This makes sense to specify resource requirements in the exact way as it was specified by the administrators that are initially responsible for exposing the resource details in GLUE2 thus contributing to semantic interoperability as described in Section 5.1.4. This is one solution for the missing links introduced earlier that are also addressed in subsequent sections in this chapter in more detail after the general design ideas of the reference architecture are introduced.

For example, in using an e-Science application (cf. Definition 9) with the exact resource requirements in JSDL, the job is sent via OGSA-BES using the baseline communication mechanism illustrated in Figure 5.7 of Section 5.1.3. Being semantic identical as the information exposed, the accurate job description leads the Grid job to the corresponding resource types for effective execution using the GIN execution environment core building block underneath.

In part (4) of the algorithm, the results of the computational executions are stored using the core building blocks such as GridFTP or HTTPS as transfers to storages (i.e. SRM core building block). This might be the final results or intermediate results, depending on the applications and the larger scientific workflow that is only known by the e-Scientist. They might decide whether the scientific workflow is finalised, otherwise the algorithm starts with part (1) again. The algorithm can be executed using only HPC or only HTC resources as needed.

The major parts of the algorithm enables execution on either HTC or HPC resources and a more refined dynamic run-time perspective is as follows. The detailed run-time approach is based on the OGSA-BES specification in general and the HPC File Staging Profile (HPC FSP)

Requirement Definition	Addressed in which manner
Definition 77 (Architecture Work Patterns)	Run-time pattern with algorithm and state model;

Table 5.8: Addressed requirements of architecture patterns on reference architecture level.

[305] in particular. HPC FSP defines a *'state model'* [169] that represents the *'basic reference architecture state model'* defining states during the execution of one Grid application on computational resources. This is useful to have an semantically interoperable progress definition of application execution in Grid middleware, but it also informs end-users about the status of the application. It is a technical architecture work element, but also, from the end-user perspective, provides a way of receiving feedback from the Grid service about the Grid job execution progress. The HPC FSP state model is based on OGSA-BES and illustrated in Figure 5.15.

Any progress information of the application execution must be encoded using the states of the HPC FSP (or implied OGSA-BES). Also information services that take the core building block GLUE2 as an information model must expose the states according to the HPC FSP states. This is thus a restriction for the usage of the `ComputatingActivityState` type of the GLUE2 specification [113] that indicates that a general accepted state model not yet exists. In contrast, this thesis uses the OGSA-BES states, including the HPC FSP extensions but also other extensions that are introduced in subsequent sections. The HPC FSP state model allows for extensions and refinements of states according to different setups. Later sections in this chapter will reveal that this extensibility of this state model is uses to extend it towards the IIRM basic state model. This does not prevent any further extensions to this state model apart from those extensions provided in this chapter. The extensibility can be useful for many purposes of other architectures that use parts of the IIRM (e.g. ESFRI RIs).

Finally, Table 5.8 summarizes the aforementioned invariants and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

5.1.6 Security Pattern for Associated Architecture Work

This section describes the IIRM *'security pattern'* that is associated to the overall reference model architectural design in order to understand security approaches. The academic studies of related work such as standard specifications often revealed that *'security is out of scope'* that is a major limitation when working with production e-Science infrastructures (cf. Definition 5). Strong security methods are often required by end-users or resource providers. Security is special and affects every layer in the architectural design and thesis studies revealed in many cases that security is one of the *'major showstoppers'* for interoperability. As illustrated in Figure 5.16, the concrete infrastructure architectures are formed from the combination of reference architecture core building blocks as well as the security pattern. The requirements addressed on the architecture level are summarized as part of Table 5.9.

This section partly addresses Definition 77 in order to provide insights to security that go beyond the IIRM security core building blocks. The security pattern describes the basic security access paradigm of Grid services (cf. Definition 13) that enhance interoperability of production e-Science infrastructures. The thesis approach refers to horizontally (i.e. compute and data entities) and vertically (i.e. information and security entities) reference model elements that have the relationships as described earlier in Section 4.1.2 using the term *'orthogonal'* as in [267].

The scope is thus beyond standardization specifications that do not specify such an approach in detail. In addition to the run-time pattern, this approach is another architecture design pattern that govern the reference architecture in order to outline a clear path on how to

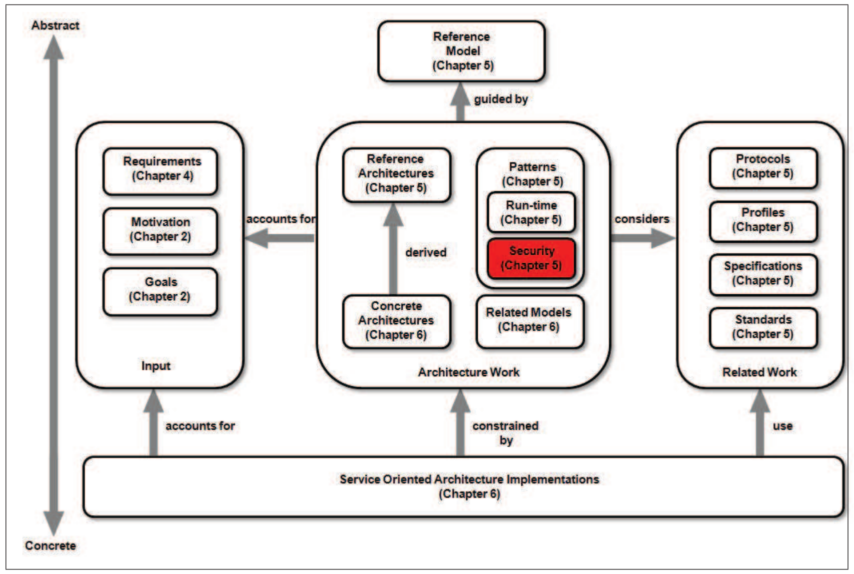


Figure 5.16: The security pattern provides insights into security perspectives of the reference architecture.

achieve a secure interoperability between the core building blocks that work on different levels in the architecture. Grid security in itself is a very complex topic and in order to not only focus on this topic this thesis only outlines the basic security approaches referring to related work where possible (e.g. [131]).

The different basic design pattern elements beyond entities and their relationships that have been already described in the general architecture work are collectively called *'plumbing'* [272]. This term represents an analogy to a house with hot and cold water plumbing, which all do not interfere with each other, but take effect at different levels within a house or cause different functionality at different levels within the overall house architecture. In the reference model architectural design, the different house levels stands for the different layers within a certain Grid service instance while processing a SOAP request and response [190]. Since an incoming SOAP request splits the different required functionality elements (authentication, attribute-based authorisation, job submission, etc.) at the Grid middleware level, a variety of different plumbing types is defined in order to increase interoperability at different levels as shown in detail in Figure 5.17.

The key invariant for plumbings is that the different plumbing types do not interfere with each other and one plumbing type cannot influence another plumbing type. They are partly addressing the orthogonal dependencies of the core building blocks introduced in Section 4.1.2 on the reference model level. They grey indicators in Figure 5.17 mark those plumbing elements not used in context.

As described in [272] and [266], the benefits of the *'plumbing design principle'* is that plumbing can be removed over time without breaking the functionality of another plumbing, and they neither interfere with each other nor influence each other in any way. The plumbing can exist in parallel and through adoption within the Grid middleware, one particular plumbing can significantly increase the chance for interoperability between Grid services deployed on current production e-Science infrastructures.

They address the requirement stated in Definition 68 supporting older concepts since new concepts and technologies emerge slowly in production setups. An old plumbing system (e.g. GSI X.509 proxies [300]) can be removed slowly over time since old technologies often remain for a long time while the e-Science infrastructure already partly adopts new technologies (e.g. SAML) as part of its evolution process and emerging plumbings. These *'legacy technology problems'* refer to old technology elements where other technology elements depend on and thus can not be removed without having collateral effects on the infrastructure architecture.

Figure 5.17, illustrates four basic types of plumbing that are indicated with numbers and that all need to be used within the reference model architecture as part of the security pattern. Production e-Science infrastructures choose their setups according to the overall governing security policies (e.g. proxies or no proxies). Actual deployments of these plumbings remains a policy decision that this thesis is not able to solve while some recommendations are given it in the seven segment-based process in Section 5.3. A short overview of the basic design idea of plumbings is given and more technical details about concepts that rely on it are described in later chapters (e.g. application case studies). While Figure 5.17 illustrates technical details, Figure 5.18 shows one example of the concrete deployment of the plumbings (indicated with different numbers) in context of IIRM core building blocks that are not directly security related (e.g. OGSA-BES, SRM, etc.) but affected, because of their *'orthogonal dependency'* [267].

The first plumbing type is the *(1) Transport Layer Security (TLS) protocol* [153]. Thus GSI connections [167], which slightly change the TLS protocol, are not allowed to be adopted in

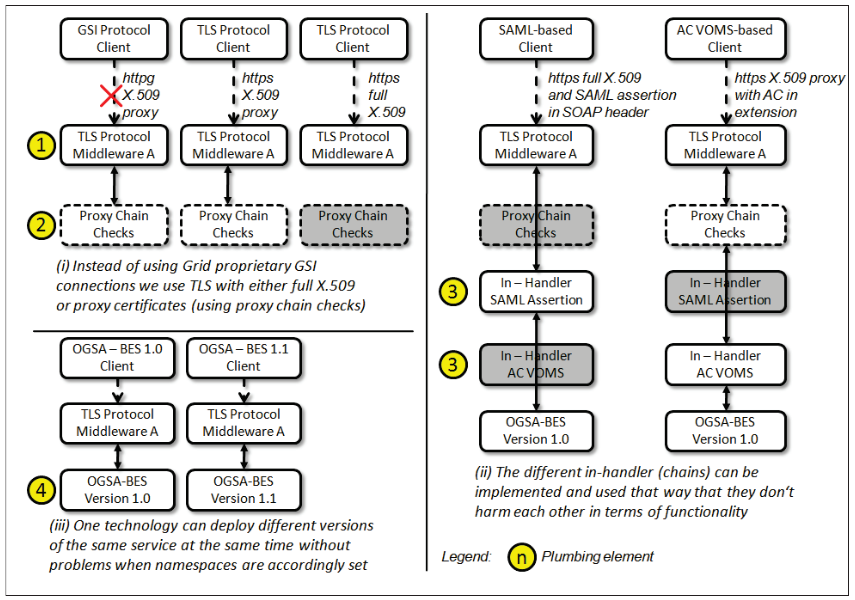


Figure 5.17: Conceptual view of the plumbing concept with reference architecture core building blocks.

Grid technologies that adopt the IIRM security pattern. A significant element of this pattern is that all architecture core building blocks support connections that are directly compliant with TLS/SSL. The second plumbing is a design principle to enable *(2) proxy chain checks* as specified as the *Proxy Certificate Path Validation Algorithm* in the IETF3820 standard [300]. This is required when proxies are used to access a certain Grid middleware and thus improves Grid interoperability significantly on the technical level, including delegation. This concept is possible since proxies are also X.509 certificates and thus usual TLS is possible, but requires a hierarchical algorithm [300] to check the proxy chain in case a proxy is used to contact an IIRM Grid service. The pattern uses proxies but those not created via proprietary GSI connections (i.e. HTTPG) [170].

The third plumbing type named as *(3) handler-chain*, refers to the known concept of using different in- and out-handlers while sending/processing SOAP requests and responses within service containers such as Apache Tomcat [149]. This type enables more than one transportation channel for the attribute-based security setup. In [302] an example of how the typical UNICORE non-attribute-based authorization is augmented with such a plumbing is described in order to enable attribute-based authorisation based on SAML assertions without breaking the usual security setup of UNICORE. This includes the usage of the WS-Security [131] core building block (cf. Figure 5.7). Hence, in the context of the larger reference model setup, this handler-chain concept is one plumbing type where vertical (e.g. security standards) and horizontal elements (compute and data standards) work together. In defining exactly which handlers are available in the architecture work, a couple of well-defined plumbing systems of this type is defined that in turn increase interoperability in many production e-Science infrastruc-

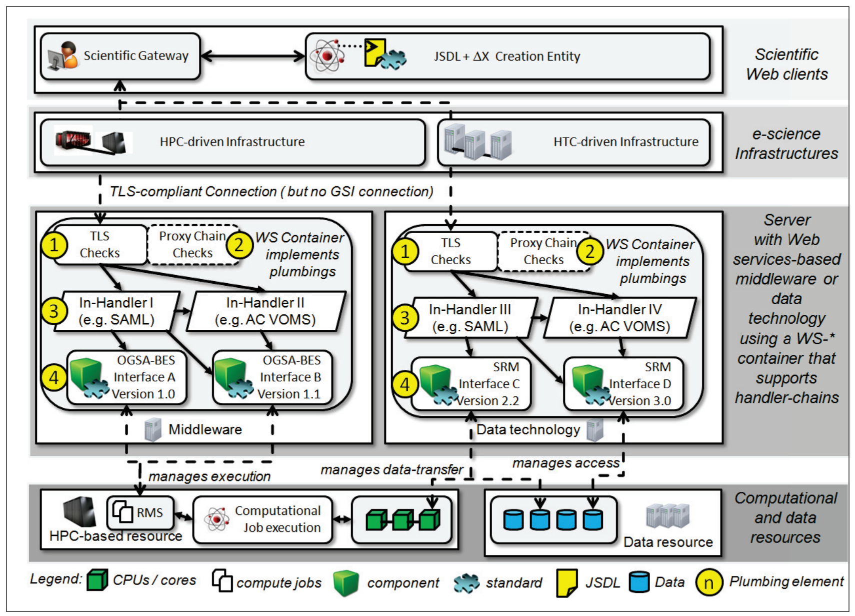


Figure 5.18: Example of the plumbing deployment concept as part of the security pattern.

tures today. As the plumbing systems are a pattern, IIRM services do not need to implement them all. The pattern rather defines the approach and provides thus a frame of reference for IIRM security setup adoptions as part of concrete infrastructure architectures that are governed by different security policies today.

Another plumbing type is the concept of *(4) parallel portType deployment*. Definition 68 mentions backwards compatibility within a Grid technology and the deployment of emerging technologies while production technologies and interfaces remain. As shown in Figure 5.18, the design concept of parallel portType [150] deployment provides an approach to enable the use of different versions of standard specification interfaces like with the OGSA-BES portType specification elements in version 1.0 and emerging 1.1 assuming they are defined in different namespaces.

There are two interesting further aspects of the plumbings approach that are important to understand in the larger context of the architecture work. The IIRM defines a very close dependency between information and security and that is one of the reasons why [272] describes the information as another plumbing (e.g. electricity vs. hot/cold water in the house analogy). It is important to know *'which Grid service supports which implemented security plumbing'* that is addressed with the information service constraints in Definition 69. This means that each Grid service offers different plumbings that are well-described and exposed in order to be accessed by other services or clients, or as a more general term *'agent'* as in Figure 5.19. The pattern thus re-uses a mechanism that is in-line with the traditional Web Service description approach as part of a fundamental Web service architecture [136] illustrated in Figure 5.19. In the context of this thesis, it illustrates that services (i.e. resources) are described (i.e. resource description) while referring to security policies (i.e. plumbings SAML, X.509 proxies, etc.). In addition, Grid clients (aka agents) discover compatible Grid services and access them in a secure manner.

The second aspect is that *'precisely defined plumbing'* do not solve the interoperability problem completely, but they significantly increase the probability of successful interoperability far better than implementations without defined plumbing concepts. This is mainly achieved through a decrease of the amount of possible interaction possibilities with the corresponding services. One example of the use of the plumbings approach is illustrated in Figure 5.18 while clients have used an information service beforehand (cf. Web service architecture [136]) in order

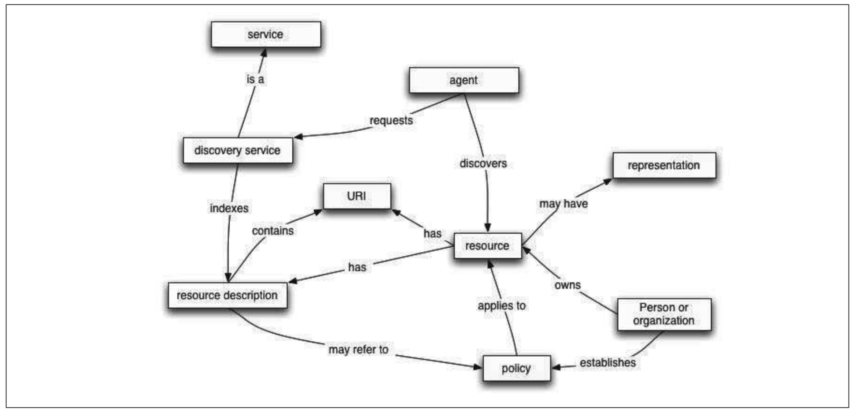


Figure 5.19: The W3C WS architecture [136] and the important principle of discovery.

to know which services offer which security plumbings. The information plumbings have been first used in order to obtain information about available security plumbings. The demand of contacting an information service beforehand is an overhead compared to direct service access, but with this pattern it clearly increases interoperability through plumbings. Furthermore, it increases the efficiency of resource usage using the concepts revealed later in this thesis.

One crucial design element of the plumbing-based security pattern in general is related to the concept of *'delegation of end-user rights'* as illustrated in Figure 5.20. This concept stands for a process that enables one end-user (or agent) to act on behalf of another end-user. Delegation is often used to enable third-party file transfers before execution (e.g. data-staging) or in brokering setups (submission delegation to another entity). Within the security domain delegation is known to be a very complex topic, but a brief discussion is needed also in this thesis in order to provide all the necessary design elements of a security pattern without leaving out essential functionality. In order to not loose the reference model focus, the work around delegation is mostly referencing related work rather than provide deep technical descriptions.

Delegation is one key benefit in using the traditional GSI connections [170] that is not used in the pattern. Delegation is based on using the PKI [195] and thus X.509 certificates including proxies as defined in [300]. One of the most significant constraint in the delegation concept using X.509 certificates is that the *'private key'* is never transferred over the network and that each private key is unique.

Being compliant to this concept, the delegation concept of this pattern is illustrated in Figure 5.20 acting as an alternative to GSI-initiated proxy generation. It reveals that a method is followed from the *'GridSite Certificate Delegation'* protocol [41] that also includes the functionality of proxy renewal (because proxies have a limited lifetime) as well as explicit proxy removal. This mechanism is different than the broadly known method of establishing HTTPG

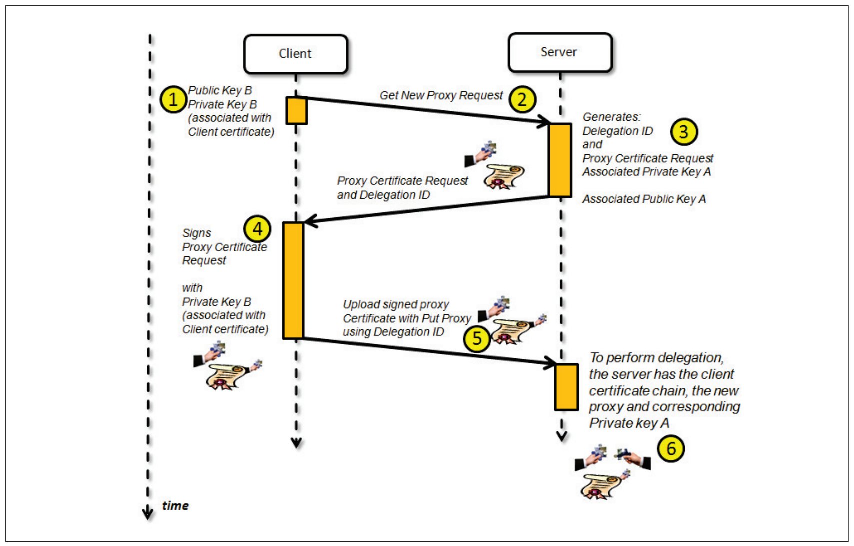


Figure 5.20: A crucial element of the plumbing concept is the certificate delegation method.

Requirement Definition	Addressed in which manner
Definition 68 (e-Science Production Technology Adoption Constraint)	Security pattern works with legacy setups;
Definition 77 (Architecture Work Patterns)	Security pattern including delegation of rights;

Table 5.9: Addressed requirements of security patterns on the reference architecture level.

connections in GSI [170]. A basic step-wise description is given in Figure 5.20 while more details can be found in [41]. The key idea with GridSite is to use HTTPS and portType operations on message level instead of traditional GSI methods (i.e. HTTPG) on the transport level.

In step (1), each end-user (i.e. client) has access to a X.509 certificate, including its Private Key B and Public Key B. In step (2), the client initiates a new proxy request. The server then creates in step (3) a delegation ID as well as a proxy certificate request and its associated Private Key A. In step (4), the client then receives this proxy certificate request and signs it with the associated Private Key B from the certificate. This signed proxy certificate is then send back in step (5) to the server using the particular Delegation ID. Step (6) concludes the process providing the server with a client certificate chain and the new proxy and the corresponding private Key A. The GridSite specification [41] provides more details.

This approach prevents the use of proprietary HTTPG connections (cf. Figure 5.17) and instead HTTP(S) connections and plain SSL are used together with several dedicated portType operations. The SSL connection does not require any special setup and is interoperable with other SSL-compliant technologies and the only check that is required is the proxy chain checks for the proxies themselves. The key idea is thus that the additional byte that breaks interoperability in HTTPG-based transport-level connections is removed and the problem of delegation is *'moved one level up'* to the service-level with dedicated portType operations specified by GridSite that a Grid service needs to implement. The delegation element is thus encapsulated from the transport element and is part of the service-level still keeping the same functionality (and the support for proxy certificates). The major gain is that each service that implements this pattern can be contacted with HTTPS connections using additionally proxies for delegation if necessary.

Starting with OGF31 [68], this delegation concept is standardised given its widespread use and its success of avoiding proprietary GSI connections. It should be noted that the delegation approach of this part of the pattern has been not invented in this thesis and as such has been only re-used to streamline the security setup and to avoid the use of GSI proprietary protocols. More details about the specific approach can be found at [41].

Finally, Table 5.9 summarizes the aforementioned invariants and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

5.2 Design Layout and Essential Functionality

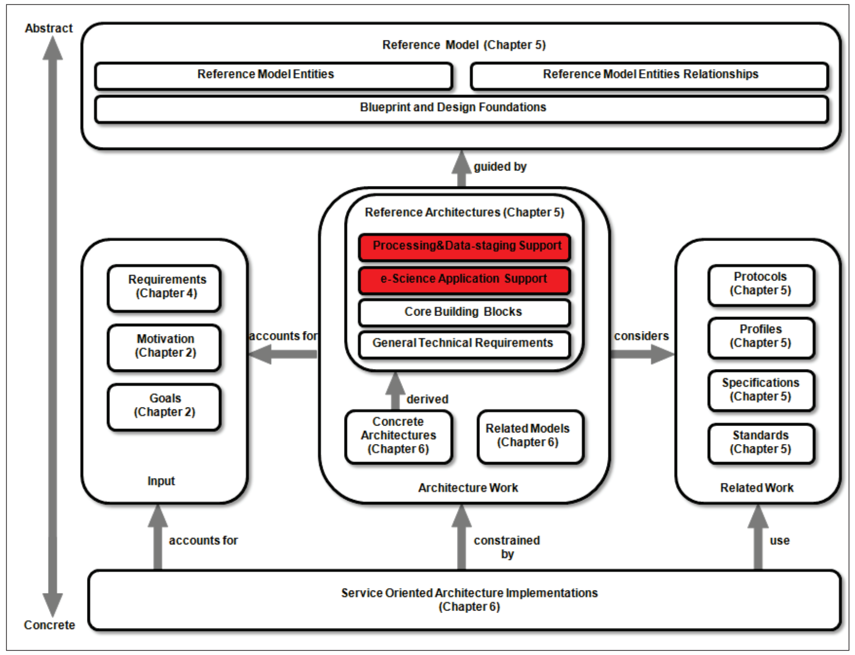


Figure 5.21: Reference Architecture Core Building Blocks Improvements.

The last section introduced the general reference model design and its associated architecture work in terms of a reference architecture that considers protocols, profiles, specifications, and open standards. The reference model and its related elements are a mechanism for understanding the problems faced within particular environments and outline potential solutions. In addition, this section describes the essential solutions with functionalities of the reference architecture addressing many functional requirements of Section 4.2. The essential functionalities are described via several concepts that are based on the core building blocks of the reference architecture and in the several refinements proposed for them. All these refinements are specifically focussed on the improvements for production e-Science application support as well as better processing and data staging concepts as illustrated in Figure 5.21.

5.2.1 e-Science Application Concepts

The general design layout supporting production e-Science applications (cf. Definition 9) is given by the reference architecture building block OGSA-BES that implies the use of JSDL. These building blocks are capable of providing the functionality required by the Grid execution management service (cf. Definition 49) and its implied Grid job description language schema. But interoperability studies revealed that these core building blocks can be improved in the context of supporting e-Science applications [270]. Grid application job descriptions need to be improved to more meaningful descriptions in order to enable Grid middleware (cf. Definition 12) in turn to more effectively execute Grid jobs with e-Science applications. The general ideas behind these concepts are illustrated in Figure 5.22 and the re-use of several parts of the GLUE2 information model schema is proposed as defined in Definition 55. The requirements addressed on the architecture level are summarized as part of Table 5.11.

Lessons learned in using Grid middleware with applications using multiple infrastructures point to more detailed descriptions about the Grid job itself [270]. It is often defined with the standard JSDL [115] as well as its extensions and profiles (i.e. JSDL SPMD Extension [281], HPC FSP [305], and HPC Profile Application Extensions [197]). A number of analysed refinements to JSDL are proposed that are published in [261] and that are illustrated in Figure 5.22, in order to understand where these improvements take effect. An overview of these functionality extensions and improvements are shown in Table 5.10.

In addition to JSDL, the concept of *(a) application types classification* addresses Definition 57 and provides useful information about the Grid job that affects its handling within Grid middleware systems (e.g. parsing effectiveness, etc.). The classification consists of the following enumeration: *serial, collection, metaparallel, parallel, pre-installed, benchmark, compilation and workflownode*. The key benefit within the reference architecture core building block JSDL is to provide Grid middleware with information that can be used to parse and process JSDL-based applications much more effectively. The indication of the application type leads to much faster executions since the implementation logic of application types often differs from each other. One Grid application can be described by more than one element of this classification. To provide an example, the processing logic of *'pre-installed'* and *'compilations'* types, are often

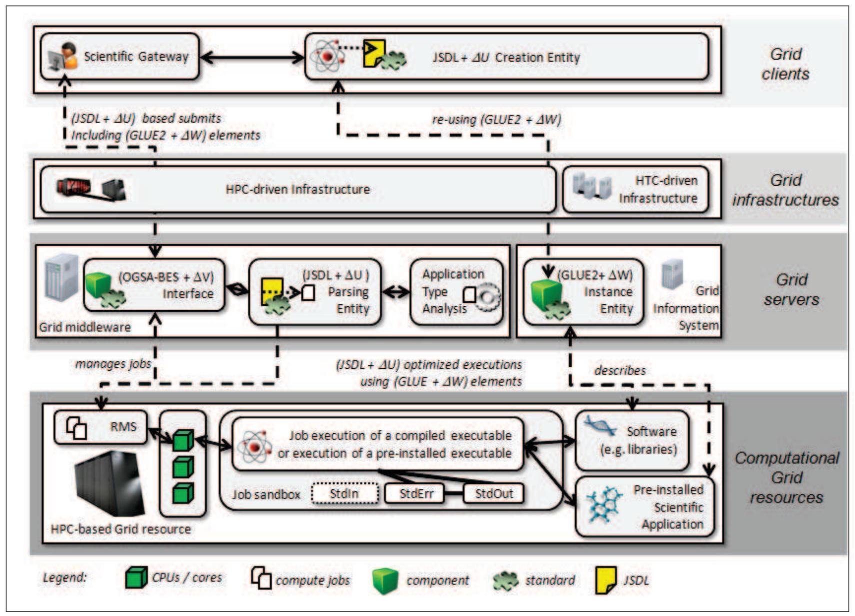


Figure 5.22: The e-Science application concepts with reference architecture core building blocks.

Functionality Extensions and Improvements	Area	Extended Standard
(a) Application types classification (e.g. parallel, etc.)	Compute	JSDL
(b) Application type refinements (e.g. pre-installed)	Info	GLUE2
(c) Revised application executable definition	Compute	JSDL
(d) Application software statement	Compute	JSDL
(e) Application family extension (e.g. library)	Info	GLUE2
(f) Application software requirements	Compute	JSDL
(g) Application output joins	Compute	JSDL

Table 5.10: Functionality improvements of the e-Science application concepts.

fundamentally different and thus execution runs can be prepared by the middleware much more effectively when the type is known.

The element *'serial'* stands for an individual stand-alone job while the *'collection'* element refers to a job that is submitted as part of a collection of individual jobs that do not communicate between each other. This is often used in HTC use cases or parameter sweep studies. The classification *'metaparallel'* refers to a job that is submitted as part of a collection of individual jobs, but these are jobs that communicate with each other via the mechanisms of meta-computing (i.e. Meta-MPI [207]). This is rarely used, but some use cases (e.g. the VPH community [219]) require this mechanism for crossing numerous production Grids. The *'parallel'* job classification refers to a job submitted as part of a larger collection, but the communication is basically performed in the job itself using parallel computing mechanisms (i.e. MPI or OpenMP). These jobs are often used in the context of HPC-driven e-Science infrastructures (cf. Definition 16).

Another interesting type is *'workflownode'*, which is submitted as part of a larger cross-site workflow and handled by a workflow engine technology. The application type *'pre-installed'* provides information to the Grid middleware system that the application is already installed on the relevant system. The *'compilation'* application type indicates that the application must be first compiled on the corresponding system before being executed. Another useful type is the *'benchmark'* type since, in many production Grids, benchmarks are often handled differently by the Grid middleware systems in general and by the underlying Grid resource in particular (e.g. different job queues on HPC resources, measurements, etc.).

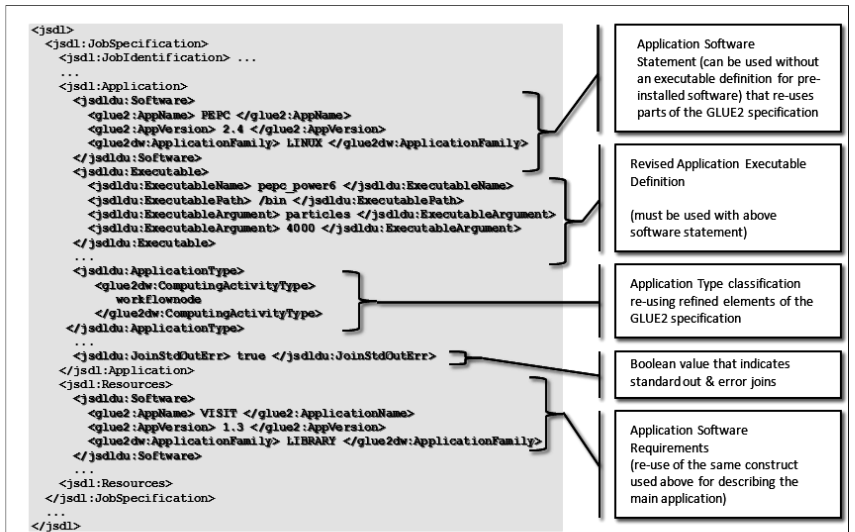
The GLUE2 specification [113] already provides an enumeration list named as *'ComputingActivityType_t'* for several of the aforementioned application types that are *'collectionelement'*, *'parallelelement'*, *'single'*, and *'workflownode'*. This is a missing link between the JSDL and the GLUE2 specification. The *'collectionelement'* is simply re-used as *'collection'* within the JSDL, and the aforementioned types are refined as follows. As part of *(b) Application types refinements*, the GLUE2 *'parallelelement'* refers to *'metaparallel'* in the reference architecture and another *'parallel'* element is added since both indicate fundamental different types of parallel computing concepts. There is a need to differentiate between those two especially when e-Science applications are used that in turn means different ways in which communication works. In addition to these refinements around parallel execution, the *'pre-installed'*, *'benchmark'*, and *'compilation'* application types are added to the GLUE2 enumeration.

An example of solving the missing link is given in Figure 5.23 that indicates *'GLUE2 + ΔW'* where ΔW here stands for the additions to the GLUE2 specification. Figure 5.23 provides an XML-based document with GLUE2 extensions elements (i.e. GLUE2dw namespace) to the original JSDL and their refined elements (i.e. JSDLdu namespace). The *jsdldu:ApplicationType* consists of one *glue2dw:ComputingActivityType* element that indicates that the listed job description defines an e-Science application that is part of a greater workflow (i.e. workflownode).

The definition of the main Grid application executable within the Grid job description represents a challenge due to the wide variety of inhomogeneous systems and resources leading to the following requirement Definition 58. All existing concepts in JSDL like its POSIX normative extension (i.e. POSIXApplication executable definition [115]), which is also used in the JSDL SPMD extension [281], or its HPC Profile Application extension [197] have only used an *'Executable'* element and an *'Argument'* element. But the path information of an executable needs be clearly separated from the executable enabling less error-prone parsing JSDL documents in setups across infrastructures.

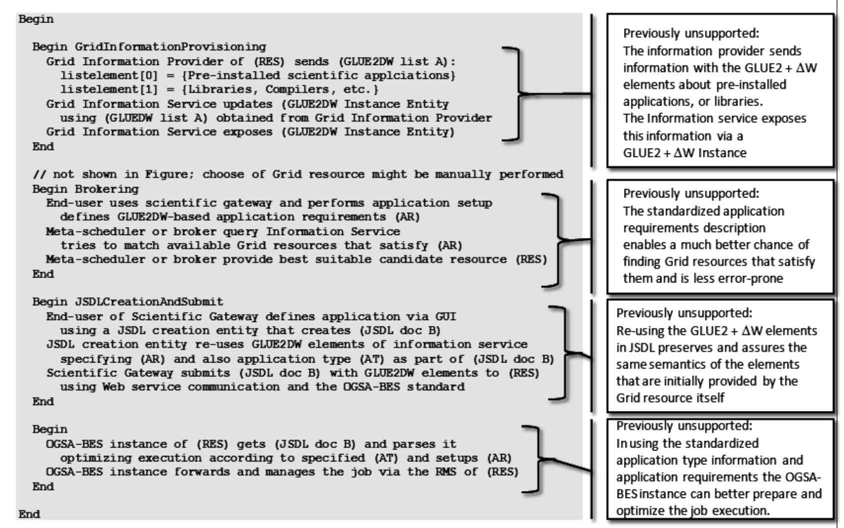
A *(c) revised application executable definition* is proposed with the three elements *'ExecutableName'*, *'ExecutablePath'*, and *'ExecutableArgument'* while the latter element can appear n times in the job description. Prototypes explored that this concept enables the most flexible support in terms of supporting different varieties of job submission approaches that are (i) compiled and executed applications in the job sandbox, (ii) pre-installed applications with a fixed and known path, and (iii) pre-installed applications that take advantage of complex constructs using environment variables (e.g. $PATH variable) and such like. One example is illustrated in Figure 5.23 describing the *'pepc_power6'* e-Science application [244] that is executed from the *'bin'* directory with two arguments.

Another refinement to JSDL is *'ApplicationName'* and *'ApplicationVersion'* information. By introducing the *(d) application software statement* concept these are addressed and the *'ApplicationFamily'* (e.g. LINUX, WIN, Library, etc.) is added. This concept is re-used to define a much clearer formulation of the *(f) application software requirements* concept. Only one instance of the application software statement concept describes the main Grid application itself in the *'Application'* element of the JSDL instance. Other n instances of it are used to describe application



(i) Listing: Example of JSDL + ΔU instance with more meaningful Grid job descriptions, also based on GLUE2 + ΔW.

Figure 5.23: Design layout for the e-Science application concepts in JSDL and GLUE2 with refinements.



(i) Listing: Pseudo-code of basic reference model algorithm that illustrates previously unsupported concept elements

Figure 5.24: Pseudo-code using the e-Science application concepts.

software requirements in the *'Resource'* element of the JSDL instance. Both address requirements defined in Definition 59.

Figure 5.23 illustrates both aspects of the concept using the example of a defined Pretty Efficient Parallel Coulomb Solver (PEPC) application [244] that requires a dedicated visualisation library called VISIT [139] during run-time for its execution on computational infrastructure resources (cf. Definition 6).

GLUE2 already provides a suitable approach that is re-used in JSDL and thus provides a solution for yet another element of the missing specification link between them. The *'ApplicationEnvironment class'* of the GLUE2 specification [113] and its elements are well suited to describe the *'ApplicationName'* with the GLUE2 *'AppName'* element and the *'ApplicationVersion'* with the *'AppVersion'* element. Only the GLUE2 *'ApplicationEnvironment'* needs to be refined by adding these elements that are collectively refered to as *(e) Application family extension*. Figure 5.22 illustrates an information system that exposes all the aforementioned pieces of information about a system and its available GLUE2-based *'ApplicationEnvironment'*. It is exposed since the ApplicationEnvironment is part of a GLUE2 *'ComputingManager'* element while this in turn is part of a GLUE2 *'ComputingService'* information. The subsequently GLUE2-based described *'ComputingService'* instance can be an OGSA-BES service instance as shown in Figure 5.22 that is used in turn to submit JSDL + ΔU job description where ΔU are partly GLUE2 + ΔW elements.

Another lesson learned that influences the design decisions of the reference architecture is related to e-Science application outputs as defined in Definition 60. This raises the need to have a concept named *(g) application output joins*, which refers to a boolean value that indicates whether or not the *'standard-out'* and *'standard-error'* outputs of an e-Science application should be joined in one file. Some applications use the *'standard-error'* for significant output (e.g. AM-

BER MD suite [242]) and thus e-Scientists are often interested in only having one output file for the application run analysis. Figure 5.23 uses the *jsdldx:JoinStdOutErr* element with the boolean value *'true'* to indicate the merger of both outputs.

In order to take the overall reference model design and its associated reference architecture into account, major parts of the aforementioned functionality are part of the missing link between the core building blocks JSDL and GLUE2. The concrete XML renderings of GLUE2 can be defined and re-used within JSDL as the example in Figure 5.23 reveals.

In order to understand the concepts of this section, the pseudo-code in Figure 5.24 illustrates which algorithm aspects have previously not been supported. It describes which refinement leads to a more effective execution within the aforementioned basic reference architecture algorithm as part of the run-time pattern (cf. Section 5.1.5).

Finally, Table 5.11 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	**Addressed in which manner**
Definition 57 (Application Type Support)	Application types for more efficient job processing;
Definition 58 (Precise Application Executable Specification)	More precise application description;
Definition 59 (Application Software Mechanism)	Better support for pre-installed software;
Definition 60 (Application Output Joins)	Concept for joining application outputs;

Table 5.11: Addressed requirements for application improvements.

5.2.2 Application Execution Adjacencies Concepts

The concepts described in this section are based on the core building blocks OGSA-BES and its implied JSDL of the reference architecture. These represent the Grid execution management service (cf. Definition 49) and its implied Grid job description language schema. This section reveals the concept of application execution adjacencies in order to better support scientific application executions with important standard-based information at the lowest possible reference architecture level meaning infrastructure resources (cf. Definition 6). The link between the JSDL and GLUE2 specification that represents our Grid information model schema (cf. Definition 55) bears a lot of potentials to enable more effective job executions, especially in setups across infrastructures. The requirements addressed on the architecture level are summarized as part of Table 5.13.

The term *'adjacencies'* emphasises the fact that this concept is very closely related to the e-Science application execution process itself and on the targeted computational infrastructure resource. Such resources (i.e. small cluster, supercomputer, etc.) are managed by some form of RMS as defined in Definition 11 (e.g. Torque, LoadLeveler, etc.) that handle the application execution described by JSDL, after processing in the Grid middleware (cf. Definition 12).

At the lowest resource level, where the application is executed, the proposed new application execution adjacencies concept is applicable as shown in Figure 5.25. With the name *'adjacencies'*, this concept is thus better distinguishable from the term *'environment'*, which is used throughout this thesis to describe the whole problem-space of the reference model or that is often misunderstood with *'environment variables'*.

The fundamental idea of the execution adjacencies concept is published in [261] describing a Grid middleware-independent *'common execution environment (CEE)'* that can be used by e-Science applications during run-time. The realisation of this particular concept includes two major parts that are *'common environment variables'* (cf. Figure 5.25 example within the job environment) and *'common execution modules'* (cf. Figure 5.25 example with AMBER [242] module). Although both sound rather trivial, lessons learned from production Grid point to e-Science applications (cf. Definition 9) that fail by assuming the same execution environments on different resources, which is often not the case today. This not only refers to scientific application-specific environment variables, but also to different setups of a suite of scientific application executables (e.g. within AMBER) and their configurations.

With the concepts described in this section, an execution environment is defined that a Grid job application is able to *'assume'* and to *'access'*. The execution environment needs to be available on each infrastructure resource that offers access through the Grid execution management service in order to promote interoperability on application levels far beyond the IIRM interface level. These concepts are illustrated in Figure 5.25, while an overview of its functionality extensions and improvements is summarized in Table 5.12.

The first aspects of the execution adjacencies are relatively simply realisable by using *(a) common environment variables* across different middleware distributions, that addresses the requirement in Definition 61. In several e-Science applications (e.g. PEPC [244]), the executed source-code makes use of environment variables such as number of cores, or available memory. Every Grid middleware (e.g. gLite, ARC, UNICORE, etc.) provided such pieces of information via environment variables in proprietary execution environments. These proprietary

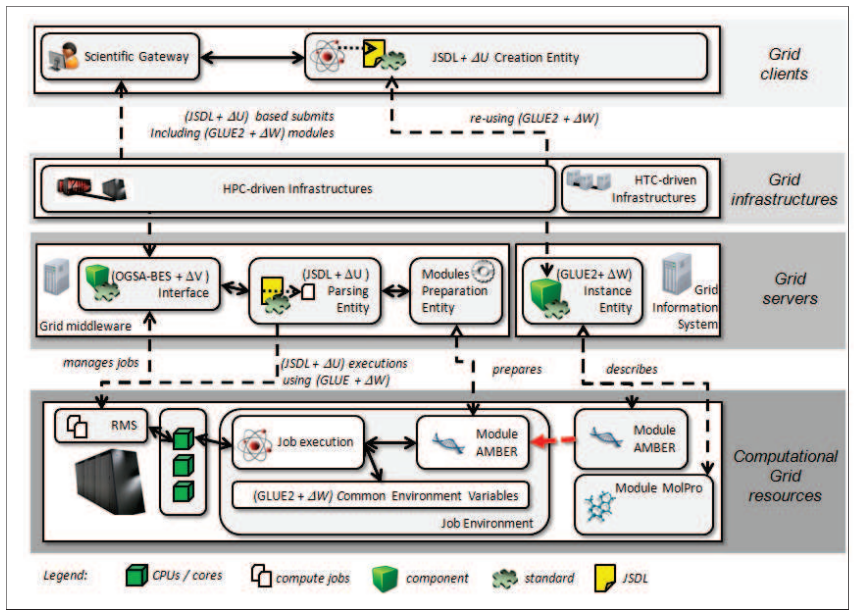


Figure 5.25: The application execution adjacencies concepts with reference architecture core building blocks.

Functionality Extensions and Improvements	Area	Extended Standard
(a) Common Environment Variables	Info	GLUE2
(b) Common Execution Modules	Compute	JSDL
(c) Execution Module Characteristics	Info	GLUE2

Table 5.12: Functionality improvements of the application execution adjancencies concept.

environments have no common syntax nor share the same common semantics. Interoperable infrastructure setups need to support multiple proprietary environments, or the applications simply fail when the executed source-code does not find an expected (and required) specific environment variable. A standardised list of environment variables is proposed defining their precise syntax and also the corresponding semantics. Additions from variables by purely locally installed applications can be added without breaking the concept. Initial work is carried out within GIN by Field et al. [38], which is linked with GLUE2 in this thesis and follows the same approach of harmonising some crucial environment information.

A few of the environment variables can be found in Figure 5.26, the content of which illustrates the major concept. The GLUE2 specification [113] defines the semantic details of the resulting detailed environment variables. Many of these variables provide information that must be consistent with information provided by a Grid information system for each Grid resource (i.e. by using GLUE2 schema elements). All pieces of information provided as environment variables need to follow exactly the same syntax and semantics as within the GLUE2 specification. GLUE2 provides attributes in the *'ExecutionEnvironment'* that provide information that is useful for running applications on Grid resources. It make thus sense to *'render'* those attributes as environment variables within the Grid middleware that is one part of the Grid job execution environment core building block defined as part of Definition 49.

A few attributes to the execution environment are added that are useful for applications during run-time. To provide an example, although GLUE2 defines the amount of physical CPUs, it make sense to provide the number of physical cores to address the different core setups (single-core, dual-core, quad- core, upcoming n-core, etc.) on computing resources. Such extensions to GLUE2 throughout this chapter and all the proposed additions to GLUE2 (indicated with + ΔW) as part of the design layout should also be rendered as environment variables. This will ensure a consistent execution environment across different infrastructures including syntax as well as semantics.

The second improvement is the *(b) common execution module* concept addressing the requirement defined in Definition 62. The concept idea is derived from DEISA/PRACE experience where support for different modules is done via *'common production environment'* [184] and using it for example in the context of the WISDOM case study [259]. In terms of realisation, a Grid middleware can use the module tool implementation from sourceforge.net [20] that was originally designed by CRAY. This concept is useful when working with applications that require a pre-defined setup of configurations like path settings in addition to environment variables. It is not about pieces of information about a pre-installed application setup as introduced previously (cf. Section 5.2.1) and instead it works on a far deeper level, meaning the level of the application execution itself.

To provide an example, the so-called *'AMBER module'* as shown in Figure 5.25 includes the configuration setup to run the AMBER scientific package [242], including 50-80 executables and programs. Hence, e-Scientists need to set up often the required details (e.g. set PATH and executable locations, AMBER environment variables, executables versions, etc.), but with this concept they just specify the required module in JSDL as part of the resource section (cf. Figure 5.25). The Grid middleware prepares the execution according to the module definition that has

```
<jsdl>
  <jsdl:JobSpecification>
     <jsdl:Application> ... </jsdl:Application>
     ...
     <jsdl:Resources>
       ...
      <jsdldu:Module>
         <glue2dw:ModuleName> AMBER </glue2dw:ModuleName>
         <glue2dw:ModuleVersion> 9 </glue2dw:ModuleVersion>
       </jsdldu:Module>
        ...
      <jsdl:Resources>
    ...
  </jsdl:JobSpecification>
  ...

</jsdl>
```

Definition of a required module for the execution of the Grid job, here a specific module is required for jobs named as AMBER

In comparison to the previously defined application software extensions, this extension is not allowed to be in the application statement and can only be used in the required JSDL resource specification re-using GLUE2

Environment Variable Syntax	Environment Variable Semantics
GLUE2: MainMemorySize	The total amount of physical RAM
GLUE2: PhysicalCPUs	The number of physical CPUs
...	*Others from the GLUE2 Execution Environment attribute specification*
GLUE2dw: PhysicalCores	The number of physical cores

The GLUE2 specification defines a set of Execution Environment attributes that we map as standard environment variables to increase the interoperability of Grid resource environments

Additionally required environment variables that we directly add to the existing Execution Environment set of GLUE2 attributes

(i) Listing: Example of JSDL + ΔU instance with module (top); (ii) Table of environment variables and GLUE2 + ΔW (bottom)

Figure 5.26: Design layout for the application execution adjacencies concepts in JSDL and GLUE2.

been previously configured for dedicated applications or libraries. GLUE2 is extended in order to support *(c) execution module characteristics* that are re-used in the JSDL resource context in order to more clearly describe the module characteristics.

In addition to the configuration of certain required scientific libraries, other use cases also influenced the design layout with modules. For instance, the handling of different versions of compilers in dedicated modules as in the case where different versions than the default compilers are also available. Another use case deals with license aspects of some scientific applications (e.g. MolPro quantum chemistry package [55]) where a license file needs to be in the end-users home directory every time an execution is performed with the MolPro executable. The invocation of the MolPro module prior to Grid job execution can solve the license issue for e-Science applications that rely on this package. While all these use cases and the overall approach of using modules for it is not new, its combination with open standards within the thesis making it a viable reference architecture element.

Grid information systems have to expose which modules are available at a corresponding Grid execution service and this can be done within the GLUE2 service description by re-using the *(c) execution modules characteristics*. Figure 5.26 illustrates an example how the AMBER module can be defined in the *'resources'* element of JSDL.

A concrete XML rendering of the introduced concepts can be thus defined addressing one aspect of Definition 49 in terms of the execution environment. The pseudo-code in Figure 5.27 illustrates which algorithm aspects have previously not been supported. It describes which refinement makes it easier for end-users that follow the aforementioned basic reference architecture algorithm as part of the run-time pattern (cf. Section 5.1.5) thus taking advantage of the proposed application adjacencies improvements.

```
Begin

  Begin GridInformationProvisioning
    Grid Information Provider of (RES) sends (GLUE2DW list A):
      listelement[0] = {Suppored Module AMBER}
    Grid Information Service updates (GLUE2DW Instance Entity)
      using (GLUE2DW list A) obtained from Grid Information Provider
    Grid Information Service exposes (GLUE2DW Instance Entity)
  End

  // not shown in Figure; choose of Grid resource might be manually performed
  Begin Brokering
    End-user uses scientific gateway and performs application setup
      defines GLUE2DW-based application requirements (AR)
    Meta-scheduler or broker query Information Service
      tries to match available Grid resources that satisfy (AR)
    Meta-scheduler or broker provide best suitable candidate resource (RES)
  End

  Begin JSDLCreationAndSubmit
    End-user of Scientific Gateway defines application via GUI
      using a JSDL creation entity that creates (JSDL doc B)
    JSDL creation entity re-uses GLUE2DW elements of information service
      specifying (AR) and also required module (M) as part of (JSDL doc B)
    Scientific Gateway submits (JSDL doc B) with GLUE2DW elements to (RES)
      using Web service communication and the OGSA-BES standard
  End

  Begin
    OGSA-BES instance of (RES) gets (JSDL doc B) and parses it
      prepares modules according to specified (M) and setups (AR)
    OGSA-BES instance forwards and manages the job via the RMS of (RES)
    OGSA-BES execution takes advantage of
      GLUE2DW common environment variables
  End
End
```

Previously unsupported:
The information provider sends information with the GLUE2 + ΔW elements about configured and available modules at a Grid site The Information service exposes this information via a GLUE2 + ΔW Instance

Previously unsupported:
The standardized module requirements description enables a much better chance of finding Grid resources that satisfy them and is less error-prone

Previously unsupported:
Re-using the GLUE2 + ΔW elements in JSDL preserves and assures the same semantics of the module elements that are initially provided by the Grid resource itself

Previously unsupported:
In using the standardized module information and the GLUE2 + ΔW common environment variables the OGSA-BES instance can prepare the job execution in a standardized way

(i) Listing: Pseudo-code of basic reference model algorithm that illustrates previously unsupported concept elements

Figure 5.27: Pseudo-code using the application execution adjacencies concepts.

Finally, Table 5.13 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	Addressed in which manner
Definition 61 (Common Environment Variables)	Common environment variables defined with GLUE2;
Definition 62 (Common Execution Modules)	Common execution modules for software applications;

Table 5.13: Addressed requirements for application adjacencies.

5.2.3 High Performance Computing Extensions

The basic design of the proposed reference model generally enables the use of HPC applications with the help of the Grid management execution entities as defined in Definition 31. Guided by these entities, the reference architecture and their concrete core building blocks OGSA-BES (cf. Definition 49) and its implied JSDL are used in HTC- and HPC-driven e-Science infrastructures (cf. Definitions 16 and 17). This section defines some refinements in terms of HPC application support of JSDL in combination with GLUE2 as the Grid information model schema (cf. Definition 55). The requirements addressed on the architecture level are summarized as part of Table 5.16.

Low-level information about resource features (e.g. available network topologies) must be exposed by information systems in greater detail in order to achieve more scalability and/or

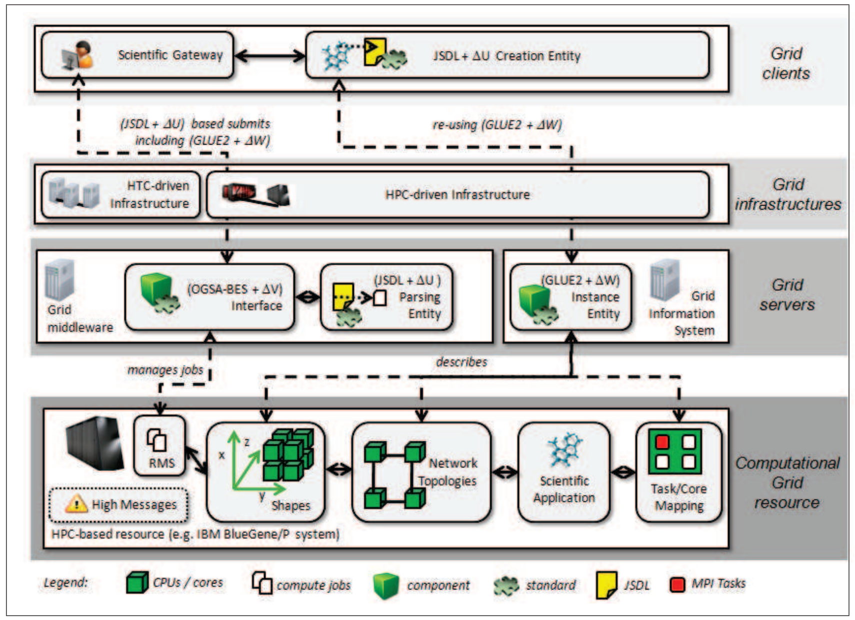


Figure 5.28: The HPC extensions concepts with reference architecture core building blocks.

the better performance of e-Science applications that take advantage of the low-level HPC functionality. The work presented in this section is a result of many different interoperability setups between production e-Science infrastructures and HPC-orientated improvements that are published in [261]. The core building block of the reference architecture is based on JSDL and its extensions, but those have been originally defined for more HTC-based environments. This is actually not only the case for JSDL itself, but also for its numerous extensions and profiles that have been mostly defined for Grid and HTC environments neglecting many of the HPC functionality although many can be found as this section reveals.

Also, they have been defined several years ago and thus they lack the support of concepts of recently used Grid resources in general and large-scale HPC systems in particular. JSDL [116], for instance, was originally defined in 2006 and revised in 2008 [115], but still lacks the required functionalities to enable an efficient HPC-oriented Grid job execution. JSDL extensions during 2007, such as the SPMD specification [281] or the HPC Profile Application extensions [197], were aimed at delivering some of these required functionalities, but do not cover the essential functionality that is illustrated in Figure 5.28.

An analysis of the experience from production Grid interoperability use cases reveals that support for HPC-based job application descriptions can be refined when using resources available within HPC-driven infrastructures (cf. Definition 16). The next paragraphs thus address the requirements raised in Definition 63. It should be noted that the basic functionality (specifying CPU, core, or memory requirements) for submitting HPC and parallel jobs is provided by using OGSA-BES [169] together with JSDL [115] profiles such as the SPMD profile [281]. But an overview of the proposed refinements in addition to these basic specifications is provided

Functionality Extensions and Improvements	**Area**	**Extended Standard**
(a) Network topology (torus, global tree, Ethernet, etc.)	Compute	JSDL
(b) Shape reservation (X x Y x Z)	Compute	JSDL
(c) Network information enhancements	Info	GLUE2
(d) Available shape characteristics	Info	GLUE2
(e) High message support	Info	GLUE2
(f) Task/Core mapping definition	Compute	JSDL
(g) Available task/core mappings	Info	GLUE2

Table 5.14: Functionality improvements of the high performance computing extensions.

in Table 5.14 with a particular focus on those extensions that have been necessary to conduct the three case studies WISDOM, VPH, and EUFORIA.

Table 5.14 and this section reveals that many concepts are influenced by large-scale systems specifically provided by one vendor that is IBM that can be at least partly explained by the fact that the majority of TOP500 listed large-scale HPC systems are IBM machines. Some concepts such as the *'3-d torus networks'* are also provided by other vendors such as CRAY in machines like the CRAY XE6 [6].

Also, the three accompanying case studies in this thesis namely WISDOM, VPH, and EUFORIA have been all mostly used with IBM machines that also explains why HPC extensions for production e-Science application improvements are mostly related to features IBM machines offer. Nevertheless, given the widespread use of the IBM systems (e.g. BlueGene/P systems [290]) and installations in supercomputer centers around the world (e.g. IBM systems in DEISA/PRACE and beyond), the concepts revealed here, although dominated by a particular vendor IBM, have still a major impact beyond one particular Grid site.

Also it is expected that towards Exascale more and more low- to medium-scale HPC systems will appear that can take advantage of the concrete examples provided in this section. It is not the focus of this thesis to create and present any possible abstraction of lower level HPC machine concepts from all vendors around the world (e.g. CRAY, Fujitsu, etc.). Instead, 5.15 provides some abstract examples where similiar approaches can be taken based on architectures of other large-scale HPC systems. In addition, the plethora of compilers and approaches of the more and more emerging Graphical Processing Unit (GPU) [208] technologies bear also potentials for further abstractions so that the specifications used in e-Science infrastructures can be used with those cutting edge technologies as well.

All in all, the aforementioned abstractions and those listed in Table 5.15 for example are not hard to define and should be tested with e-Science applications in parallel of standardization efforts. But the plethora of systems available using these cutting edge concepts require standardized access methods also on higher levels such as Grid specifications to promote their use also on e-Science infrastructures (without requiring shell tools). With implementing the concept in this section, the uptake of the broader concept of e-Science infrastructures can be fundamentally supported offering and exposing HPC extensions on the infrastructure level via Grid middleware (cf. Definition 12).

Hence, this thesis thus provides only very concrete examples of the used concepts in the thesis case studies outlining an approach for other activities that have to ensure a broader coverage of other existing systems in the field. Firstly, the concepts presented here is given as an input to the standardization activities in PGI [270] and the standardization process in the corresponding group (e.g. JSDL) need to ensure a greater coverage of existing systems in the field. Secondly, the reference model adoptions in concrete architecture deployments can create

Machine Example	Vendor-specific Concept Short description
CRAY XE6 [6] (CRAY)	Machine specifically supports two vendor-specific modes [7] that could be abstracted as the IBM concepts: (1) Extreme Scalability Mode (ESM) for application-specific performance tuning and scaling; (2) Cluster Compatibility Mode (CCM) for ISV out-of-the-box applications;
K Supercomputer [32] (Fujitsu)	Machine specifically supports vendor-specific TOFU interconnect [33] that could be abstracted as the IBM concepts: 6-dimensional Mesh/Torus Topology Network Technology;

Table 5.15: Other potential large-scale HPC feature examples for similiar abstractions.

similiar HPC extensions using the here described overall approach and bring them back to the standardization groups for a further update on state-of-the art systems.

One necessary concrete extension to JSDL as part of the case studies is the support for different types of *(a) network topologies*. The choice of network connections can have a big influence on the performance of applications that make use of parallel programming models (e.g. MPI). To provide an example, the state-of-the-art BlueGene/P HPC system as shown in Figure 5.28 offers three different types of network connections. These are *'three dimensional torus'* [290], as illustrated in Figure 5.29, but also *'global tree (collective network)'* [290], and *'10 Gigabit Ethernet (functional network)'* [290]. Which one is actually used is often dependent on the type of application, and as such depends on the description of the application by the e-Scientists (cf. Definition 10) themselves.

Another extension to JSDL that is necessary to efficiently run parallel programming applications on current HPC-based Grid resources is the *(b) shape reservation* functionality as supported by current BlueGene/P systems [290]. The optimal shape for an application depends on the communication pattern of the MPI-based code, and thus it is application-specific and in turn should be part of the job description. A shape is indicated with *'X x Y x Z'* where X, Y, and Z are positive integers that indicate the number of partitions in the X-direction, Y-direction, and Z-direction of the requested job shape.

The above described required extensions to JSDL are also added as extensions to the GLUE2 specification and then re-used in JSDL. The GLUE2 *'NetworkInfo_t'* data type [113] already describes network information, but is limited to a few certain values and do not cover technologies that offer torus networks, or collective networks (e.g. global trees). The GLUE2 standard is thus augmented with corresponding *(c) network information enhancements*.

One specific example for a three dimensional torus network extension to GLUE2 is illustrated in Figure 5.29. An abstraction of functionality from the large-scale HPC resource is created that forms later standardized information as part of GLUE2. The overall idea is thus not restricted to those listed as part of this section since there are several features that can be exposed to end-users using this approach. Many of them represent the benefit of enabling a more efficient computation of e-Science applications.

Such information about Grid resources is exposed more accurately and thus also *(d) available shape characteristics* are added as extensions to the GLUE2 standard. An example of these extensions are illustrated in Figure 5.28 where ΔW indicates the GLUE2 extensions and ΔU marks the JSDL extensions.

In the context of the GLUE2-based description of Grid resources, important messages of the day (i.e. high messages) are added as extension to the GLUE2 standard addressing the requirement of Definition 64. GLUE2 provides attributes about DownTime information (i.e. service availability) or a more general possibility to link to certain information on the Web via the

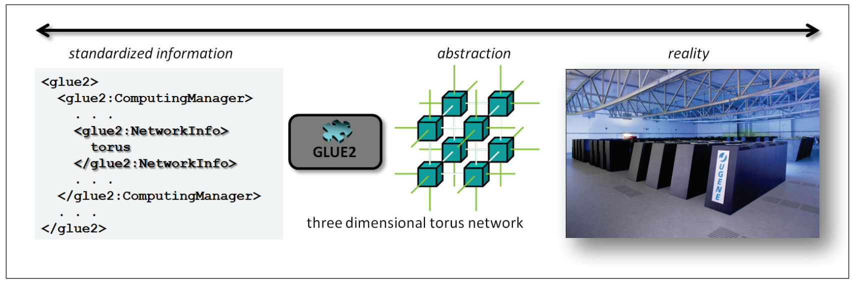


Figure 5.29: HPC extensions example with a three dimensional torus network of HPC resources.

StatusInfo attribute (as part of the *Service Entity* GLUE2 element), or another completely general *OtherInfo* attribute. Examples of these messages include temporary important information about file system usage (e.g. directory movements within a shared filesystem) or about certain changes in complex compiler configurations. High messages also inform the user about local storage situation changes (i.e. local storage cluster access) or other administrative pieces of information such as the transition period from one HPC-driven Grid resource to another (e.g. general account information). Because of its general applicability and major importance, the

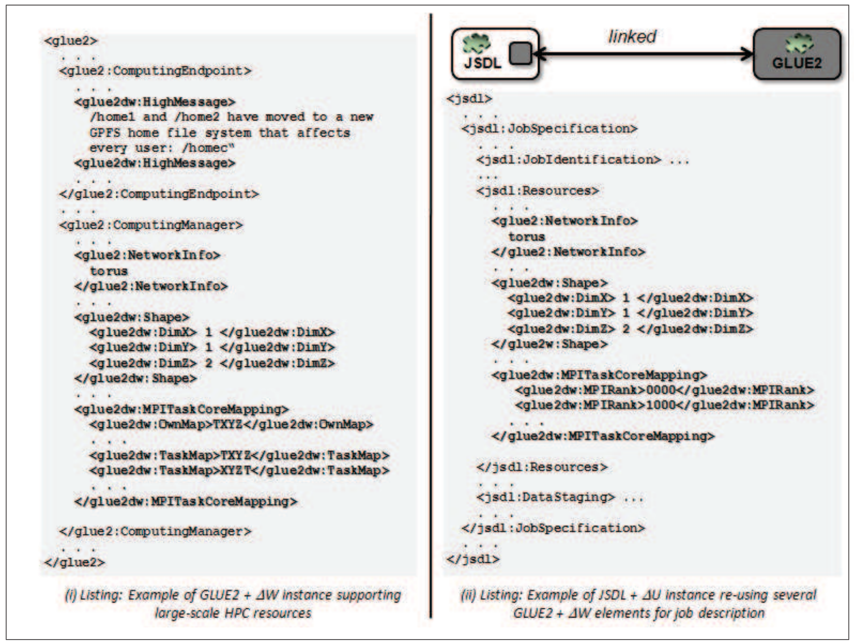


Figure 5.30: Design layout for the HPC extensions concepts in JSDL and GLUE2 with refinements.

```
Begin
  Begin GridInformationProvisioning
    Grid Information Provider of (RES) sends (GLUE2DW list A):
      listelement[0] = {available network topologies}
      listelement[1] = {possible shape setups}
      listelement[2] = {messages of the day}
      listelement[3] = {available task/core mappings}
    Grid Information Service updates (GLUE2DW Instance Entity)
      using (GLUE2DW list A) obtained from Grid Information Provider
    Grid Information Service exposes (GLUE2DW Instance Entity)
  End

  // not shown in Figure; choose of Grid resource might be manually performed
  Begin Brokering
    End-user uses scientific gateway and performs application setup
      defines GLUE2DW-based network requirements (NR)
    Meta-scheduler or broker query Information Service
      tries to match available Grid resources that satisfy (NR)
    Meta-scheduler or broker provide best suitable candidate resource (RES)
  End

  Begin JSDLCreationAndSubmit
    End-user of Scientific Gateway defines application via GUI
      using a JSDL creation entity that creates (JSDL doc B)
    JSDL creation entity re-uses GLUE2DW elements of information service
      specifying (NR) and also suitable shape setup (SS) in the (JSDL doc B)
    Scientific Gateway submits (JSDL doc B) with GLUE2DW elements to (RES)
      using Web service communication and the OGSA-BES standard
  End

  Begin JSDLParsingAndForwarding
    OGSA-BES instance of (RES) gets (JSDL doc B) and parses it
      managing execution according to specified (NR) and shape setups (SS)
    OGSA-BES instance forwards and manages the job via the RMS of (RES)
  End
End
```

Previously unsupported:
The information provider sends information with the GLUE2 + ΔW elements about available network topologies, shape setups or task/core mappings. The Information service exposes it with a GLUE2 + ΔW Instance

Previously unsupported:
The standardized network requirements resource description enables a much better chance of finding Grid resources that satisfy them and is less error-prone

Previously unsupported:
Re-using the GLUE2 + ΔW elements in JSDL preserves and assures the same semantics of the elements that are initially provided and exposed by the Grid resource itself

Previously unsupported:
In using the standardized shape setup information and network requirements the OGSA-BES instance can better prepare the job execution handled via the RMS.

(i) Listing: Pseudo-code of basic reference model algorithm that illustrates previously unsupported concept elements

Figure 5.31: Pseudo-code using the high performance computing extension concepts.

(e) high message support is added as a dedicated element to GLUE2. One example is provided in Figure 5.30 using the *glue2dw:HighMessage* element.

When working with HPC applications, it was also identified that GLUE2 and JSDL lacks support with respect to the supported MPI task and core mapping [290], thus addressing the requirement in Definition 63. The reason for this is that in practice such defined mapping can have a significant impact on performance and scaling of many e-Science applications running on HPC resources.

The analysis of the working practice of e-Scientist specifically working with HPC resources reveals that they often optimize their code using the low-level features of computational resources. While some even use assembler optimizations that is not addressed in the thesis, there is merit to understand the following concept that can be abstracted in a meaningful way similiar as our three dimensional torus example shown in Figure 5.29. One example is a domain composition of an 2D image analysis application where the image is divided into 8x8 tiles and in each process these tiles need to communicate with their 8 neighbours (1 process left & right; 3 processes top & bottom). The challenge in this example of parallelizing image processing tasks is the proper domain decomposition that specifies how the task is distributed among the various processors/cores available. This becomes even more complex when performing appropriate 3D image analysis domain decompositions.

Using concrete *(f) task/core mapping definition* reduce communication overheads and balancing the load (i.e. MPI tasks) across the different processors (i.e. cores). Applications mostly use the MPI cartesian grid communicator [241] definitions that are ideal for nearest-neighbour communications, but are also architecture independent, and work best with hardware-level support. Figure 5.30 describes one example with the *glue2dw:MPIRank* element.

The description of the task core mapping is very complicated and thus more functionality and details are provided. Figure 5.30 shows two possibilities that the improvement offers. The first one is to use pre-defined mappings of the corresponding computing system that are often encoded via XYZ and T while X, Y, Z are the coordinates of the processors of the Grid resource and T stands for a core coordinate within a processor (e.g. T = 0,1,2,3 in a quad-core processor). These pre-defined *(g) available task/core mappings* are exposed via GLUE2 as shown in Figure 5.28, and there can be pre-configured multiple mappings like XYZT or TXYZ.

In addition to using the pre-defined mappings there is also the second option of using own defined mappings (i.e. *glue2dw:OwnMap*) that we also support by simply using a list of *glue2dw:MPIRank* elements in the corresponding resource description within JSDL as illustrated in Figure 5.30. In this context the order of the elements matter and thus indicate that the first MPI rank (i.e. process 0) is mapped to a particular Cartesian coordinate (e.g. 1,0,0), also including the mapping of one particular core of the processor addressed by this communicator.

The exact definition is very application-specific, but the general characteristic of the manual task/core map is also exposed using the *glue2dw:OwnMap* element with one particular encoding that re-uses the XYZ and T values (e.g. TXYZ). Possible mappings are permutations of the XYZT mappings as supported by the Grid resource or XML elements that contain user specific MPI topology information. Both significantly increase the performance and scalability of scientific applications and are thus required to be defined as part of the concepts. But although they are often machine-specific, the same or similiar machines with the same functionality are often available multiple times within interoperable e-Science infrastructures.

The pseudo-code in Figure 5.31 illustrates which algorithm aspects have previously not been supported. It describes which HPC aspects and refinements lead to a more efficient execution within the basic reference architecture algorithm as part of the run-time pattern (cf. Section 5.1.5).

In order to take into account the overall reference model design, the aforementioned functionality around HPC concepts represent a missing link between the JSDL and GLUE2 specifications. In terms of the design model layout, XML renderings can be defined for GLUE2 and re-use them within JSDL as illustrated in an example in Figure 5.30.

Finally, Table 5.16 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	**Addressed in which manner**
Definition 63 (State-of-the-art HPC Support)	HPC extensions accessible via core building blocks;
Definition 64 (High Message Exposure)	High message exposure via core building blocks;

Table 5.16: Addressed requirements for HPC extensions.

5.2.4 Sequence Support for Computational Jobs

The design provides functionality to submit and manage 'simple' computational jobs using the Grid execution management service (cf. Definition 49). The difference between Grid workflows and resource-orientated application sequences are introduced in this section pointing to another missing concept in this context.

The aim is to jointly support different types of application execution modes (i.e. serial, parallel) for one e-Science application. This is useful so that e-Scientists can conveniently use ref-

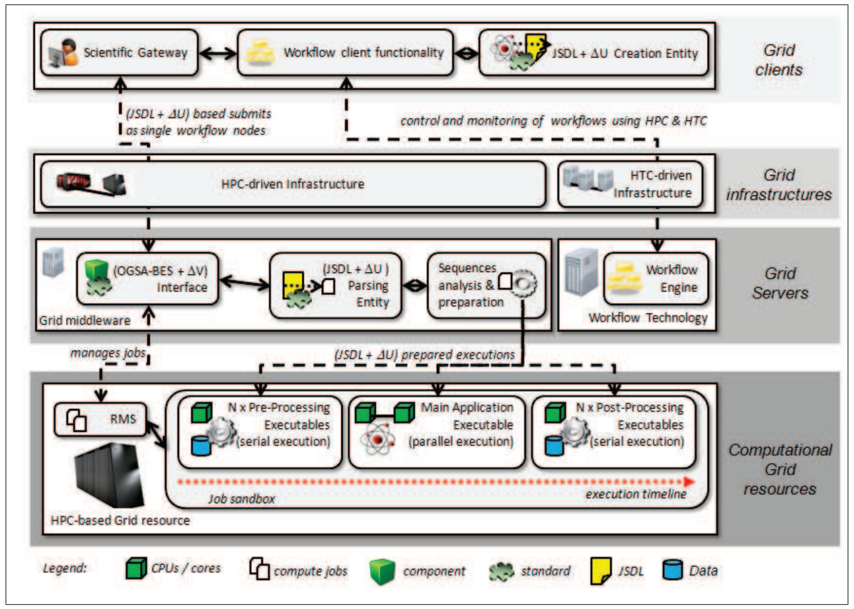


Figure 5.32: The application sequence execution concepts with reference architecture core building blocks.

erence model implementations that fit their needs in terms of remote compilation and pre- and post-processing functionalities required in Definition 65. This affects the core building block JSDL defined as part of Definition 49, including the following refinements. The requirements addressed on the architecture level are summarized as part of Table 5.18.

A thoroughly undertaken analysis of lessons learned (e.g. obtained from the WISDOM use case [259]) leads to specific missing features encountered during production Grid interoperability with respect to the support of automatically started pre- and post-processing functionalities within JSDL using different application execution modes.

Figure 5.32 illustrates one example of the molecular dynamics package AMBER [242] that consists of a set of applications, some of which are used to transform input data in a suitable format for production runs and/or transform outputs in several other formats necessary for further analysis. These transformations and short running pre-processing steps are often executed in a serial mode, while the actual corresponding AMBER molecular dynamic simulation (e.g. psander [242]) is executed in a parallel mode. In order to save time on rare HPC resources, the pre-processing steps that are serial can be executed before the actual parallel slots are reserved by the underlying RMS, but should share the same working directory of the application. The key idea is to support application sequence executions within one Grid sandbox supporting multiple types of application execution modes (i.e. serial, parallel).

Another analysis of lessons learned from production Grid interoperability efforts (e.g. VPH [263]) is the demand for remote compilation of source-code, thus avoiding the need to login manually with SSH and to locally compile the source-code. Many applications have to be installed beforehand on execution sites using SSH, since a suitable support for remote compi-

Functionality Extensions and Improvements	Area	Extended Standard
(a) Pre-job sequences (pre-processing, compilation)	Compute	JSDL
(b) Post-job sequences	Compute	JSDL

Table 5.17: Functionality improvements for the sequence support concept.

lation is missing in JSDL and in the production Grid middleware adoptions.

This approach is basically feasible when the source-code of the application is reasonable stable, but the lack of remote compilation becomes a real challenge when the source-code of e-Science applications are subject to change as often in HPC-driven e-Science infrastructures (cf. Definition 16). An overview of the refinements in this section is provided in Table 5.17. The ideas are published in [261], while a descriptive example is illustrated in Figure 5.33.

In the context of the above described obstacles, the differences between Grid workflows and sequences are important to consider. It makes sense that compilation and execution are performed in one Grid working directory (aka job sandbox). Otherwise the application is compiled in one workflow step and then the compiled executable needs to be transferred to another working dirctory to be executed as another workflow step. There is no exact boundary and one can realise the described approach with both workflows and sequences. But when using the proposed sequences unnecessary data-transfers are avoided between different job sandboxes and the necessary configuration activities specifying the locations of the source-code do not need to be repeated. In many cases, the source-codes are specific to some types of architecture, which in terms of many-core is even more and more evolving. More and more end-users require compilation prior to production runs, especially with HPC-based e-Science applications linking several libraries.

As a consequence of the afore-discussed points, JSDL is extended with the capabilities to execute *(a) pre-job sequences*. This concept enables the definition of n pre-processing applications that are serially executed before the main Grid job application. This also satisfies the demand for remote compilation since one or many of these pre-processing applications defined in the pre-job sequence can be used as compiling activity. In turn, this compilation sequence step is serially executed before the main freshly compiled Grid job application is started.

In an analogy to the pre-job sequences, the proposed improvements also cover *(b) post-job sequences* in order to support n post-processing applications. This sequence is started when the main (often in parallel executed) Grid job application is finished. Examples of both functionality extensions are shown in Figure 5.33, illustrating the *jsdldu:PreJobSequence* as well as the *jsdldu:PostJobSequence* elements. As shown in this illustration, it is possible to create XML-based renderings of the aforementioned concepts to augment the JSDL standard with these functionalities.

An overview of the concepts described in this section is provided with the pseudo-code in Figure 5.34 that illustrates the basic reference architecture algorithm (cf. Section 5.1.5) in the context of features that were previously not supported.

Finally, Table 5.18 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	Addressed in which manner
Definition 65 (Computational Job Sequences)	Sequence support via core building blocks;

Table 5.18: Addressed requirements for the sequence support.

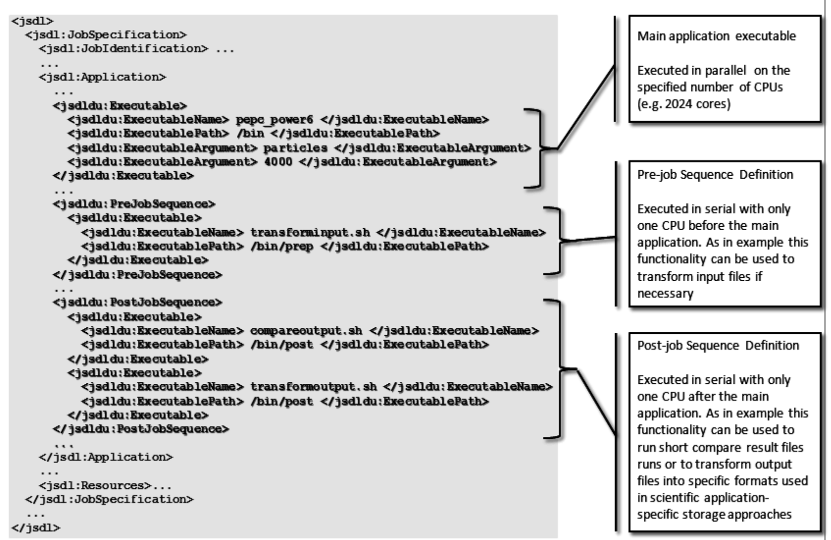
5.2.5 Manual Data-staging Concepts

The concepts described in this section are based on the core building blocks OGSA-BES and its implied JSDL of the reference architecture. The traditional *'automatic data-staging'* concept of JSDL [115] is often used in production e-Science infrastructures and a well-established link between the Grid execution management service (cf. Definition 49) and its implied Grid data transfer protocols is an important element of the reference architecture. The *'manual data-staging'* needs to be supported by the core building blocks of the reference architecture as well, because end-users require the flexibility to manually perform data-staging via their scientific clients.

This section thus address the requirement raised in Definition 66, and also Definition 67 since the concept inherently also covers how end-users get more required Grid job manipulation capabilities.

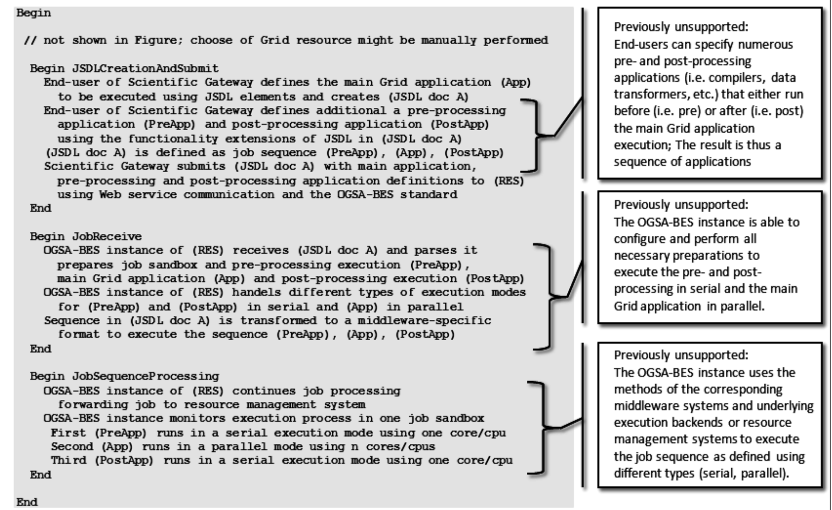
Especially larger scientific workflows and manual scientific investigation processes make it necessary to support the manual data-staging in addition to the already existing automatic data-staging. The requirements addressed on the architecture level are summarized as part of Table 5.20.

The analysis of lessons learned from production applications revealed that in many cases the end-users require more job control and more flexible data-staging functionality. They need to better coordinate distributed data and computation with their manual intervention in the process. This is true, irrespective of whether the data is transported to where the computational resource resides, or if computation is decomposed and job submissions are performed at the physical location of the data. In a wide variety of use cases (e.g. WISDOM [259]), the manual



(i) Listing: Example of JSDL + ΔU instance with application as well as pre- and post-job sequence definitions

Figure 5.33: Design layout for the application sequences concepts in JSDL and its refinements.



(i) Listing: Pseudo-code of basic reference model algorithm that illustrates the new job sequence concept

Figure 5.34: Pseudo-code using the sequence support concepts.

interaction of e-Scientists is necessary to carefully select data that is considered to be staged in or out at a given point in time in the overall process making use of human scientific expertise. An overview of the manual data-staging concept is given in Figure 5.35.

As published in [265], a fundamental drawback of OGSA-BES [169], JSDL [115], and the HPC FSP [305] is that they only support the so-called *'data-pull'* approach. That means end-users can specify data locations to be staged as part of the JSDL, and once submitted invoke execution logic within Grid middleware systems (cf. Definition 12) to automatically pull the data from the specified locations into the job working directory on behalf of the end-user. This approach has proved to be a successful method, thus it is also one important feature of the reference architecture.

But in addition to this approach, a rather client-initiated *'data-push'* approach enables end-users to manually perform data-staging by using manually tools like secure copy (SCP) [275] or features of SRM implementations. This latter approach is often motivated by manual scientific investigation processes performed by end-users within more complex scientific workflows (e.g. WISDOM [259]). In several cases, the expertise of e-Scientists (cf. Definition 10) is needed

Functionality Extensions and Improvements	Area	Extended Standard
(a) Pre-defined hold points	Compute	JSDL
(b) New hold states in addition to HPC FSP states	Compute	OGSA-BES
(c) Manual manipulation of job states	Compute	OGSA-BES
(d) Job sandbox location exposure	Compute	OGSA-BES

Table 5.19: Functionality improvements for the manual data-staging concept.

within the process that in turn requires manipulation of the job management progress (e.g. pause/continue) or more data-staging flexibility (e.g. intermediate result analysis).

While the *'data-pull'* approach is rather straightforward, the support of the *'data-push'* approach is not trivial. Multiple core building blocks of the reference architecture are involved and several improvements that affect the functional interfaces of OGSA-BES and also its internal state-model [169] of the execution process are needed. Hence, this requires an enhancement of our basic run-time pattern described in Section 5.1.5. An overview of these proposed extensions and refinements are presented in Table 5.19.

Manual data-staging-in/-out elements could be specified within the JSDL, but specifying the exact location by end-users would make the approach very unflexible. The use of JSDL elements would thus make no sense. In contrast, the proposed improvement named *(a) pre-defined holdpoints* within JSDL affects the whole execution process in the Grid middleware. Holdpoints influence the state model [169] of the middleware that implements OGSA-BES and JSDL. They can be specified similarly to *'supported states'* as defined in HPC FSP [305] (e.g. Running:stage-in, Running:stage-out, etc.). Such additional states influence the OGSA-BES core building block and the HPC FSP (i.e. JSDL profile) that is a viable basis. The state model is adopted, but augmented with *(b) new hold-states* as shown in Figure 5.36 thus extending the basic run-time pattern of the broader reference architecture as introduced in Section 5.1.5.

The benefit of using holdpoints is that they are more generally applicable and not only limited to data-staging activities while offering maximum flexibility to end-users. Figure 5.36 illustrates the *Running:Executing* state could also be used with a holdpoint, which in turn enables, for instance, the manual pre-processing of previously staged-in data. For example, before the

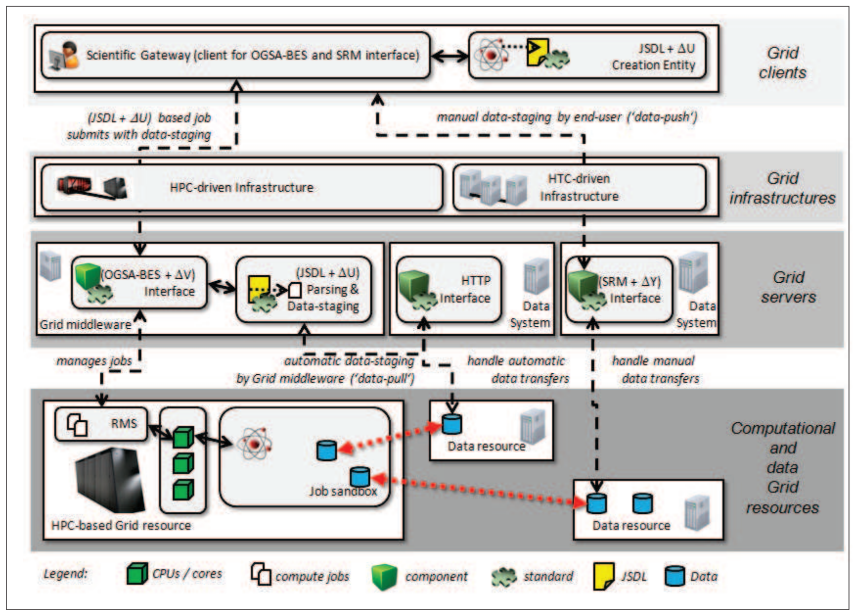
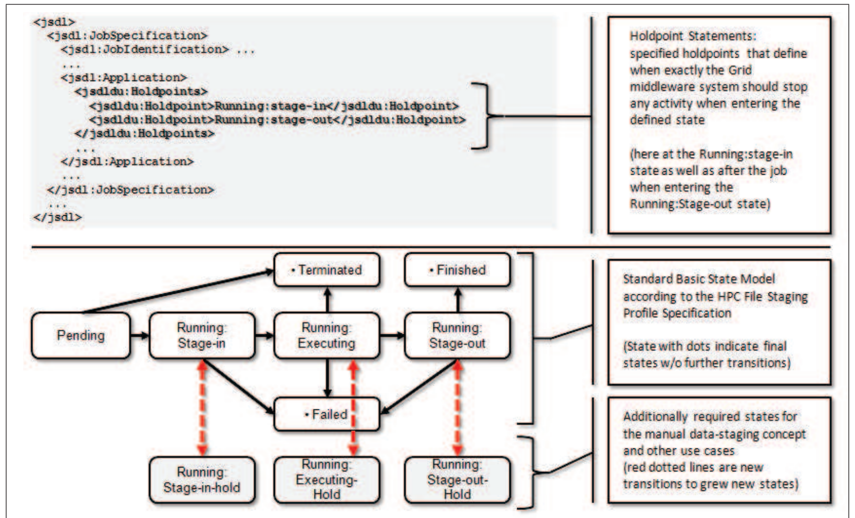


Figure 5.35: Manual Data-staging concepts with reference architecture core building blocks.

actual execution starts, the e-Scientists is able to check whether the right data has been staged or makes small manual modifications to data.

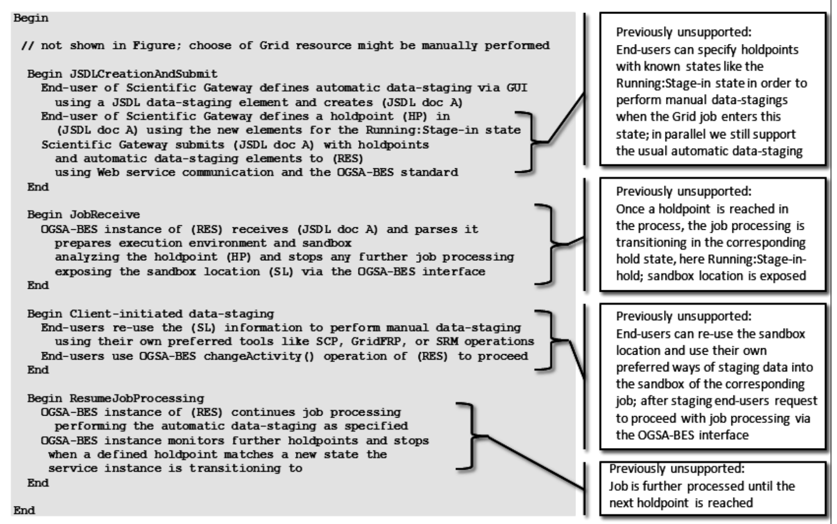
As illustrated in Figure 5.36, multiple hold states for each of the corresponding staging or executing states are proposed, because this enables much better feedback to end-user clients than just using one general hold state. Holdpoints are not considered to be *'breakpoints'* known from debugging tools. In contrast to breakpoints that interrupt the application execution itself, the holdpoints only interrupt the state transition process (cf. Figure 5.36) within the state model.

The counterpart of the JSDL holdpoint improvement is the *(c) manual manipulation of job states* which provides the functionality as part of a proposed OGSA-BES operation *changeActivity(desired state)*. This operation is required in order to resume job-processing after a defined holdpoint is reached. Instead of a dedicated *continue()* operation without a desired state parameter, a more general operation is proposed that enables the request of *'job suspends'* during the 'Running:Executing' state in order to address the requirements defined in Definition 67. Some RMS systems support the suspend feature and as such the WS-based interface within Grid middleware should offer the functionality in a well-specified manner rather than just being added to job management interfaces in a proprietary way. End-users are also able to use this additional OGSA-BES operation to request a transition into a corresponding hold state without specifying holdpoints in the JSDL. The transition is not always possible as requested, and as such concrete reference architecture implementations must ensure that only correct state transitions are possible. The general operation approach is also useful for further state model extensions that are applicable on-top of the reference architecture. Based upon the basic state model, only state transitions are allowed according to the state model in Figure 5.36 and any



(i) Listing: Example of JSDL + ΔU instance with holdpoints (top); (ii) State model with required hold states (bottom)

Figure 5.36: Design layout for the manual data-staging concept with basic state model extensions.



(i) Listing: Pseudo-code of basic reference model algorithm that illustrates the new manual data-staging concept

Figure 5.37: Pseudo-code using the manual data-staging concepts.

other requested state transition must fail within Grid middleware that adopts the reference architecture.

Closer investigation reveals that for the support of manual data-staging as defined in Definition 66 another improvement is needed named as *(d) job sandbox location exposure*. End-users require the exact location where the job is running (i.e. the job sandbox location) in order to stage data manually into the particular job sandbox location as shown in Figure 5.35. There are two ways in which the exposure is supported.

First, the sandbox location should be directly provided in the response of a corresponding *createActivity(JSDL)* of OGSA-BES. This requires a fast parsing of the submitted JSDL document and a fast processing of all consecutive actions such as creating the job sandbox, that is particularly unlikely for brokers that implement the reference architecture and simply forward submitted JSDLs to underlying computing services. Therefore, as a second method, the sandbox location must be exposed as soon as available together with other information about the Grid job (e.g. current status). This is not part of the OGSA-BES interface functionality but part of the information model GLUE2 by exposing information about this particular service.

Security aspects also matter when performing manual data stagings. When using Grid middleware features like the OGSA-BES-based job submission, end-users use their Grid credentials, that are X.509 certificates. This identity is then further used in automatic data-staging functionality leading to the use of delegation mechanisms like using X.509 proxies [300] explained as part of the reference architecture security pattern in Section 5.1.6.

In terms of manual data-staging, it is also expected that Grid credentials are re-used like using an X.509 identity to SSH [275] into the exact sandbox location in order to manually perform data-stagings. The SSH on the corresponding system should be properly configured to enable this since the interactive access to Grid resources in detail is out of scope of the reference architecture design layout. Different security mechanisms are used like typical xlogins with username and password, or SSH into the sandbox location, or the use of File Transfer Protocol (FTP) [275] for data-transfer. The reference architecture does not mandate any solution, but the Grid single sign-on feature [170] must be maintained whatever security setup is used by reference architecture adoptions.

The proposed concepts can be formulated as XML renderings of the improvements and functionality extensions. The pseudo-code in Figure 5.37 highlights the basic reference architecture algorithm (cf. Section 5.1.5) with aspects which have been previously not been supported.

Finally, Table 5.20 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	**Addressed in which manner**
Definition 66 (Manual Data-staging Mechanism)	Manual data-staging via core building blocks;
Definition 67 (Grid Job Manipulation Functionality)	Improved Grid job control via core building blocks;

Table 5.20: Addressed requirements for the manual data-staging support.

5.2.6 Enhanced Accounting and Data Management Concepts

The concepts described in this section are based on the core building blocks UR, SRM, and WS-DAIS of the reference architecture that are all open standards. But these standards also require several refinements that are presented as part of this section. The particular requirements addressed on the architecture level are summarized as part of Table 5.22.

The UR [216] is a standard to track computational resource usage. In [177], the specification is used to integrate the LLView resource monitoring tool with UNICORE. These results have been presented at the German e-Science conference in Baden-Baden in 2007, also mentioning the fact that more detailed computational information is needed in order to correctly track the resource usage of large-scale HPC resources and to correctly use the UR format with LLView. This input has been given at this time into OGF, but since then never found its way into a new UR specification despite several ongoing UR activities during OMII-Europe [69].

In OMII-Europe, work on WS-DAIS-based technologies like OGSA-DAI [117] and gLite components that offered SRM interfaces such as Castor [249] also pointed to necessary improvements. Most notably, the UR format should be extended towards the usage of storage information that is not covered by the computational-driven UR standard. The work in this section thus aims to address necessary extensions of requirement defined in Definition 56.

There is a also need to have more granularity in resource usage tracking. In a wide variety of use cases (e.g. EUFORIA [225]) and for work on invariants introduced in Section 5.1.4, more details about how infrastructure resources (cf. Definition 6) are used are needed. An overview of the proposed refinements are presented in Figure 5.38 with a particular focus on the UR specification. Resource tracking needs a common schema for computing and storage resources. A scalable way of exposing UR + ΔZ is an information service.

The previously introduced refinement concepts are all very much related to computation itself, in contrast, this section reveals some refinements from the data management and information area. The UR standard is traditionally in the field of accounting, but in this thesis

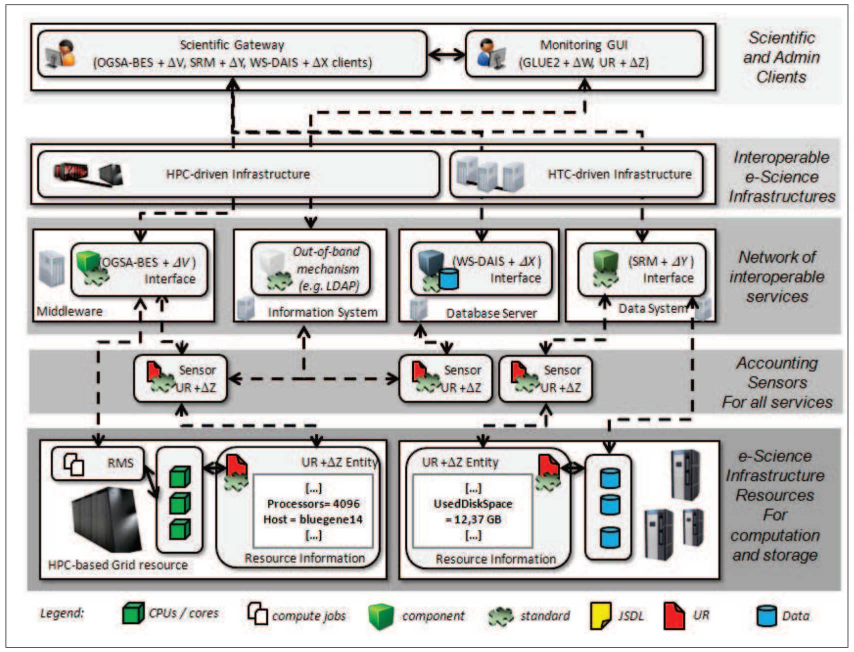


Figure 5.38: Enhanced accounting and data management of reference architecture core building blocks.

the UR is in the information area, because of its close relationships with information systems in general and GLUE2 in particular. The focus is truly on computation rather on the data management that is best reflected in the previous concepts as well. Nevertheless, Table 5.21 provides an overview of the functionality extensions addressing non compute core building blocks such as UR, SRM, WS-DAIS partly also re-using GLUE2.

The adoption of the UR specification as part of the OMII-Europe project included several HPC resources that was published in [177]. In addition to the precise description available in the UR specification [216], the academic analysis of UR usage on HPC resources such as supercomputers revealed that the UR needs a *(a) tracking of more computational resource details*. Although information is available via *Processors* or *NodeCount*, a more finely-granular resource

Functionality Extensions and Improvements	Area	Extended Standard
(a) Tracking of more computational resource details	Information	UR
(b) Tracking of VO information	Information	UR
(c) Re-use elements of GLUE2	Information	UR
(d) Tracking of storage resource details	Information	UR
(e) Clarified space tokens usage	Data	SRM
(f) More scalable query results	Data	WS-DAIS

Table 5.21: Functionality improvements of enhanced accounting and data management.


```xml
<glue2>
  . . .
  <glue2:ComputingEndpoint>
    . . .
  </glue2:ComputingEndpoint>

  . . .

  <glue2:ComputingManager>
    . . .
    <glue2:NetworkInfo>
      torus
    </glue2:NetworkInfo>
    . . .
    <glue2dw:Shape>
      <glue2dw:DimX> 1 </glue2dw:DimX>
      <glue2dw:DimY> 1 </glue2dw:DimY>
      <glue2dw:DimZ> 2 </glue2dw:DimZ>
    </glue2dw:Shape>
    . . .
    <glue2:PhysicalCPUs>
      1024
    </glue2:PhysicalCPUs>
    . . .
    <glue2dw:PhysicalCores>
      4
    </glue2dw:PhysicalCores>

  </glue2:ComputingManager>
  . . .
</glue2>
```

(i) Listing: Example of GLUE2 + ΔW instance supporting large-scale HPC resources

```xml
<urf>
  . . .
  <urf:JobUsageRecord>
    <urf:JobIdentity>
      <urf:LocalJobId>j41f.84777</urf:LocalJobId>
    </urf:JobIdentity>
    ...
    <urf:UserIdentity>
      <urf:LocalUserId>pool45</urf:LocalUserId>
      <urfdz:VOName>WISDOM</urfdz:VOName>
    <urf:UserIdentity>
    ...
    <urf:Processors>
      <glue2:PhysicalCPUs>1024
      </glue2:PhysicalCPUs>
      <glue2dw:PhysicalCores>4
      </glue2dw:PhysicalCores>
    </urf:Processors>
    ...
    <urf:NodeCount>4
    </urf:NodeCount>
    ...
    <urfdz:TasksPerNode>1
    </urfdz:TasksPerNode>
    ...
    <urf:Network>45</urf:Network>
    ...
    <urfdz:NetworkInfo>
      <glue2:NetworkInfo> torus
      </glue2:NetworkInfo>
    </urfdz:NetworkInfo>
    ...
  </urf:JobUsageRecord>
  . . .
</urf>
```

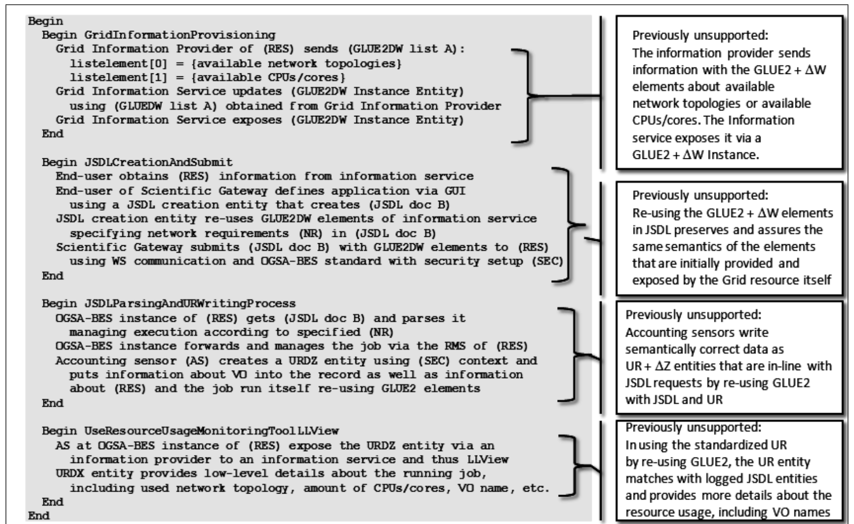
(ii) Listing: Example of UR + ΔZ instance re-using several GLUE2 + ΔW elements for usage tracking

Figure 5.39: Design layout for the enhanced accounting and data management concepts.

usage tracking is required such as the amount of used physical cores or tasks per node as shown in Figure 5.38.

In performing the thesis case studies between UNICORE in DEISA/PRACE and gLite in EGEE/EGI (e.g. WISDOM [259]), another limitation was identified named as *(b) tracking of VO information*. This is important since during the case studies many HTC resources, pool accounts are used that do not identify a specific user or VO and thus the UR entity needs more details than the existing *LocalUserId* element that is often used for the concrete account (at least on HPC resources). This information is obtained from the security credentials that are part of the security pattern (cf. Section 5.1.6). Both the SAML assertions and X.509 AC proxies give this detail as part of their security attributes about end-users. Grid middleware (cf. Definition 12) needs to take care of providing this information to a corresponding accounting sensor illustrated in Figure 5.38.

Another aspect of these particular studies is again another missing link between the specifications GLUE2 and UR where a lot of information in UR should be kept in sync with GLUE2 in order to maintain semantic interoperability. This improvement is named as *(c) re-use elements of GLUE2* within the UR (similiar to that already shown in JSDL). This not only avoids having error-prone translators and adapters between UR and GLUE2, but also supports the process of understanding whether requested resource requirements in a GLUE2 enhanced JSDL document have really been satisfied by checking the UR resource usage entity. One example of the concept behind the refinements is shown in Figure 5.39.



(i) Listing: Pseudo-code of basic reference model algorithm that illustrates previously unsupported concept elements

Figure 5.40: Pseudo-code using the enhanced accounting and data management concepts.

The same is true for the proposed UR extensions to support *(d) tracking of storage resource details* in addition to computational resource usage only. Also in this case a complete listing is out of scope, but Figure 5.39 aims to illustrate the key idea for compute transferred to the data domain re-using GLUE2 elements for storages [113] in a similiar way.

When working with SRM installations during OMII-Europe, academic analysis revealed that there is a need to *(e) clarify space tokens usage* since the SRM specification [286] leads to ambiguities. Hence, this indirectly addresses the requirement in Definition 50 since the disambiguities of the SRM specification was affecting the interoperable storing and retrieving of data across different SRM implementations. Adoptions (e.g. Castor [249], dCache [178]) implemented this concept in slightly different and non-interoperable manners. In the long-term this just needs to be clarified in the specification, but as part of the reference architecture work we adopt the *'WLCG profile'* [104] usage of it that has been given as an input to the SRM working group and is in discussions.

In the case of the WS-DAIS specification [9] and its implementation in OGSA-DAI [117], during OMII-Europe the demand for *(f) more scalable query results* have been identified. Hence, also this indirectly addresses the requirement in Definition 50 since the WS-DAIS specification points to limitations in their adoptions that are not implementation details, but are issues with the specification design. End-users couldn't really work with the WS-DAIS adoptions that have been fully standard compliant. The specification itself uses a concept that provides the results as part of one SOAP message that in turn is a scalability problem. The concept of the W3C WS-Enumeration specification [107] need to be added to the WS-DAIS specification in order to make query results more scalable (i.e. limit the number of returned datasets and iterate over them). Hence, we do not break the existing specification, but suggest to add a more scalable mechanism on top of the existing mechanism for results.

Also these lessons learned have been given as an input to the OGF WS-DAIS group several years ago. In the meanwhile it is therefore tackled in PGI while it is a second priority after working on PGI inputs to computational specifications (e.g. OGSA-BES 1.1).

In contrast to the work on UR, both latter refinements have been gathered together with experts from the data domain, but no publications are available for thise findings. Hence, this thesis does not take credit for these proposed extensions, but they are part of the reference architecture and refinements as they significantly support production usage. Nevertheless, the studies revealed that those refinements are still important to be included as part of the overall reference architecture in order to keep interoperability indicated with WS-DAIS + ΔX and SRM + ΔY in Figure 5.38. In order to take the overall reference model design and its associated reference architecture into account, several parts of the aforementioned functionality extensions are part of the missing link between the core building blocks GLUE2 and UR. Concrete XML renderings of GLUE2 can be defined that are re-used within UR as illustrated in the example in Figure 5.39.

The following pseudo-code in Figure 5.40 illustrates the algorithm aspects that had previously not been supported. It describes which refinement leads to a more accurate resource tracking within the aforementioned basic reference architecture algorithm as part of the runtime pattern (cf. Section 5.1.5) with a particular focus on computation. The resource usage tracking using UR entities for storage can be considered to be very similiar to using SRM services and WS-DAIS services that write UR + ΔZ by re-using GLUE2 elements from the storage such as *'StorageShare'* in the GLUE2 specification [113].

Finally, Table 5.22 summarizes the aforementioned improvements of reference architecture elements and provides an overview how the requirements of Chapter 4 are addressed on the reference architecture level.

Requirement Definition	**Addressed in which manner**
Definition 50 (Grid Data Management Service)	Clarifications of SRM core building block;
Definition 50 (Grid Data Management Service)	Improvements of WS-DAIS core building block;
Definition 56 (Grid Usage Record Format Schema)	Storage and compute extension to UR core building block;

Table 5.22: Addressed requirements for the accounting and data management concepts.

5.3 Seven Segment-based Process for Infrastructure Interoperability

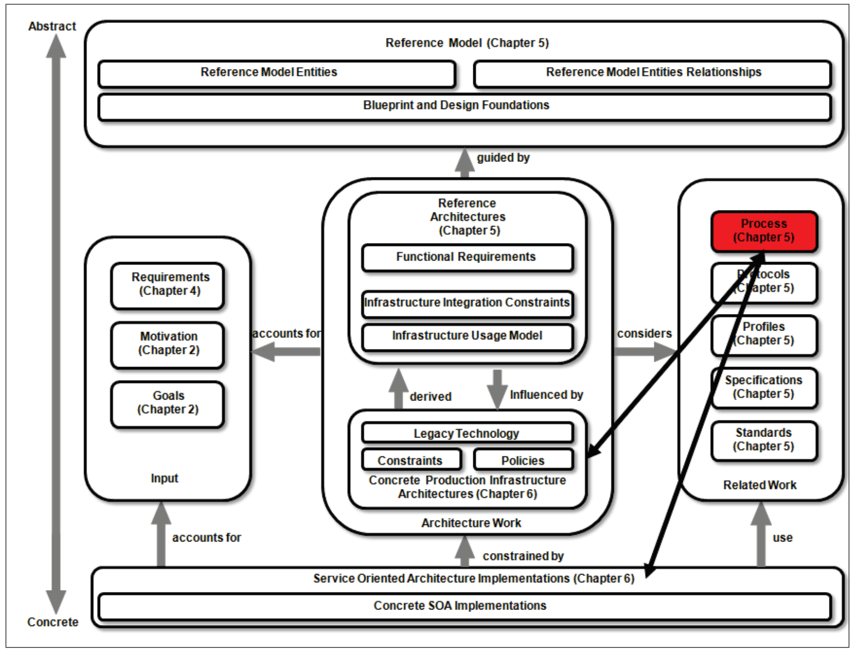


Figure 5.41: The reference model and architecture associated process.

The initial chapters described the state-of-the-art in e-Science infrastructures (cf. Definition 4) and also explained the open issues and challenges in having a reference model and associated architectural elements. This section describes more long-term activities as a process alongside the rather technical reference model and associated architecture work as shown in Figure 5.41. The requirements addressed in the architecture work are summarized as part of Table 5.23.

It seems to be obvious that those elements are based on open standards according to Definition 47. The benefit is to prevent transformation logic (cf. Definition 27) as concluded in Chapter 3. These findings are clearly relevant within the greater e-Science community today, and as a consequence more and more technologies that are relevant for production e-Science infrastructures (cf. Definition 5) adopt open standards (cf. Definition 14).

But at the same time the rate of interoperability is not increasing significantly, while in some cases once established interoperability between Grids is not sustainable. As a consequence, the majority of infrastructures are not interoperable (cf. Definition 19). One of the reason for this relies in the complexity of production e-Science infrastructure interoperability that consists of far more challenges than those which can be easily overcome by only technical work.

Adopting common open standards in technologies guided by a reference model is surely an approach as concluded from Chapter 3. But a reference architecture with open standards is only one step in the right direction, since it must be complemented with a whole process as required in Definition 78 and as shown in Figure 5.41.

Such a process needs to take the dynamics of change of the infrastructure environments (i.e. technologies) and governance (i.e. policies) into account, and more specifically several non-functional requirements raised in Section 4.2. This contributes to tackle one of the key challenges in e-Infrastructures referred to as *'Balance'* in [210].

The particular contribution of this sub-chapter is therefore the provision of some key segments of such a process that bears the potential to achieve and to sustain interoperability between production e-Science infrastructures on a long-term perspective. Such a process is published in [254], referring to seven distinct segments (often earlier named as steps) in order to provide a complementary guide to the reference model and its associated architecture work.

The process also addresses those aspects that go beyond pure technical issures and challenges. The process provides thus guidance to technology developers as well as infrastructure providers on how certain interoperability problems at different levels (technology, policy, etc.) needs to be tackled to be sustained. All the seven segments describe abstract mechanisms towards interoperability and thus are not intended to provide particular solutions optimised for specific components, architectures, or infrastructures. Figure 5.42 illustrates an overview of the seven segments that bear the potential to gradually increase the chance of interoperability.

The overview of the seven segments shown in Figure 5.42 clearly points to a lot of complementary activities in the field of e-Science Infrastructures and Research infrastructures as a whole. Many of these aspects mentioned in the segments are in-line with those proposed in various reports such as e-IRG papers, e-Infrastructure concertation meeting reports, or reports

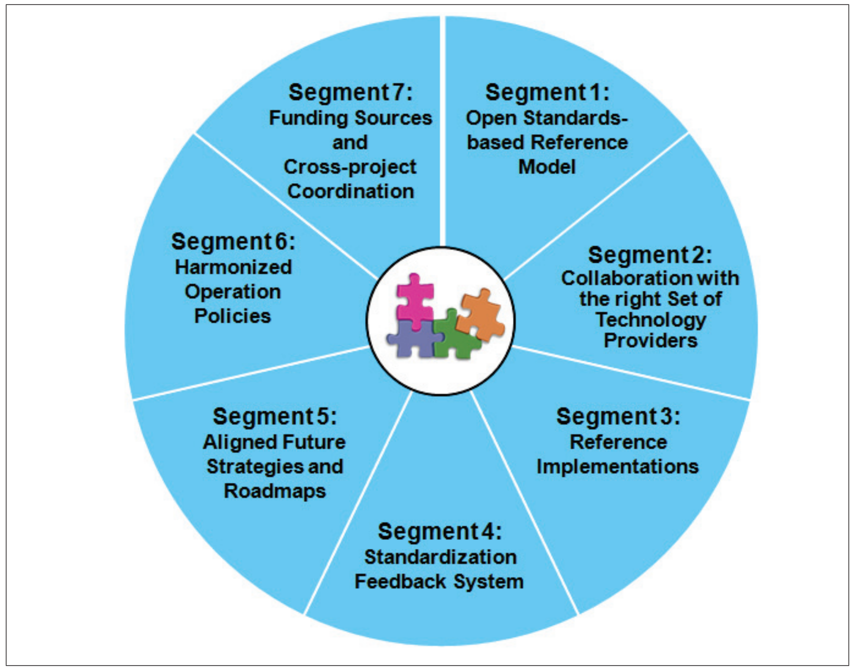


Figure 5.42: Overview of the seven segments towards production e-Science infrastructure interoperability.

Requirement Definition	Addressed in which manner
Definition 78 (Process for Sustained Infrastructure Interoperability)	Process defined with seven specific segments;

Table 5.23: Addressed requirements as associated elements to the architecture work.

of the European e-Infrastructure Forum (EEF) [23]. However, the scope of these reports often goes far beyond the problem space of this particular thesis. As a consequence, the process segments are not invented in this thesis and completely new. Instead, the segments represent a specific collection of *'many recommendations'* gathered from several of those inputs with the particular focus on 'e-Science infrastructure' interoperability. The goal is thus to provide a clear focussed guideline for those that would like to implement a path towards infrastructure interoperability in e-Science. Finally, Table 5.23 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

5.3.1 Segment 1: Open Standards-based Reference Model and Architecture

The first segment towards the interoperability of production e-Science infrastructures is to create a standards-based reference model. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.24. As previous sections of this chapter indicate this also includes the definition of associated architecture elements (e.g. reference architecture, patters, etc.) as defined in Definition 79. A reference model refers to a broad term that stands for an 'umbrella' and for a lot of associated architecture work, including standards, profiles, specifications, and others. In the context of production e-Science infrastructures, it is important that this reference model embodies the fundamental goal of an interoperable network of services (cf. Definition 15). The goal is that such a model can be looked up for various purposes and consists of a number of well-formed entities, relationships, and concepts that guides and provides the focus for a more concrete reference architecture and subsequent derived concrete architectures for many different infrastructures. The findings of Chapter 3 pointed to the fact that it is recommended in this step that the reference architectures are slim and production-oriented to have a chance to be used in practice (cf. TCP/IP vs. ISO/OSI). Another important key recommendation for the creation of the reference model is obtained from the European Informations, Communications, and Consumer Electronics Industry Technology Association (EICTA) white paper 2006 by maintaining the *'focus on interoperability requirements'* [158] rather on requirements that not necessarily are important for the interoperability of e-Science infrastructures (e.g. very unique capabilities).

The first key element of the reference architecture design approach is to use open standards (cf. Definition 14) in order to prevent transformation logic (cf. Definition 27). The use of standards should be inherent in the design for both services guided by reference model entities and their relationships with one another. The reference model should be established with an approach driven by *'interoperability by design'*. Nevertheless, from a strict technical perspective standards only can be referred on the architecture level guided by the overall reference model as illustrated in Figure 5.41. As such the term 'standard-based reference model' refers to the term 'reference model' again as an umbrella term that should be the outcome of this segment in context of the larger seven segment-based process.

With this approach, the aspect of the work of developers in creating objects, which behave precisely according to the reference model and reference architecture standards is made easier. The core building blocks of this reference architecture implementations, which are guided by the entities of the greater reference model, should use open standards where possible. The use of standards enables developers to re-use reference architecture implementation parts again, and also end-users can switch technologies more easily.

Even more important is the standards-based approach for achieving the goal of an interoperable network of Grid services realised by adopting the reference model architecture. By using open standards rather than proprietary interfaces, there is a greater probability that the majority of Grid technology providers adopt the standards of the reference architecture in order to obtain interoperability between services from different technology providers. This enables a *'strive for innovation and competition based on agreed standards'* [158] as recommended in the EICTA white paper 2006. Having these standards-based service implementations from different providers deployed on the wide variety of production e-Science infrastructures brings the vision of an interoperable network of Grid services closer to end-users. The overall reference model design and the reference architecture layout should be significantly driven by such a vision, as illustrated in Figure 5.43. This includes the creation of individual infrastructures (cf. Definition 20) defined on-demand by end-users according to their needs.

From the technical perspective this segment with the reference model and its associated reference architecture elements is by far the most important one compared to other segments. It even influences all other segments providing focus on those entities and relationships of an infrastructure architecture that need to be handled by subsequent segments. In-line with sw-engineering practices, the reference model approach and the use of standards need to work according to the principle of defining an architecture independent from its implementation.

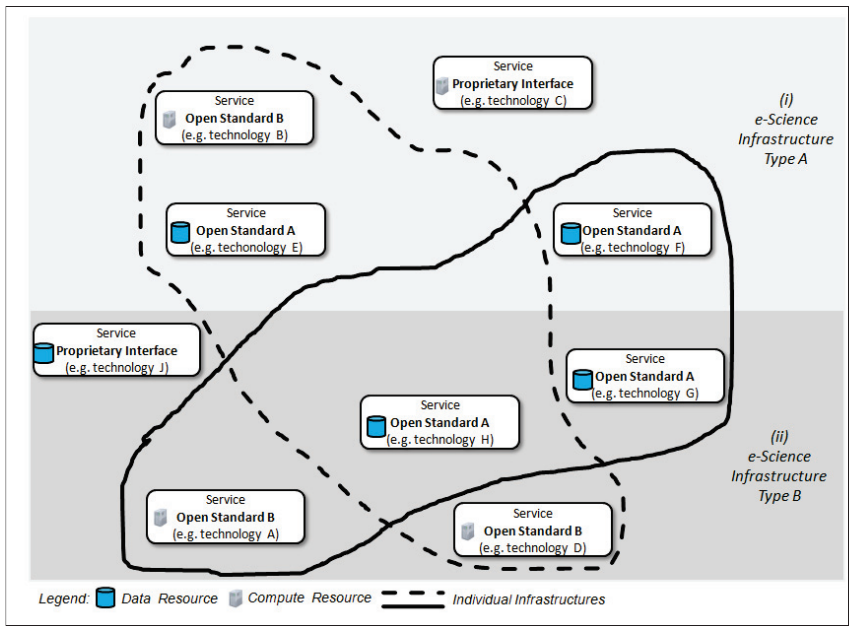


Figure 5.43: A network of interoperable Grid services enabled by a common reference model.

Evidence in the past in e-Science infrastructures shows that this is not always the case that also partly contributes to non-interoperable infrastructures existing today. For example, the EGEE project was at the same time also an infrastructure and technology provider (i.e. gLite) what makes it hard to define an architecture independent from its implementation.

Finally, it should be noted that several activities in e-IRG and at e-Infrastructure concertations meetings pointed to similiar activities, but the creation of a reference model as described in this segment one was not recommended until today. Instead, tables of project outcomes have been defined or larger frameworks that cover many EU projects, but still lack which tool (i.e. reference model) should be used to increase interoperability as a long-term process.

Finally, Table 5.24 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	Addressed in which manner
Definition 79 (Common Reference Model Creation)	Segment one recommending a standard-based reference model;

Table 5.24: Addressed requirements as part of process segment one.

5.3.2 Segment 2: Collaboration with the Right Set of Technology Providers

The second segment of the process towards interoperability of technologies is about influencing a significant fraction of the landscape of e-Science infrastructures with the right set of technology providers. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.25.

This segment recommends that the real impact towards interoperability especially in using the aforementioned segment about reference models and associated architecture recommendation can only be achieved with the closest possible collaboration among the key technology providers in the field, as defined in Definition 80. This idea is not new, and there are several activities in the field that aim to encourage collaborations by bringing together key players of the community.

One example is a meeting with *'150 key players of the e-Infrastructures community in order to take stock of the current developments and to discuss future actions'* [134]. These meetings are named as *'e-Infrastructure Concertation meetings'* organized by the European Commission (e.g. [132, 133, 134, 156] and present an excellent opportunity to enable collaboration. But the scope of this activity is rather high-level, e.g. *'to communicate the Commission's policy on e-Infrastructures and e-Science'* [134] or *'to discuss the future of directions of e-Infrastructures in Europe'* [134]. While both topics are very relevant, the lack of focus on particularly important aspects to interoperability distinguishes this segement from those meetings although some of them have been performed with a particular focus on standardisation (e.g. [132]). Hence, the recommendation of this particular event is to collaborate much more closer than only driven by event-based approaches.

Technology providers have different roadmaps, but also in many cases having overlapping interests in-line with the greater community. Although this segment seems like good scientific practice, and as such sounds pretty obvious, it is often neglected and its true power in getting to agreements is underestimated.

One of the key aspects in this segment is that such collaboration needs to be established as early as possible. When a group of technology providers in a specific field aims to create and adopt a specific reference architecture that promotes interoperability, it is wise to get as much of the technology providers in the field *'on board'* as soon as possible at the beginning. When a sub-fraction of known technology providers first creates a reference model and other technology providers join later, it is very unlikely that the majority of concepts of such a model is accepted without an enormous amount of discussion and clarification (if accepted at all).

But involving all the technology providers of relevant e-Science infrastructures in the process is not enough. Even more important and challenging is the mutual understanding of the providers during the process. Often, the different backgrounds and unique motivations of the technology providers in general and their corresponding roadmaps in particular lead to huge divergence of requirement collections. But in many cases, many of these requirements are duplicates since the mutual understanding about different terms and their semantics is a true challenge, while many different terms actually refer to the same *'fact'*. The aforementioned collaboration issue is another important aspect where communication elements among individuals come into play in this segment. In many cases the missing common semantics about terms (e.g. sandbox, workspace, job-space, etc.) and concepts (e.g. data-push and client-initiated data-staging, etc.) lead to situations where long discussions in the end reach the conclusion that all are talking about different things but referring to the same *'thing'*.

A very important part of this step towards interoperability is to understand the other technology providers and their background before being understood by them, which, in turn, is a key principle known from Covey [151]. Another aspect is collaborating with all technology providers early on in the agreement process about design issues. This segment towards interoperability cannot provide concrete detailed solutions in order to reach agreements.

Therefore, valuable methods known in literature provide good approaches such as the co-called *'principle negotiation concept'* from R. Fisher and W.L. Ury published in their book *'Getting to Yes'* [166]. The general guideline as part of this particular collaboration segment is surely to seek WIN/WIN situations [151]. It is better to adopt two concepts as part of the same standard in parallel (if possible) rather than leaving important concepts out of the specification and resulting in a standard that misses valuable concepts and because of this is not used at all. Another mechanism is voting with majority decisions that is a better solution than leaving important functionality out of the standard when only the minority is against it and leaves the standardization effort, but at least is then used by the majority of technology providers. There are also technology providers that are invited to a collaborative process (e.g. in SDO working groups), but hesitate to join the efforts from the beginning. But it is not to be underestimated that an agreement among the majority of technology providers in one specific field influences others. Such important agreements, especially on standards that are later used in practice, often also influence those hesitating technology providers.

Finally, Table 5.25 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 80 (Key Technology Providers Collaboration)	Segment two recommending collaboration with key technology provider;

Table 5.25: Addressed requirements as part of process segment two.

5.3.3 Segment 3: Reference Architecture Implementations

The third segment towards interoperability augments the theoretical consideration of a reference model with practical expertise of its associated reference architecture adoption within concrete technologies. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.26.

This segment is a complementary effort to the rather theoretical reference model design in general and the recommendation of using open standards (cf. Definition 14) in particular. The reference architecture implementations and early prototypes of the overall reference model entities, relationships, and concepts reveal many important practical insights that theoretical architectural blueprints often cannot provide. Hence, it makes sense to develop reference architecture implementations and prototypes of the major concepts during the design process as defined in Definition 81.

One practical example during this process is the implementation of emerging open standards that did not fully reach their specification status or that are refined towards a second version of them. Although sometimes being cumbersome and *'re-factorings'* can often be applied to them, the reference architecture implementation with prototypes provide clarity to theoretical thoughts. It thus also brings attention to the design process, avoiding drifts into overly theoretical approaches that are not useful in practice (cf. TCP/IP vs. ISO/OSI).

In the majority of cases, a full reference architecture implementation in parallel is basically not possible due to the lack of effort that needs to quickly adapt to numerous changes. Nevertheless, the concept prototypes created in parallel to the reference model design process and standardisation process is a necessary requirement to achieve real working production solutions. The 2006 white paper of the EICTA also offers one specific recommendation in this

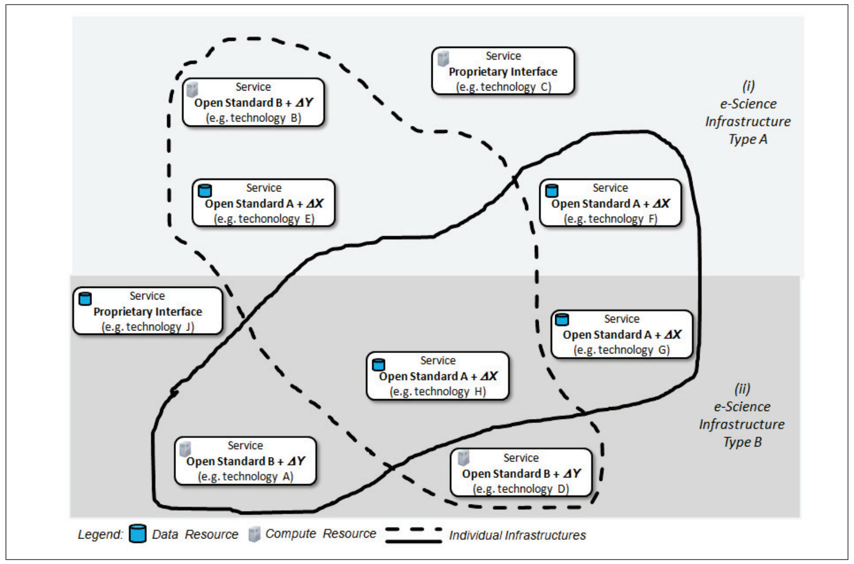


Figure 5.44: Reference architecture core building blocks refinements over time.

particular context for the standard development process: *'Consider making a range of prototype implementations and interoperability testing part of the standard development'* [158].

The reference architecture implementations of a reference model and its major concepts as well as its standard-based architecture lead to insights as to how open standards can be improved. Based on the experience of the early adoption, such standards also need to be refined with improvements as illustrated in Figure 5.44 in which refinements are indicated with Δ standing for additions to open standards. These refinements (cf. Definition 48) should be based on an analysis process, which takes many different existing approaches as well as lessons learned of experimental field studies into account, especially those gained from working with applications across infrastructures. Expected outputs of this analysis are a couple of non supported, but additional concepts, which have the potential to significantly improve the efficiency of applications across infrastructures, but do not break the standards themselves.

Finally, Table 5.26 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 81 (Reference Implementation Developments)	Segment three recommending reference implementations;

Table 5.26: Addressed requirements as part of process segment three.

5.3.4 Segment 4: Standardisation Feedback Ecosystem

After the short-term initial segments towards interoperability in the previous sections, this section addresses aspects on a medium to long-term perspective in order to reach sustained interoperability of reference architecture implementations. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.27.

This segment is also about the process of improving open standards, but goes beyond the previous segment. The idea is not new and also recommended in the EICTA white paper 2006 with *'Provide feedback channels for standards maintenance'* [158]. This segment provides concrete feedback to the standardisation process via such a channel by creating a whole *'group ecosystem'* with a production-oriented focus as raised in the requirement in Definition 82.

The implementation of the previous segment should improve the overall reference model design in general and should enable end-users with e-Science applications across infrastructures in particular to take advantage of initial interoperability setups. This early adoption work contribute to important lessons learned on how standards can be improved, but this is achieved in many different technical areas at the same time with using specifications together as a whole solution for a given problem (e.g. security with job submission). But that does not directly affect one specific standardisation working group and therefore the feedback channel is unclear. Investigations into the e-Science applications revealed as part of the case studies of this thesis (i.e. WISDOM, VPH, and EUFORIA) that in many cases the use of one standard is coupled with the use of another standard of a different technical area.

One practical example is the use of a computing standard within e-Science infrastructures that also implies the use of multiple particular security standards. These security standards do not directly specify how it should work with the specifications of the computational area. The other way around, computational area specifications also often argue that security is out of the scope. The corresponding working group responsibility of working on standards together in order to work on production feedback is not always clear. The lack of responsibility from such

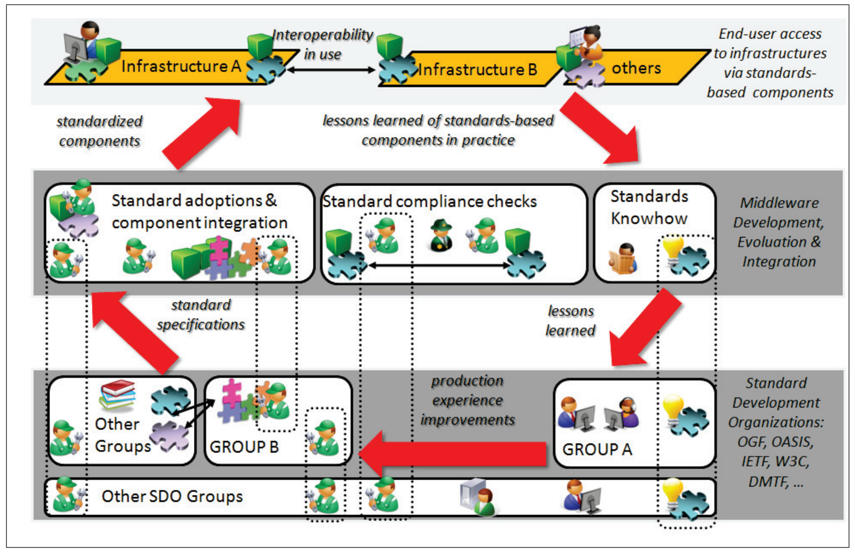


Figure 5.45: Standardization feedback system within an SDO with two complementary groups.

a dedicated group represents a hindrance to providing production feedback to existing open standards.

Based on these findings, the fundamental idea of this segment is to establish an overall *'standardisation feedback ecosystem'* that provide an appropriate channel back to the standardisation activities that in turn is not trivial to establish. The fundamental challenge of providing this feedback channel is that the use of standards in real production e-Science infrastructures (cf. Definition 5) is very complex, because one activity often covers the joint use of multiple standards at the same time to achieve certain functionality. One example is the job submission activity (i.e. job related standards) that in turn requires data-staging (i.e. data related standards) to and from the computational resource in a secure manner (i.e. security related standards). Hence, not every technical detail required for interoperability of technologies is specified by a single standard.

The joint use of standards in one particular activity in turn makes it difficult to channel all the different lessons learned about technology interoperability and end-user experience into one respective working group within one particular SDO. SDOs have working groups focussed on one dedicated concern (e.g. job management w/o security) or the aforementioned activities take advantage of standards from different SDOs like OGF (e.g. OGSA-BES) and OASIS (e.g. SAML) for example.

This thesis thus argues that SDOs need to establish two special kinds of overarching groups that channel experiences from the use of standards back to such single working groups. This is in particular also helpful, because one standard alone does not achieve necessary interoperability between production technologies that use a wide variety of different standards at the same time. Only a whole specified set of standards jointly used together across different technological areas (e.g. data, security, etc.) increases the interoperability in e-Science infrastructures.

This in turn generates lessons learned of how each individual standards can be improved for production runs.

The aforementioned two special groups are illustrated in Figure 5.45, named Group A and B. This figure shows in the upper part the infrastructure A and B that have already established initial interoperability through the use of (emerging) standard components. Using these standards-based components in practice generates a lot of lessons learned about standards that need to be properly fed back to the corresponding SDOs with a well-specified process.

In order to provide such as process, it makes sense to establish one dedicated group (here GROUP A) within one SDO that consists of members from both technology and infrastructure providers. GROUP A needs to drive interoperability activities within cross-technical areas using real e-Science applications with standards, thus collecting lessons learned from practical field studies until a critical mass has been gathered. Those relevant parts of the gained production experience is given to another dedicated standardisation group (here GROUP B) that is responsible for understanding the joint use of the standards in practice. GROUP B needs to break gathered improvements from GROUP A into chunks that can be processed by single-focussed standardisation groups where appropriate (e.g. computational and security standards). After the augmentation of these lessons learned into new specifications, their adoptions are likely to be deployed again on infrastructures, thus making the illustrated *'red cycle of the feedback ecosystem'* in Figure 5.45 complete.

Finally, Table 5.27 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 82 (Open Standard Evolution Process)	Segment four recommending a standardization feedback ecosystem;

Table 5.27: Addressed requirements as part of process segment four.

5.3.5 Segment 5: Aligned Future Strategies and Roadmaps

The fifth segment of the process towards interoperability is about the alignment of future strategies among different technology providers as defined in Definition 83. In many cases, interoperability could have been realised in e-Science environments by a common design when the corresponding technology providers would have worked together on a common roadmap rather than creating duplicate components over the years. One good example is the duplicate developments of Grid middleware technologies in order to access the RMS of a particular computational resource. The source-code and access paradigm to transfer jobs to the RMS (e.g. Torque) is identical in many middleware systems such as within the wide variety of UNICORE Target System Interfaces (TSIs) [293] and the Globus Gram component [167]. A good example thus in this context for this segment is the common roadmap from the European Middleware Initiative (EMI) project [27] where ARC RMS adapters have been considered to be re-used in the gLite middleware system. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.28.

This step goes one step further than the fourth segment about standardisation in terms of an even closer collaboration between technology providers than just within the SDOs. While in academically-driven e-Science this sounds relatively straightforward in the sense of collaboration among scientists, such a collaboration is problematic in the field of business where exposing future designs lead to risks in terms of time-to-market. In the scientific domain, however, the sharing of technologies and approaches plays a much more significant role that could lead to decreased maintenance costs and better sustainability options. The e-Concertation meetings of the commission aim at similiar benefits: *'The Concertation meetings were thus initiated by the Commission to be a bottom-up process for identifying commonalities and synergies between projects and the various national action and research initiatives, standardisation activities, etc.'*[134]. These meetings are good for an initial understanding of the scope and goals of different projects, while this segment can be considered as a next step where the opportunities for synergies is much more explored on the technical level.

Within the context of e-Science following the vision expressed in Definition 3, the demand to align whole technology roadmaps of component sets or future strategies on Grid developments is surely critical. This step is one step beyond working together in standardisation groups to define a couple of interfaces or protocols for components that are designed or already used in production. The benefit of this segment relies on taking advantage of the different, often complementary, experiences of the technology providers often gained by a special dedicated focus potentially kept over years.

This indirectly also contributes to the vision of a common e-Infrastructure and its benefits of *'leverage existing expertise and experience'* as described in the e-IRG Blue Paper 2010 [210]. There is no lack of innovation since new innovative concepts and prototypes can be still tested in testbeds or simulation environments, but should be kept away from production e-Science infrastructures until they become mature innovative technologies. The alignment of future strategies also enables better *'re-factoring strategies'* for harmonisation among the components deployed and used on e-Science infrastructures. Aligned roadmaps enable the use of components from one technology provider by different components of another technology provider

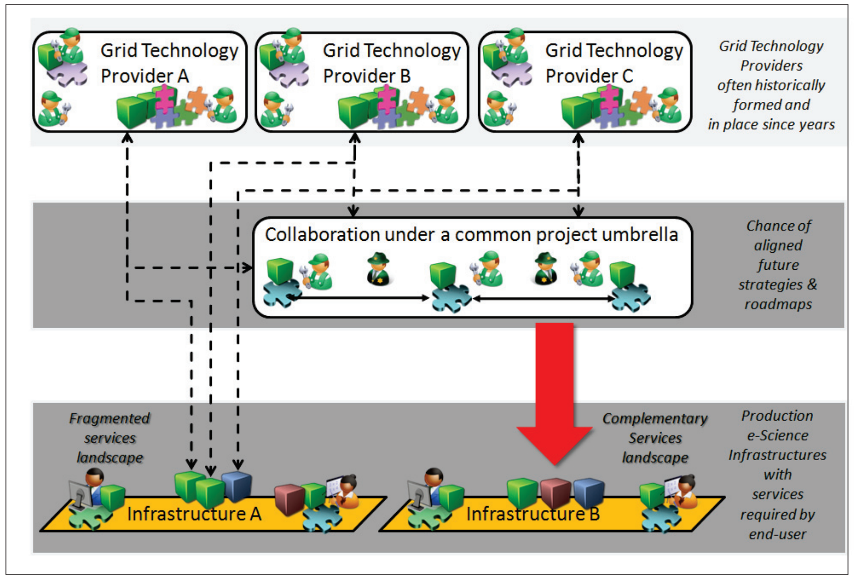


Figure 5.46: A common project that acts as an umbrella of different technology providers to harmonize activities.

and vice versa, avoiding duplicate developments thus finally reducing costs and maintenance efforts. This contributes to another benefit mentioned in the e-IRG Blue Paper 2010 towards a common e-Infrastructure *'avoiding unnecessary duplication in provision of ICT solutions'* [210].

One possible implementation of this segment is to create a joined project together with interested technology providers as illustrated in Figure 5.46. This project is then an umbrella of several distinct technologies used on various production e-Science infrastructures by different user groups. Under this project umbrella these technology providers closely collaborate in terms of aligning future strategies and technology roadmaps and thus jointly define the design of the next generation of technologies used on infrastructures.

The prioritisation of technology requirements based on end-users is important in this respect since innovation should be clearly not purely driven by technology innovations. This includes the balance of interests of technology providers that are interested in engaging in the fulfillment of end-user requirements. The work under such an umbrella can leverage the complementary experience of the different technology providers, thus leading to a fruitful collaboration even over decades to come.

Over years the technology providers have been in competition and such a project collaboration is a medium-term process that surely needs time to establish trust among them. Figure 5.46 summarises this segment with an illustration depicting how all discussed aspects fit together, namely infrastructures, technology providers, components, and common projects.

Finally, Table 5.28 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 83 (Future Strategy Sharing and Common Roadmap Definition)	Segment five recommending a alignment of future strategies;

Table 5.28: Addressed requirements as part of process segment five.

5.3.6 Segment 6: Harmonised Operation Policies

Another important factor of moving towards full interoperability between e-Science infrastructures are harmonised operational policies as defined in Definition 84. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.29.

It is important to understand the difference of what is possible in terms of technology interoperability and what is possible in terms of interoperability governed by policies within production e-Science infrastructures. Full operational interoperability can only be achieved when the technology-based interoperability is present, but where harmonized operation policies also do exist. The latter is a difficult topic, and where a concrete solution cannot be given here, but some elaboration of the problem is needed to raise the awareness that policy harmonisation is essential in order to achieve interoperable e-Science infrastructures (cf. Definition 21).

One recommendation of of this segment is the creation of dedicated policy groups that work on the harmonisation of operational polices on a regular basis in different areas (e.g. resource allocation, accounting, security, etc.). The aim of such a group is to collaborate towards the goal of seamless interoperation of production e-Science infrastructures by respecting the different requirements as best as possible. In many cases, such policy harmonisation is not a short-term achievement thus being rather long-term activities with a gradually increasing harmonisation over the years. Regular meetings are initially cumbersome but pay off after years of collaboration with increased seamless use of different production e-Science infrastructures by end-users.

This segment also touches negotiation elements that leave the field of computer science where methods from the *'Harvard Concept'* [166] are helpful to come to a consensus about policies. Such fields vary from one type of infrastructure to another, but certainly a common often relevant list of technology fields can in many cases be identified. Gained from experience over the years, such typical fields that require a harmonisation of policies are security (authentication, authorisation), accounting, and, most notably, resource allocation. Other topics of interest include the handling of security incidents in general as well as common user infrastructure access methods such as clients, portals, or dedicated science gateways.

Security and mutual trust issues are often the major showstopper in interoperability setups across different e-Science infrastructures, and as such this topic is a major topic of such groups. Agreements can be reached by the openness of decision-makers or executive board members, but in many cases operational policy agreements can be significantly supported by technology improvements and demonstrative use cases. Examples are the availability of common technology features (e.g. common accounting records) in different components but also the harmonisation of technologies to support a common interface. Other examples include additional features that are dedicated to enable technical interoperability but also solve issues at the policy level (e.g. finely-grained security features that enable flexible setups).

This segment addresses thus parts of the daily operation of e-Science infrastructures like other topics (e.g. support, etc.). It surely makes sense to consider a start even before production e-Science infrastructure starts its operation, or if it needs to be embedded into existing infrastructure ecosystems.

The harmonisation of policies is often time-consuming and the challenge relies not seldom on the mutual understanding of stakeholders. Even partial success in the harmonisation of operational policies is already a success since it would not be an optimal situation when technical interoperation would be possible, but the corresponding infrastructures disagree on a common operational policy. Common calls for applications organised from different infrastructures together could stimulate the aspects of this segment by having strong arguments by application end-users and their desire to have interoperable infrastructures instead of *'operational Grid islands'*.

Finally, Table 5.29 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 84 (Operation Policy Harmonisation)	Segment six recommending operation policy harmonisation;

Table 5.29: Addressed requirements as part of process segment six.

5.3.7 Segment 7: Funding Sources and Cross-Project Coordination

The final segment of the process towards a sustained interoperability of production e-Science infrastructures (cf. Definition 5) is the synergy of funding sources and cross-project coordination that are both tightly coupled in many ways, as also revealed in Definition 85. The requirements addressed alongside the architecture work as part of the process are summarized as part of Table 5.30.

This thesis is focussed on technical interoperability relevant to the scientific community and thus it only briefly outlines recommendations for funding-related topics. Often the reason for non-interoperable technologies deployed on production e-Science infrastructures relies on the support from different funding sources (e.g. US vs. EC). This is partly due to the fact of different non-cooperative projects funded by those different sources or even among projects with the same funding source.

The ideal situation would clearly be to have a joint source of funding that is not project grant-based (i.e. not time-limited) that in turn would enable more sustainable planning by technology providers. Being aware of its innovative need from a particular perspective, this thesis is not in the position to change these funding models but would like to raise awareness to potentially influence funding sources where possible to increase the levels of cooperation between funding sources and between funded projects. It is also clearly understood that research and its goal of researching innovative concepts for the society is the focus rather than sustainable operations across decades for a single technology space, while the latter also includes many innovative concepts over time. The desire to collaborate in a more structured way among the different funding sources is not new and this segment emphasizes on the following e-IRG recommendation: *'The e-IRG recommends that Global Collaboration should move from its ad-hoc character to a more structured and continued mode adequatly supported by the EC and national funding agencies;'* [206].

There is a need to acknowledge that cooperation is not always possible due to numerous constraints (e.g. organisational vs. project interests). But it makes sense to direct this segment to known issues and raise the awareness that full sustained interoperability of academically-driven e-Science infrastructures goes partly beyond the control areas of infrastructures and technology providers. Technology providers and more notably production e-Science infrastructures need to establish models of getting sustained either by national membership fees for a governing organization or industry collaborations where possible.

Today, partly interoperable production e-Science infrastructures already exist that are indispensable for the effective support of research even across its borders (i.e. emerging interoperability setups). *'Increasingly, the boundaries between local, national and international resources are blurring as researchers pool their resources to reach the resource scales required for their research'* [210]. This affects many areas like application enabling, user support, 24/7 operation monitoring, technology knowhow and dedicated deployments for particular e-Science communities.

Getting all these indispensable services sustained as a long-term strategy is one of the major parts of this final segment that naturally affects it more than perhaps any other segment. But this thesis can only highlight to work on this topic while a concrete solution is out of scope thus remaining a major challenge of the community. This critical challenge is best reflected by the follwing statement in the 6th e-Infrastructure concertation report: *'The long term sustainability of e-Infrastructures has emerged as perhaps the most hotly debated topic in the world of European e-Infrastructures'* [133]. Without sustainable infrastructure operations, its interoperability setups with other infrastructures become meaningless and efforts are not even be started knowing that some projects only have a limited life-time (i.e. trust issues). The aforementioned cooperation thus includes the exchange of ideas about such sustainability and relevant technology roadmaps on the one hand, but also the prioritisation of the use and joint adoption of common open standards on the other. This latter part ensures *'international interoperability'* among technologies from different funding sources by promoting the use of open standards (cf. Definition 14).

Apart from cooperation between funding sources, one concrete implementation option is to establish dedicated projects per technology area that addresses the aforementioned requirements and review progress along the lines of standardization and the adoption of standards. The European Commission used the e-Concertation meeting approach *'to review the recent progress made by leading standardisation bodies and to allow for the sharing of best-practices between e-Infrastructure projects (technical perspective)'* [133]. But this approach is only helpful as quick check or start of activities since after the meeting there are no efforts directly available to further facilitate the exchange of information for cross-projects work on a day to day basis.

In constrast, a dedicated project underpinned with funded efforts has much more the potential to facilitate cooperation across projects that are all funded for their unique strength and capabilities but in many cases also provide room for synergies and collaboration with other projects although not being its priority. Here it is important to understand that this cross-project aspect is different than segment 5 where one project was proposed that has the different technology providers as members, thus referring to intra-project collaboration.

In contrast, the inter-project approach should focus on common roadmaps with relevant projects making sure that a community view is adhered to in contrast to a single project-based roadmap. The commitment of other projects to such an aforementioned neutral project can be achieved by involving the funding source representatives in the process as close observers. One example of such a project was the OGF-Europe project [64].

Other mechanisms are cross-project deliverables and milestones that need to be provided in regular intervals to funding organisations.

Finally, Table 5.30 summarizes the aforementioned improvements of associated elements of the architecture work and provides an overview how the requirements of Chapter 4 are addressed.

Requirement Definition	**Addressed in which manner**
Definition 85 (Funding and Cross-Project Coordination)	Segment seven recommending cross-project coordination;

Table 5.30: Addressed requirements as part of process segment seven.

5.4 Conclusion

In this chapter it is described how the proposed reference model relates to various architectural inputs (i.e. standards, specifications) from the distributed systems domain in general and from the Grid domain in particular. The proposed IIRM addresses all the requirements of Chapter 4 taking into account the results in analysing a wide wide variety of lessons learned that have been surveyed in Chapter 3.

The overall design requires aspects on different levels, meaning entities and relationships that have been defined as part of the reference model level, while their more concrete, architectural core building blocks have been defined as part of the reference architecture level. As a consequence of the academic analysis in Chapter 3, the core building blocks are defined as (partly emerging) open standards meaning that all are well-specified and publicly available. Having every aspect of the architecture well-specified significantly increases the chance of interoperability of infrastructures and the possibility for any technology provider to provide implementations that give end-users the benefit of not having vendor locks. At the same time the evolution of standards and additional features are crucial to infrastructure production needs as well, and thus different refinements are addressed in order to address the field study experience. All these refinements are given into the standardization process and via the PGI working group it is ensured that all concepts influence the next evolutions of specifications of the core building blocks.

The design layout in terms of technical details is presented and certain patterns described that define more specific categories of the design such as the overall basic IIRM algorithm that promotes interoperability. Another conclusion from this chapter is that security is present on multiple levels and that the architectural design and security pattern address all those that are relevant for production e-Science infrastructures, whilst also acknowledging their slow migration and update cycles. Concrete architectures will arise from a combination of reference architectures, architectural patterns, and additional work such as the guidance given by the seven segmented process. While the reference architecture forms the basis of core building blocks for solutions, concrete architectures will define specific middleware adoptions of these core building blocks thus providing concrete solutions. Several concrete architectures from different scientific application domains are presented in the next chapter that are all significantly based on the findings of this more abstract design chapter. An important conclusion from this, in turn, is that the specification of the reference model design and its associated architecture is not implementation-specific, thus separating design from implementation being in-line with software engineering principles. Reference architecture invariants have been also defined that are often neglected when infrastructure integration activities are performed thus hindering seamless interoperability in many aspects. The reference architecture services are able to satisfy the requirements for a secure submission of Grid jobs across HTC and HPC resources including Grid storage.

Based on the findings of previous chapters, a complementary process to the reference model and its architecture elements is defined. This seven segment process bears the potential to increase interoperability between production infrastructures, specifically addressing those issues and challenges that cannot directly be covered by technical reference model approaches. These segments are basically known principles, but their collective power is often underestimated meaning that over the years only a few segments have been followed by infrastructure and technology providers. The final conclusion about the segments truly relies on the fact that their implementation will take time and that they are put in a sequential order from various different sources.

Chapter 6

Impact and e-Science Applications

In the initial chapters of this thesis, a model of the problem space was created that addresses challenges of interoperability between production e-Science infrastructures. After a comparison with existing solutions in Chapter 3 and the definition of requirements in Chapter 4, potential solutions are given in Chapter 5. The IIRM is presented as a technical solution alongside a non-technical associated process addressing long-term interoperability challenges. Based on these findings, this chapter makes important claims about the impact and applicability of the IIRM and its reference architecture design.

Nevertheless, there are known challenges in the e-Infrastructure community *'measuring the contribution of using the e-Infrastructure and its importance to the final science output'* [156]. Where possible, claims are underpinned with evidence from various sources such as existing production e-Science infrastructures (cf. Definition 5) as well as real e-Science use cases (i.e. WISDOM, VPH, EUFORIA) that take advantage of more than one infrastructure. Where possible, lessons learned from OGSA are taken into account and how the thesis findings influence the problem space highlighting thesis impact and contributions in several key sections.

In the first section, it is described how the seven segment-based process was implemented in context of the given problem space using concrete architectures of real production infrastructures in order to present its impact on e-Science. This chapter thus aims to explain why the reference model design is applicable in the given problem space, including examples of the achieved impact on existing production e-Science infrastructures. The recommendations incorporated in the seven segment-based process of Chapter 5 are discussed and information about their implementation alongside the creation of this thesis is given.

In the second section, the reasons for significance and thus impact for e-Scientists (cf. Definition 10) of the model is underpinned by architectures based on real production e-Science infrastructures. These architectures will be reviewed in the light of how they adopt significant parts of the reference architecture that is part of this thesis. Further reasons for the applicability and broad impact of the reference model are given in this chapter by highlighting roadmaps of key reference implementations of the European Middleware Initiative (EMI) [27] and the eXtreme Science and Engineering Discovery Environment (XSEDE) [29] projects. Signficant parts of the reference architecture are an integral part of both the EMI and XSEDE work plans thus influencing the landscape of e-Science infrastructures in Europe and US.

In the remaining sections, scientific case studies from different scientific domains are discussed in detail. These sections give insights how the case studies adopt the IIRM thus indicating that the thesis findings are applicable in e-Science leading to scientific innovation by using infrastructures as a *'commodity tool'*. The aim is to present how the thesis represents a benefit to e-Scientists in the bio-informatics, e-Health, and fusion domain, including insights on scientific activities emerging in various ESFRI projects.

6.1 Seven Segment-based Process Implementation and Impact

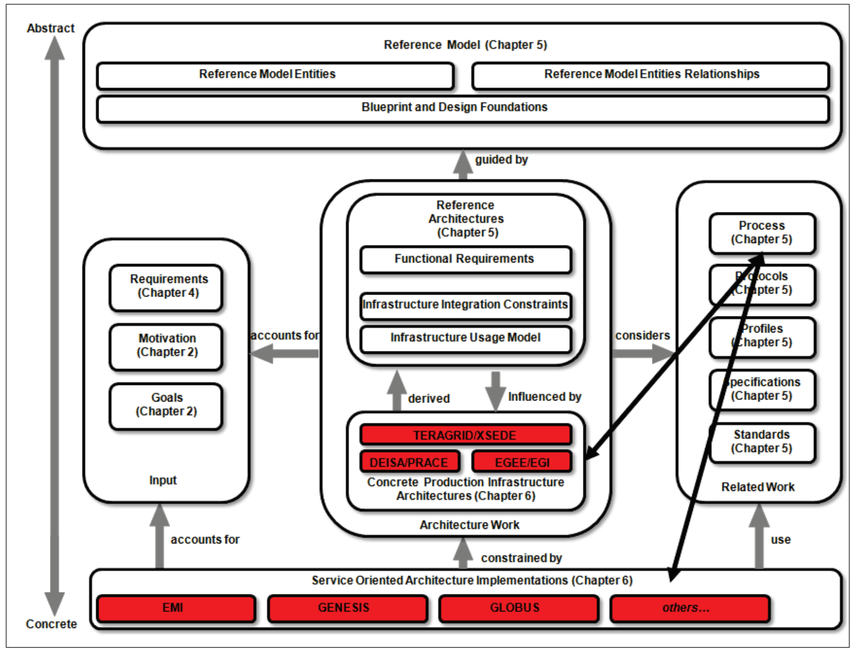


Figure 6.1: Reference model associated process implementation in context of concrete architectures and SOAs.

The initial chapters of this thesis indicated that the interoperability of production e-Science infrastructures (cf. Definition 5) is not existing today. The lack of current approaches to adapt to changes in infrastructure environments driven by various operational policies was also introduced. It is thus a challenge being able to cope with the dynamics of production infrastructures while satisfying unique end-users demands in a reliable fashion. Chapter 3 concluded that the adoption of open standards is not enough and a comprehensive reference model is required in order to provide guidance to infrastructures and technology providers how standards can be jointly used, developed, and maintained together.

In order to sustain interoperability, a complemtary process alongside the proposed reference model and its architecture is introduced in Section 5.3. It proposes the *'seven segment-based process for infrastructure interoperability'* while each of its segment bears the potential to increase the chance of interoperability between production e-Science infrastructures. The segments presented in Section 5.3. are technology- and infrastructure-agnostic and do not provide specific solutions and rather represent *'recommendations'* how interoperable solutions can be reached as a long term process. In contrast, this section describes how the different segments have been implemented in the context of concrete production e-Science infrastructures and their technology providers as illustrated in Figure 6.1 as part of the thesis studies.

Table 6.1 provides an overview of the seven segments, including the areas of work concering the implementation of the whole process that is collaboratively implemented with the

No	Segment	Areas of Work
1	Open Standards-based Reference Model	Reference Model design layout and its implementations in EMI and XSEDE
2	Collaboration with the Right Set of Vendors	EMI project with major EU middlewares and XSEDE in US
3	Reference Implementations	Open standards reference implementations including proposed additional concepts
4	Standardization Feedback Ecosystem	Ecosystem of OGF GIN and PGI to channel production feedback to standards
5	Aligned Future Strategies & Roadmaps	Aligning EMI and XSEDE roadmaps including ARC, gLite, UNICORE, and GENESIS
6	Harmonized Operation Policies & Roadmaps	Middleware security group activities supporting security harmonization activities
7	Funding Sources and Cross-Project Coordination	EU EMI and US XSEDE participation including activities in SIENA

Table 6.1: Seven segments with areas of work as part of the process implementation.

broader Grid community. The subsequent segments reveal how the thesis contributed to the interoperability of production e-Science infrastructures and its technologies through this process. The information about the implementation of the segments in the next sections thus also aim to verify and validate that the segments make sense in practice.

The design and implementation of the process is published in [254] as a guide for other e-Science infrastructures such as emerging ESFRI RIs. The process was presented and discussed at the EGI technical forum 2010 in Amsterdam [26] and at Cloudscape-III in Brussels [88] thus taken as input for the SIENA roadmap process [87].

6.1.1 Segment 1: IIRM and Standards-based Reference Architecture

The implementation of the first segment towards interoperability is one major part of this thesis by defining a reference model with an aligned standards-based reference architecture in Chapter 5. The architectural design is defined under the umbrella of the broader reference model term and named IIRM. The IIRM is analysed in this paragraph based on the factors (cf. Definition 26) and indicators (cf. Definition 25) defined in Chapter 3 and several requirements identified in Chapter 4.

The verification need to start with the check whether the IIRM is in accordance with the general design principles of reference models in software engineering as defined in Definition 28. The architectural design of the IIRM is abstract meaning that the entities and its reference architecture are only an abstract representation of potential deployments. The concept of Grid services is used in the core building blocks (i.e. OGSA-BES, SRM, etc.) making no arguments about real specific services in existing production infrastructures. Not any particular OGSA-BES deployment is referenced, for instance as part of DEISA/PRACE or a specific SRM deployment within EGEE/EGI. Being also in-line with Definition 28, a clear set of basic entities of the reference model is defined, including their relationships.

The principle of defining a clear focused problem space for the reference model and its reference architecture is followed. The problem space is set as production e-Science infrastructures (cf. Definition 5) with HPC, HTC and storage resources.

No assumptions about specific Grid middleware systems (cf. Definition 12) such as gLite, UNICORE, or ARC are incorporated in the model or its derived reference architecture. But many lessons learned from applying these systems to scientific use cases that take advantage of more than one Grid have been used to refine core building blocks with several concepts. These concepts have been specifically defined to support computational processing and related data-staging activities as well as better e-Science application support and thus does not leave the clearly defined problem space.

The IIRM is thus technology-agnostic and is described independently from real existing production e-Science infrastructures although IIRM concepts have been used in practice with these infrastructures to get feedback during the overall reference model design process and to underpin the theoretical academic studies. As the previous paragraphs reveal, the IIRM is in-line with the basic principles of reference models in software engineering and thus satisfying Definition 28.

Relevant Factors	**Indicators**	**OGSA**	**IIRM**
(a) Service based (Reference Model)	(1) Service Oriented Architecture (SOA)-based design for a distributed system	Yes	Yes
	(2) Entities offer service interfaces	Yes	Yes
	(3) Clear unique service semantics	Yes	Yes
(b) e-Science Context (Reference Model)	(4) Focussed on scientific use cases	No	Yes
	(5) Grid execution management elements	Yes	Yes
	(6) Grid data management elements	Yes	Yes
	(7) Grid security elements	Yes	Yes
	(8) Grid information elements	Yes	Yes
(c) Specified relationships between different relevant functional areas (Reference Model)	(9) Information and Compute	No	Yes
	(10) Information and Data	No	Yes
	(11) Security and Compute	No	Yes
	(12) Security and Data	No	Yes
	(13) Information and Security	No	Yes
	(14) Compute and Data	No	Yes
(d) Details for implementation (Reference Architecture)	(15) Based on concrete Web services (WS)-based Architecture as SOA implementation	Yes	Yes
	(16) Concrete specifications with referenced portTypes (e.g. Operations) or schemas	No	Yes
	(17) Invariants and constraints for the use of information and security data exists	No	Yes
(e) Production-oriented (Reference Architecture)	(18) Number of core service entities is lower than 5 (model not too broad)	No	Yes
	(19) Definition of core entities that must be in place to form an infrastructure	No	Yes
	(20) Used normative specifications are already defined and publicly available	No	Yes
	(21) Project that implements core entities and relationships (funding exists)	No	Yes
(f) Standards based (Reference Architecture)	(22) Based on normative standard specifications	No	Yes
	(23) Specifications from real SDOs	No	Yes
	(24) No break of existing established standard specifications	No	Yes
(g) Adoption in e-Science production technologies (Derived Concrete Architectures)	(25) EGEE / EGI technologies	No	Yes
	(26) DEISA / PRACE technologies	No	Yes
	(27) Production middleware systems (e.g. ARC, gLite, Globus, UNICORE)	No	Yes

Table 6.2: Factors and indicators of the IIRM in comparison with OGSA.

Complementary to this analysis about general reference model principles, another key contribution of this paragraph is the evaluation whether the IIRM overcomes the limitations of OGSA as published in [264]. The reference model factors and indicators from Chapter 3 are used that have been obtained by critically reviewing OGSA in the light of the given problem space. The summary of the analysis is shown in Table 6.2 thus claiming that the IIRM, in contrast to its major related model OGSA, fulfills all factors and indicators. It illustrates one key contribution of this thesis that is the definition of a reference model with associated architecture elements that overcomes limitations of OGSA. In more detail, Table 6.2 shows that for each indicator where the OGSA analysis revealed a 'no', the IIRM analysis reveals a 'yes'. This can be at least partly explained by the fact that the illustrated factors and their indicators include lessons learned from OGSA experience after one decade that also majorly influenced the requirements in Chapter 4.

Evaluations on the Reference Model Level

On the reference model level, the IIRM fulfils the requirement of being *(a) Service based* as Section 5.1.1 reveals by being in-line with SOAs as defined in Definition 29. This factor is fulfilled by having entities that are all service-based by offering abstract service interfaces and clear service semantics as required by the Definitions 31, 32, 33, and 34 as Section 5.1.1 describes (cf Table 5.2).

These entities are all derived from the IIRM focus on the *(b) e-Science Context* incorporated in the requirements of Chapter 4 that mentions no e-business use cases or commercial requirements. The IIRM is thus in-line with Definition 30. The IIRM has *(c) specified relationships between different relevant functional areas* as Section 5.1.1 reveals thus addressing all requirements defined in Definitions 35, 36, 37, 38, 39, and 40 (cf. Table 5.2).

Evaluations on the Reference Architecture Level

On the more concrete reference architecture level, the IIRM and its reference architecture presented in Section 5.1.2 provides all *(d) details for implementation* necessary to realize the more abstract IIRM design. The reference architecture thus satisfying the requirements raised in Definition 41, 42, and 43 (cf. Table 5.4).

The IIRM reference architecture is *(e) production-oriented*, because its design is addressing the requirements defined in Definition 44, 45, and 46 (cf. Table 5.4). In addition, as Section 6.2 will reveal, there are projects (i.e. EMI and XSEDE) that implement significant parts of the IIRM reference architecture thus indirectly leading to an impact in real production e-Science infrastructures. While more details are given later, evidence is provided as part of the EMI *'description of work'* [15], which content plans adoptions of many core building blocks and concepts of this thesis for the middleware systems ARC, gLite, UNICORE, and dCache. Furthermore, an XSEDE article [235] higlights that several XSEDE core building blocks are identical with the IIRM reference architecture core building blocks. The aforementioned indicators and evidence together sum up to a key impact of this thesis meaning that the IIRM is a compact solution that can be applied in e-Science today and is not just a theoretical model for *'testbeds'*.

One of the key design element of the IIRM reference architecture is being *(f) standards based* as required by Definition 47 and defined in Section 5.1.2. Although the reference architecture also propose several refinements as defined in Definition 48, they do not break the backwards compatibility of the standards-based core building blocks. These refinements are already fed back to the standardization process via the OGF PGI working group influencing already other relevant groups such as OGSA-BES, JSDL, or GLUE2.

Evaluations on the Concrete Architecture Level

Section 6.2 will reveal more evidence that the reference architecture is considered for *(g) adoption in e-Science production technologies* as part of more concrete infrastructure architectures. For the overall assessment of the IIRM in this section, again the EMI 'description of work' [15] provides evidence that ARC, gLite, and UNICORE are released together as EMI release that in turn is part of the Universal Middleware Distribution (UMD) [14] deployed on the EGEE/EGI infrastructure. Evidence that the reference architecture adoption in UNICORE is used in PRACE is provided in [78]: *'UNICORE allows seamless access to distributed resources via the Internet. It is deployed and proven in PRACE, the Partnership for Advanced Computing in Europe, to access systems in the European HPC ecosystem'*.

6.1.2 Segment 2: Collaboration of Infrastructures with Technology Providers

The second segment towards interoperability of e-Science infrastructures is described in Section 5.3.2 and recommends a collaboration with key technology providers that have a real impact on existing production e-Science infrastructures (cf. Definition 5). The mechanisms of e-Concertation meetings was introduced, but the segment recommends collaboration beyond such meetings that only happens once or twice a year.

The recommendation further indicates that the collaboration should start as early as possible, because later joins of new techology providers makes it unlikely that they accept certain technology directions, like for instance, following the proposed reference model in segment one. One example is the standardization effort of the initial OGSA-BES standard where many commercial and academic technology providers have been involved (e.g. Microsoft, UNICORE, etc.). Several academic-oriented providers such as gLite have been out of the process for quite a while although having a significant impact on EGEE/EGI deployments. That led to an initial rejection of the OGSA-BES concepts from the key technology provider gLite and therefore it is still not widely deployed in the EGEE/EGI infrastructure. A closer collaboration between the key technology providers UNICORE, gLite, and Globus changed this within OMII-Europe where gLite got funding to develop CREAM-BES [220].

But also OGF acts as a forum for collaboration as part of the GIN community group and its spin-of PGI group wherein the majority of technology providers from the production e-Science infrastructure community are involved as shown in Table 6.3. Evidence for the discussion and collaboration around use cases is provided as part of the PGI Use case document [270] that contains a table with contributors that is the source for Table 6.3. To provide one example for the importance of collaboration, as part of PGI, all the different technology providers came up with more than 200 requirements [77]. The reference architecture and its refinement concepts of this thesis have been also given as a major input to the process and are part of this table. As part of the ongoing collaboration in GIN/PGI, the technology providers listed in Table 6.3 agreed to implement such requirements over the course of the next years given funding is provided. And even if one provider might leave, often the pressure of community needs and the desire of end-users to choose the technology they want can finally make a difference leading to pressure to implement a standard for the *'leaving provider'* when all others implement it. Furthermore, if funding is provided through projects for technology providers, they are more likely to implement standards as the OMII-Europe case reveals implementing CREAM-BES for gLite [220].

The international production Grid coverage of partners in Table 6.3 and in [270] clearly reveals one key impact of the thesis as part of this segment: The collaboration in GIN and PGI activities leads to a broader international IIRM adoption since many PGI requirements are derived from this thesis [77].

Grid Technology Provider/Project	Production e-Science Infrastructure
ARC	NDGF, EGEE / EGI
gLite	EGEE / EGI, OSG (as part of VDT)
UNICORE	DEISA / PRACE, TeraGrid / XSEDE, EGEE / EGI
Globus (IGE project), GENESIS	TeraGrid / XSEDE
NAREGI / RENKEI	NAREGI / RENKEI Infrastructure
EDGES / EDGI	BOINC-based infrastructures (i.e. Desktop Grids)

Table 6.3: Key technology providers and projects with related infrastructures.

Having a collaboration with key technology providers such as in PGI is non-trival but the evidence of the implementation of segment 2 shows that this can be done using a neutral forum such as a SDO. Table 6.3 obtained from [270] clearly indicates that the implemented collaboration consists of quite a significant fraction of key technology providers that in turn contribute to the impact of this thesis. More evidence is given by the fact that the IIRM and its associated reference architecture has been given as a starting input to PGI reported in [79]. Already at the PGI kick-off session [67] the idea to follow the core building blocks of the thesis reference model and associated architecture was presented and agreed including the idea of refinements. Further evidence for impact is provided by the media [54, 79].

6.1.3 Segment 3: IIRM Reference Architecture Implementations

The third segment as described in Section 5.3.3 augments the theoretical work around reference models and associated architectures with practical expertise arising from proof-of-concept implementations. The key recommendation of this segment is to *'consider making a range of prototype implementations and interoperability testing part of the standard development'* as mentioned in the EICTA white paper 2006 [158].

One practical benefit is that the implementions usually cover more than one technical areas like information and job management or security and data management. This directly leads to dependencies between the reference architecture core building blocks meaning the relationships and potential missing links between the choosen standard specifications. A reference model should not only emphasize on its entities, but should also carefully define relationships between different entities. The refinement concepts in Section 5.2 provides examples of filling so called *'missing links'* between open standard specifications (e.g. JSDL and GLUE2) on the reference architecture level. Reference architecture implementations can thus help to understand various issues related to the use of multiple core building blocks of the reference architecture in practice. This also implies creating prototypes of whether the choosen core building blocks can work with existing infrastructure boundary conditions such as dedicated security setups (e.g. proxies [300]).

Many thesis studies and reference architecture concept implementations have been performed to complement the theoretical reference model with practical verification of their applicability in real production infrastructure setups. Many of these proof-of-concept implementations have been published with contributions listed in Section 1.4. Examples are the use of SAML and OGSA-BES in [302], the GLUE2 integration with OGSA-BES in [224], using UNICORE OGSA-BES with proxies and benchmarks [191], or OGSA-BES interoperability between glite and UNICORE [220], just to list a few out of Section 1.4. Such proof-of-concept field studies are also listed throughout the definition of the reference architecture and its refinements in Chapter 5. These publications of early developments points to one key contribution of this thesis that is implemented through this segment meaning the performed proof-of-concept implementations of IIRM reference architecture elements. Other reference architecture adoptions can be found as part of the EMI 'description of work' (e.g. GSI removal of EMI components or SAML adoptions) [15].

The implementation of this segment also contributes to the finding that the reference model approach is different from the existing profiling approach [122]. Profiling often takes a collection of available standards and defines it together as a profile while often the important relationships between them are neglected or only rarely described. By implementing this segment, *'missing links'* between standard specifications became often visible. Profiles are often focused on one dedicated technical concern (e.g. HPC-BP profile [154]) and realized as complex *'hierarchies of profiles'* meaning that they reference several specifications, often from the same technical area, that in turn are often based on profiles. The approch is thus not in-line with Definition 44. When the number of profiles rises, the probability of having different technology providers adopt the same collection of profiles significantly descreases when not being guided by a greater reference model.

For example, although first good experiences with implementations of the HPC-BP profile [154] exists, it needs refinements to match production use case requirements. Examples are the HPC extensions listed as part of Section 5.2 to efficiently run e-Science applications (cf. Definition 9) on production HPC machines.

In contrast, the thesis approach defines the relationships between entities thus filling *'missing links'* between specifications such as GLUE2 and JSDL. In this context, the proof-of-concept implementations during the IIRM definition have been a valuable step since they pointed in many cases to missing links particularly arising from combining different functionality areas such as data, compute, information, and security. The IIRM design and refinements in Section 5.2 provides many concepts that include linked specifications in order to offer solutions for these 'missing links' originally identified by reference implementations. The listed publications in Chapter 1 as well as the IIRM refinement concepts presented in Section 5.2 contribute to one of the major contributions of this thesis by identifying 'missing links' between specifications of different technical areas and providing solutions.

The core building blocks and their inter-links create a reference architecture that defines the basic functionality required in production e-Science infrastructures as Chapter 4 reveals. The set of well-specified inter-linked and refined standard interfaces are underpinned with experience from various IIRM concept prototypes and implementations that have been published and listed in Section 1.4.

Finally, this segment also contributes to the fact of being *'production-oriented'* by having performed several reference architecture prototypes to verify theoretical concepts (cf. Chapter 5) with production e-Science applications (cf. Chapter 6 case studies) before full-blown production begins. The overall design is based on the known idea of *'keep it simple and focus on essential services'* according to the TCP/IP approach reviewed in Chapter 3, since the more complex (e.g. OGSA), the more manifold the standard adoptions (e.g. profiles of profiles) and less likely is full interoperability.

6.1.4 Segment 4: The GIN and PGI Standardization Feedback Ecosystem

The standardization feedback ecosystem is another important segment towards interoperability that has been described in Section 5.3.4 in detail. The key recommendation of this segment is to *'provide feedback channels for standards maintenance'* as mentioned in the EICTA white paper 2006 [158]. This section focuses thus on the implementation of this particular segment in the given problem space and describes how lessons learned in practically using standards are channeled back to the OGF.

One particular cornerstone throughout Chapter 5 is the interoperability around the OGSA-BES standard [169] that several relevant technology providers have already deployed in production e-Science infrastructures like UNICORE in DEISA/PRACE [263]. This is only one example, where the use of this open standard under the umbrella of the OGF GIN group was used in production use cases by end-users (e.g. VPH e-Health community [263], WISDOM biomed community [259], and fusion community [225], etc.). These activities lead to numerous important lessons learned on how these standards can be improved.

The immediate question arising from such an aforementioned example is how the experiences are channeled back to the SDO while the OGSA-BES group is inactive for quite a while and several insights have been also touching JSDL that is yet another group. The OGF errata process [146] might be used or the creation of an *'experience document'* [146] but both are only about correcting a specification or documenting experiences with a specification rather than actively working on the next generation of specifications.

The JSDL experience document [222] has been already written back in 2008 and also GLUE2 is related when considering production experience in terms of resource descriptions for large-scale HPC resources (e.g. network topologies, etc.). The essense of this segment is therefore to transfer lessons learned from production that cross the area of different specifications and groups. This can not be done by a single group that is only focussed on one particular standard while the gained experiences often crossed the border of one particular other standard. The proposed solution to these aforementioned problems that arised in many similiar interoperability setups is the implementation of this segment being illustrated in Figure 6.2.

Figure 6.2 illustrates the implementation of this segment by establishing an standardization feedback ecosystem in the context of the SDO OGF providing an active channel for standards maintenance of OGF specifications. The author of this thesis has been actively involved via co-

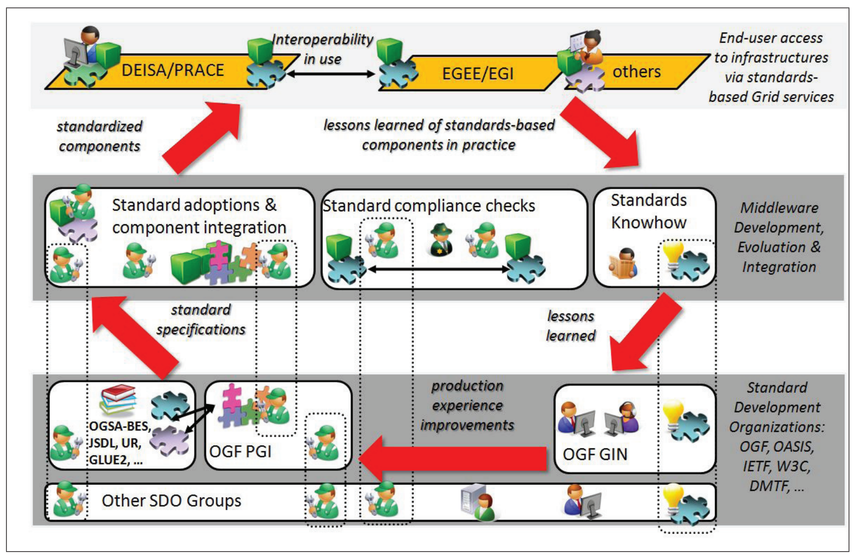


Figure 6.2: Implementation of segment 4: Standardization feedback system in OGF with GIN and PGI.

chairing of the GIN and PGI groups and works together with early adopters of open standards under the umbrella of the GIN and PGI groups in order to enable cross-Grid applications on real production e-Science infrastructures [270].

Performing realistic production e-Science infrastructure use cases and applications is thus a key element of the fourth segment in general and the overall ecosystem in particular. It ensures that the early open standard adoptions are already used with real e-Science applications from production Grids (cf. Definition 7) and thus lead to valuable lessons learned. Using open standards with real e-Science applications goes far beyond the effectiveness that standard compliance checks or simple interoperability demonstrations with *'bin-date-based applications'* can provide.

The aforementioned direction beyond demonstrations prooves real technical feasibility in practice but also guiding directions where specifications need to be tuned or refined (cf. Section 5.2). It must be ensured that lessons learned of standard-based components in practice are channeled back to OGF via valuable feedback for the next iteration of specifications (e.g. OGSA-BES 1.1 or 2.0). This is illustrated via the overall red cycle in Figure 6.2 where the knowhow of using standards in practice within GIN are taken by the PGI group for discussion. The aim of such an important element of the feedback cycle (i.e. PGI group) is to differentiate the concerns in terms of lessons learned from different technical areas and specifications. Then channel relevant feedback to the corresponding working groups of the different single technical areas (e.g. compute, security, data, information, etc.). In order to provide evidence, the PGI use case document [270], that among other contributions, contains major key contributions of proposed standard refinements from this thesis (i.e. Multi-Grid Drug Discovery Workflow), is taken as an input by other groups such as OGSA-BES, JSDL, GLUE2, and others. One example is the input from PGI to JSDL as presented in [76] that significantly overlap with this thesis contributions.

Such a process is by far underestimated in complexity requiring technical knowledge and knowhow of e-Science infrastructure setups. But the aforementioned facts and evidence highlight another impact of this thesis by establishing an active standardization feedback system with OGF GIN and PGI that is very active and continues to improve existing open standards with lessons learned of production e-Science applications (e.g. JSDL and PGI group [76]).

In both groups GIN and PGI, it is important that members of production Grids and relevant technology providers are part of such groups as the use case document reveals in the authors section [270]. This is required, because these experts act as a mediator between complex multi-technical-area production requirements and multiple corresponding one-technical-area-focused standardization groups. Regularly interactions between the channeling group (e.g. PGI) and the other standardization groups (e.g. JSDL, OGSA-BES, GLUE2, UR, etc.) are crucial.

In an ideal situation members of the channeling group should be actively participating in the other groups, too. In the described implementation, members of groups like OGSA-BES, JSDL, and GLUE2 contribute to PGI while PGI members essentially overlap with members of these groups.

The field studies WISDOM [259], VPH [263], and EUFORIA [225], contribute to GIN and PGI via lessons learned of how specific standards can be improved as documented in the PGI use cases document [270]. The PGI group is actively steered by the thesis author to ensure that the reference model proposed as part of this thesis also guides the work within the PGI working group from its start [67].

Standard refinements that are relevant to the IIRM core building blocks (e.g. OGSA-BES, JSDL, GLUE2, SRM, UR, etc.) are priorities in PGI while other standards (e.g. WS-Agreement, DRMAA) being not a high priority [77, 67]. With having production infrastructure and key technology provider members in the group [270] there is a high potential that the PGI requirements in general and the reference model in this thesis lead to new iterations of OGF standard specifications (e.g. JSDL 1.1 [76]). With the implementation of this segment, the suitability of open standards in real production environments is significantly increased through new emerging versions of the specifications. One of the key contributions of this thesis is thus with its case studies in Chapter 6 that revealed numerous lessons learned of using OGF standards in practice and by implementing this segment over years (e.g. OGF25 PGI session dates back to March 2009 [67]).

6.1.5 Segment 5: Aligning Middleware Roadmaps with EMI and XSEDE

As described in Section 5.3.5, the fifth segment towards interoperability is complementing the rather short-term and medium term technical activities from the previous segments with a long-term strategic endeavour. This paragraph provides insights into the contributions to the alignments of future strategies and roadmaps relevant for the scope of this thesis.

To recall from Section 5.3.5, the basic recommendation of this segment is to align future strategies and roadmaps that go beyond meetings that take place at EU concertation meetings [156] by affecting the day to day business of projects. In the particular context of this thesis, benefits include the use of products from one provider by different products from another or just providing complementary components across the technology providers. This avoids services duplication and leads to a complementary services landscape on e-Science infrastructures as illustrated in Figure 6.3. It also shows that not implementing this segment might lead to a fragmented service landscape resulting in non-interoperable infrastructures (cf. Definition 19).

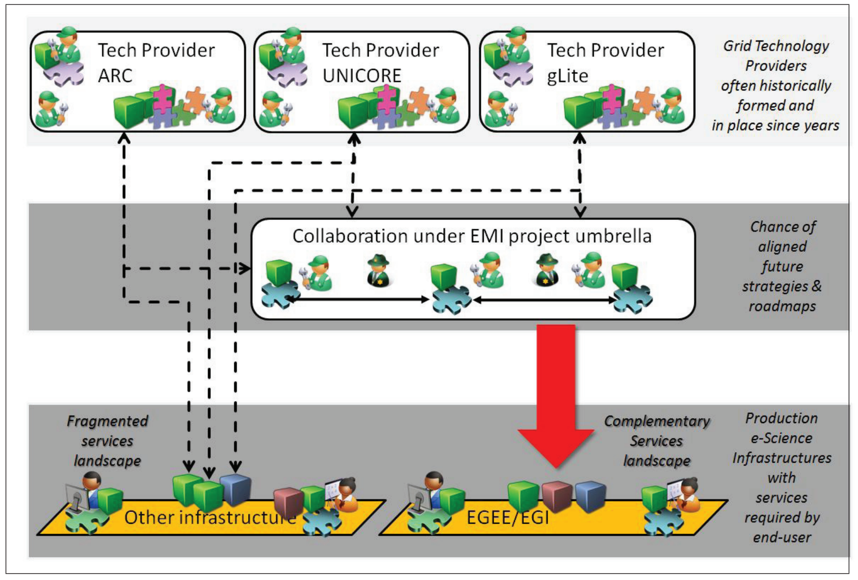


Figure 6.3: The Implementation of segment 5: Common EMI project harmonizing ARC, UNICORE, and gLite.

Many aspects of the IIRM have significantly contributed to the technical work plan of one of the EGEE/EGI major technology providers named as EMI as its 'description of work' reveals [15]. Figure 6.3 illustrates the EMI project as an umbrella of three distinct Grid middleware technologies (cf. Definition 12). These Grid middleware systems are used on various production e-Science infrastructures [14, 78] such as gLite traditionally used in EGEE/EGI, ARC traditionally used in NDGF, and UNICORE traditionally used in DEISA/PRACE.

By implementing this segment, within EMI these middleware providers collaborate in terms of aligning future strategies and long-term technology roadmaps as the *'description of work'* describes often mentioning also *'harmonization'* activities [15]. The EMI project thus jointly defines the design of the next generation of Grid technologies used in e-Science for the next couple of years in Europe. The work on the reference architecture and its core building block have been essential in the creation process of the EMI project and are incorporated in many places within the EMI *'description of work'* [15]. The contributions of this thesis to the EMI *'description of work'* (e.g. harmonization of security, standardization, etc.) [15] significanly contributed to the alignment of roadmaps between ARC, gLite and UNICORE being thus another key impact of this thesis.

One concrete example are the EMI execution service (EMI-ES) [283] prototypes that use many of the proposed concepts listed in Section 5.2 while being currently standardized in PGI. But the EMI-ES specification [283] also reveals that good concepts from OGSA-BES are taken as a basis such as the data-staging concept.

By focussing first on short-term harmonization between the EMI middlewares with the EMI-ES, the first steps are taken towards more long-term standardization activities and proof-of-concept implementations in the spirit of segment three. In parallel, PGI enables the long-term standardization of the implemented EMI-ES prototype concepts including other middlewares (e.g. GENESIS) with the aim to contribute to improved OGSA-BES specifications (e.g. OGSA-BES 1.1 or even 2.0) that will most likely be finished after the EMI project. Nevertheless, this can be still considered as a very important contribution to standardization as described in the 6th e-Infrastructure concertation meeting report: *'The participants agreed that although the full cycle of development of standards may sometimes exceed the lifetime of a typical EU project, but the projects can still make substantial contributions to at least part of the standards' development cycle (if not all of it)* [133].

In thise sense, the key concepts of the EMI-ES specification then need to be refactored towards the final standard specifications such as OGSA-BES 1.1 or JSDL 1.1 while standardization activities of those specifications have been already started in OGF. Due to the long standardization process, standard adoptions might even occur after EMI is finished (e.g. from GENESIS, UNICORE, etc.). Another EMI example is the use of SRM with non-GSI methods and the ideas of delegation avoiding the proprietary use of HTTPG (cf. Section 5.1.6) that is part of the EMI work plans [15]. Studies of missing links have been taken up like, for instance, that GLUE2 is broadly considered in the JSDL evolution parts of the EMI-ES job description [283] concepts that is also in standardization within the JSDL group [76].

The roadmaps of the three EMI technology providers was different in the past where UNICORE was focused on a design that satisfies HPC end-users where gLite was focused to satisfy the needs of HTC-driven and data-oriented end-users. ARC was primarily used and developed in the nordic countries serving the needs of WLCG customers through the NDGF Grid. Although the roadmaps of the three technologies are still having their own identity (e.g. security setups), a significant fraction of each of the roadmaps is influenced by key themes of this thesis since the EMI description of work [15] follows key concepts of this thesis.

To provide an example, the roadmap of UNICORE and its components is defined by the UNICORE collaboration, but several components are influenced via the EMI roadmap to achieve a better harmonization with gLite and ARC [15]. The difference in focus, usage, and uniqueness in supporting different technology aspects or computational paradigms is one of the most EMI benefits of the alignment and harmonization process given by this segment. For instance, UNICORE is more than 10 years used in HPC while gLite requires HPC support and thus it is wise to cooperate when roughly 10 years of experience can be transferred within a collaborating project from one technology provider to another.

While one can consider to even agree on one computational Grid middleware and merge all three ARC, gLite, and UNICORE, all of them have still unique functionalities that can not be easily merged without giving up their key unique architectural designs (e.g. brokering vs. explicit choose of sites). One example might be the *'component per Grid service'* approach in gLite versus the *'multiple Grid services all-in-one container'* approach as followed by UNICORE. The support for large-scale distributed data-management in UNICORE can be improved (e.g. SRM interface for UNICORE storage services [15]) while the gLite collaboration has experience with this topic over 10 years now.

But focussing on Europe alone is not enough as the e-IRG white paper 2009 indicates with *'It has been long recognised that e-Infrastructures are not restricted to country borders; on the contrary they must be planned and set up as global infrastructures in order to create and effective and competitive scientific ecosystem'* [206]. In addition to European activities, this thesis in general and this segment in particular also influenced the XSEDE architecture [235] through collaboration in the project XSEDE and by having the thesis author a part of the architecture team. The XSEDE architecture [235] is influenced by the IIRM reference architecture with OGSA-BES, JSDL, and other standards, by having members from XSEDE participating in GIN and PGI (e.g. GENESIS) [270] over several years as a result of fruitful collaboration. The Juelich Supercomputing Centre is an official project partner of the XSEDE project [78] bringing Europe and US closer together through the interoperability activities between UNICORE and GENESIS.

Also Globus is still a technology provider of XSEDE and members of Globus are also part of the architecture team to ensure a harmonized architectural design. The implemented segments have been essential for the establishment of this unique collaboration within one common project. Impact and evidence for the aforementioned segment implementations around standardization is best reflected by recent XSEDE media articles [80, 81, 78]. Hence, another key contribution of this thesis is the alignment with XSEDE roadmaps between GENESIS, UNICORE and Globus under the XSEDE project umbrella.

6.1.6 Segment 6: Harmonized Security Setups and Operation Policies

This segment recommends complementary work to technical activities related to policy harmonization via dedciated groups that all share the goal to ease the usage of (multiple) e-Science infrastructures for end-users. Also, many activities undertaken as part of the e-Infrastructure concertation meetings aim to lower the barrier for end-users to use e-Science infrastructures (e.g. [133, 134, 156]). Section 5.3.6 highlights that this particular segment is important and represents a complementary field to technical interoperability towards a full operational model where infrastructure boundaries are lowered.

This thesis segment was only implemented to a limited degree majorly because of its technical focus. This paragraph will thus give only some examples that have either directly or indirectly influenced the findings of this thesis via several related minor contributions.

Section 5.3.6 explains that dedicated groups with specific key topics are a very important *'tool'* towards operational interoperability. The field of Grid security groups that harmonize security policies (e.g. authentication and authorization) is one example to work on interoperability between production e-Science infrastructures. While many challenges in security are tackled with technology approaches, important work needs to be done for harmonizing security policies as well. One first example of such a policy oriented group is the Joint Security Policy Group (JSPG) [50] that was regrouped to the EGI Security Policy Group (SPG) [13]. In this policy group members of EGI, OSG, and other infrastructures discuss security policies that lead to a more streamlined security setup over the long term and should be extended to other e-Science infrastructures.

The aforementioned group complements results of the former Middleware Security Group (MWSG) [56] where the thesis author participated. This group is focused on technology interoperability in the field of security but having also members of production e-Science infrastructures. The thesis author as MWSG member [57] influenced roadmaps towards SAML and XACML as a solution for infrastructure security setups having shown presentations and demonstrators via UNICORE being interoperable with gLite [302] security setups at MWSG meetings (e.g. [57]). This includes initial work in OMII-Europe [69] that afterwards influenced EMI work plans [15] more recently. This evidence highlights another key impact of this thesis by enabling the proxies, SAML, and XACML security harmonization between gLite and UNICORE as part of the *'description of work'* [15] of the EMI project as a result of being involved in MWSG [57].

It often remains a policy decision which technology end-users can use in production infrastructures. But also technical progress has the potential to influence those policies and realizing a harmonization in infrastructure interoperability setups. Many findings of this thesis have been given as an input from a UNICORE perspective to the international MWSG (e.g. [57]). The aforementioned aspects point to another key contribution of this thesis by promoting within the MWSG the IIRM reference architecture security pattern based on open standards (e.g. SAML and XACML [57]). As a consequence of this work within the MWSG group, SAML and XACML play a crucial role for all three EMI middleware systems as well also influenced by the contribution of the reference architecture security pattern (cf. Section 5.1.6) to the EMI work plans [15] as security harmonization.

One of the case studies where the aforementioned policy aspects matter is EUFORIA [225] that is presented in detail in Chapter 6. In short, the EUFORIA framework [225] access to different e-Science infrastructures is largely based on proxies that have been in the past neither supported by UNICORE nor accepted by the DEISA/PRACE infrastructure. But as part of the thesis work, EUFORIA uses EGEE/EGI and DEISA/PRACE resources jointly together with the EUFORIA framework using parts of the security plumbings (cf. Section 5.1.6). The agreement in terms of DEISA/PRACE security policies was supported by technology improvements (i.e. X.509 proxies in UNICORE) that in turn led to harmonized operational policies. UNICORE was enhanced with optional proxy support and the functionality of UNICORE was increased to express more fine-granular policies using XACML and providing support for VOs using attribute-based authorization methods [302]. This technological advancement enables fine-grained policies where only end-users from EUFORIA are allowed to use proxies while all others have to use their full certificates. Another example of this segment is work undertaken in the EEF [23] where members also discuss operational issues between production e-Science infrastructures and their emerging requirements.

6.1.7 Segment 7: Funding and Cross-Project Collaborations

The final segment towards interoperability is massively driven by sustainability and the desire to have synergetic solutions across the projects of the whole e-Science community. Also the implementation of this segment is undertaken partly with e-Infrastructure concertation meetings as sources reveal [133].

Section 5.3.7 introduces this segment by acknowledging that different funding sources and *'scientific project-like'* funding is enabling the mainstream of thesis activities. The thesis contributions have only implemented this segment to a limited degree, because of its rather technical nature.

But a first concrete example of minor contributions is part of the thesis work and about the different funding sources of UNICORE and Globus. Both could have benefit from a very close collaboration, but they have been developed over a decade by different funding sources meaning US fundings for Globus and EC funding for UNICORE. This led to non-interoperable solutions in the past that can be at least partly explained by these different funding sources.

The collaboration is increased in terms of the EU project collaboration of EMI [27] and the IGE project [45], which represents Globus in Europe. Hence, this collaboration is supported by one dedicated European project that bridges to the US-funded Globus project in USA while OMII-Europe [69] was a joint project with Globus and other European middleware provider (i.e. UNICORE and gLite) in the past. Relevant for this thesis is the fact that OGSA-BES was implemented in Globus during OMII-Europe as a prototype. A OGSA-BES frontend for Globus is provided in the 2.0 release of the IGE project that reveals the importance of such projects [44]. Furthermore, the IGE project contributes to PGI [270] activities paving the way to better interoperability across US and EU.

Apart from the previously mentioned XSEDE collaboration [235], another example for cross-project collaboration activities is the FutureGrid project [34] that is open for European and US technologies. Funded by the US government, this training infrastructure is used by EMI for demonstrations and is performing end user trainings on it [16]. The basic idea of FutureGrid is to have a permanent testbed for interoperability, evaluations, and trainings constantly up and running. This directly address one challenge often observed in performing interoperability across infrastructures within GIN where endpoints seem to be very unreliable. FutureGrid provided evidence that interoperability matters for the US funding sources as within Europe through OMII-Europe, EMI and IGE respectively.

All the aforementioned evidence sum up in another minor contribution of this thesis work that includes cross-project collaboration among international partners mainly through OMII-Europe, EMI, as well as XSEDE. This also includes work with FutureGrid being the testbed for technologies in XSEDE with UNICORE installations and now used in EMI as well [16].

Finally, Section 5.3.7 mentions that one central project should be in place to promote the collaboration between other projects in terms of standardization roadmap activities. This segment is implemented by the thesis contributions to the SIENA project [87]. SIENA coordinates a joint roadmap across six so-called DCI projects. These are EGI-InSpire [25], EMI [27], Initiative for Globus in Europe (IGE) [45], European Desktop Grid Initiative (EDGI) [24], VENUS-C [99], and StratusLab [93] while the SIENA project represents a technology neutral project. SIENA discusses long-term standardization across these projects, including future possibilities using cloud computing techniques. Several results of this thesis have been given as an input to the SIENA roadmap process including the overall seven segment-based process that was presented at Cloud-Scape III [88]. The contributions given to the SIENA roadmap process provides another evidence of the impact of the proposed seven segment-based process. The result of SIENA is a common roadmap across the six projects providing also pieces of information of how cloud technologies influence the e-Science infrastructure landscape. The fruitful collaboration within SIENA is another evidence that this segment and such projects are crucial to enable long-term interoperable production e-Science infrastructures.

6.2 Concrete Architectures of Production e-Science Infrastructures

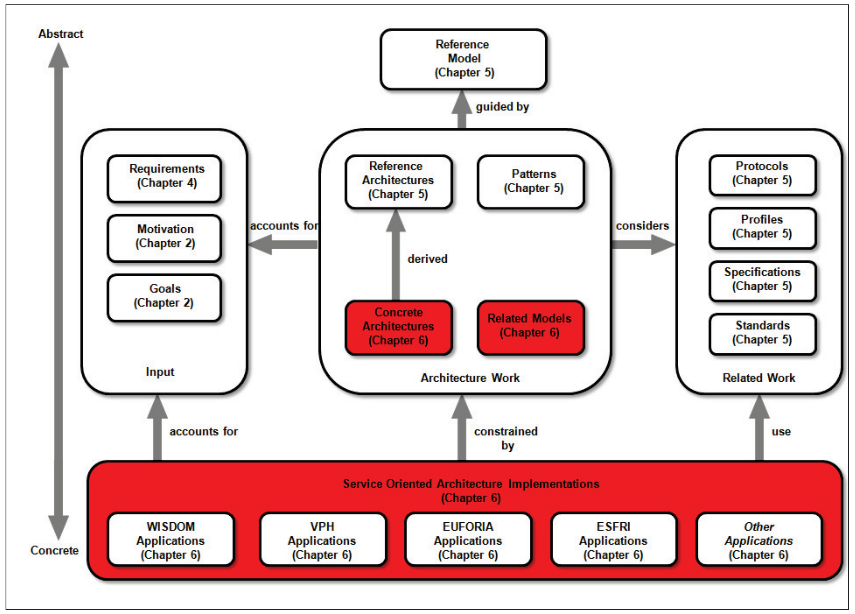


Figure 6.4: Concrete architectures of the reference model are derived from the abstract reference architecture.

Chapter 2 provided the necessary background about the state-of-the-art e-Science infrastructures and several of their interoperability challenges that exist today. After identifying the need for a reference model, its architecture, and an associated process in Chapter 3, the IIRM design and the seven-segment process are presented in Chapter 5. Both are based on a precise set of requirements raised in Chapter 4. Starting in Chapter 6 the seven-segment process is described and how it is specifically implemented in the given problem space where the IIRM is one important segment of it.

The necessary next step as part of this section is to provide evidence of how all the previous abstract and theoretical considerations lead to practical impact on production e-Science infrastructures (cf. Definition 5). This impact is shown in terms of technology improvements as well as in terms of scientific innovation through real e-Science use cases that take advantage of more than one infrastructure. This moves the focus one step further from the abstract to the more concrete as illustrated in Figure 6.4. It illustrates that the concrete architectures are derived from the reference architecture. SOA-based implementations are constrained by the concrete architectures, but enable applications well-embedded in an *'interoperability by design'* environment. Several related models can be used together with these concrete architectures but not need to be guided necessarily by the broader reference model design.

The next section thus aims to provide concrete architectures that are influenced with the reference model elements, while subsequent sections reveal insights on the undertaken application case studies in context. The concrete architectures as part of this section provide the overviews of the basic infrastructure setups in Europe, US, and other regions in the world, including the basic SOA implementations in Grid technologies that then provide an environment for well-embedded scientific applications. Infrastructure architecture deployments are outlined and the key contributions in this chapter are clearly the scientific case studies, since they use concrete e-Science infrastructure setups, which are also introduced in this section. In subsequent sections of this chapter, details of how the conducted scientific case studies and concrete applications benefit from the presented concrete architectures and the IIRM are presented.

6.2.1 Reference Model and Architecture Adoptions

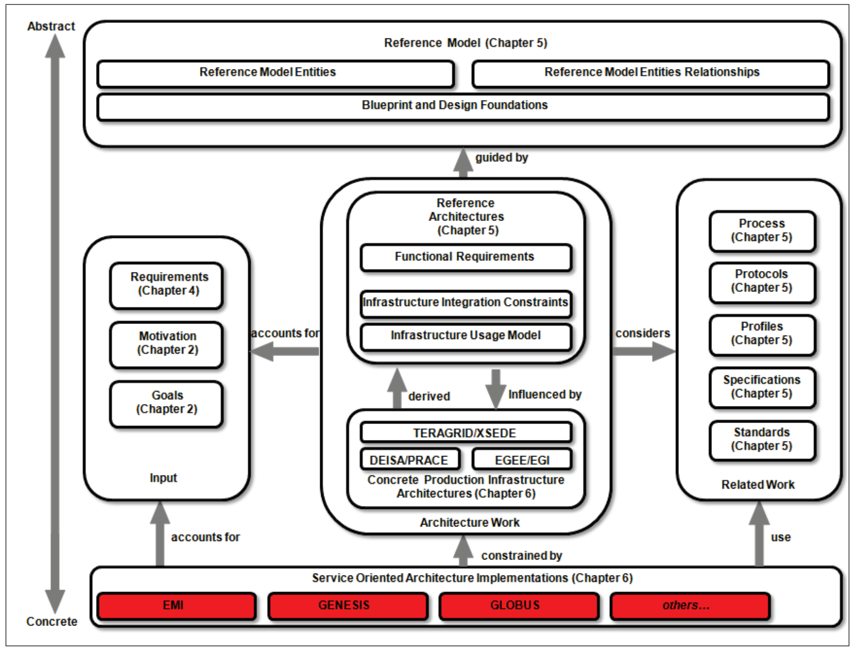


Figure 6.5: Reference Model and Architecture Adoptions Overview.

One key principle in software engineering is to differentiate between *'architecture and its implementation'* that in turn can be best supported by the adoption of open standards. As Figure 6.5 revealed, the thesis contributions are in-line with this principle thus also addressing the IIRM requirement defined in Definition 28. Chapter 5 focussed on defining rather abstract technology-agnostic concepts and core building blocks of a reference architecture that are implemented with concrete middleware adoptions in order to be deployed in production e-Science infrastructures. Emerging implementations of the IIRM concepts and core building blocks in Grid middleware (cf. Definition 12) are explained in this section. Although these are all in-line with the non-functional requirement raised in Definition 68, all presented Grid middleware implementations are independent from the concrete architecture implementations of e-Science infrastructures.

In this section, the focus is set on Grid middleware and their IIRM adoptions using SOAs, while concrete production e-Science infrastructure architectures will be described in subsequent sections. This is a key aspect since firstly the Grid middleware can be deployed on more than one production e-Science infrastructure using different configuration setups of its technical capabilities. Secondly, the concrete production e-Science infrastructure architecture itself should be kept technology-neutral from its architectural design while its implementations use Grid middleware that, in principle, can be exchanged. This is required in order to satisfy the requirement defined in Definition 73. The EICTA white paper 2006 refers to this important requirement as *'enhancing users choice'* [158].

The European middleware systems gLite, ARC, and UNICORE, adopt the core building blocks and the IIRM concepts through the EMI project [27] and evidence supporting this claim is already given in numerous segments of the implemented process at the beginning of this Chapter, but also given by the description of work of EMI [15]. Under the umbrella of the EMI project, gLite, ARC, and UNICORE work together on work plans that are massively influenced with the IIRM work over years [15]. The project therefore focusses on *harmonization* instead of pure developments and as such the adoption of open standards (cf. Definition 14) plays a major role in all of its technical areas [15].

The improvement concepts derived from the case studies in Chapter 6 and related GIN activities have influenced the IIRM design and therefore its refinement concepts have been given as an input to the PGI working group [270] of OGF and some parts are already in standardization within the JSDL group [76]. PGI works with representatives of all EMI middleware systems and beyond (i.e. listed in table within [270]) on the next versions of OGSA-BES, JSDL, GLUE2, SRM, UR, to ensure that important production e-Science infrastructure requirements are addressed. The standardization itself is a long-term activity that goes beyond the scope of this thesis, especially when considering that standardization takes a long time and even might be not finished during the run-time of the EMI project. These reworked lessons learned of production applications within PGI [76] are considered for concrete adoptions through EMI (e.g. EMI-ES [283, 15]) that in turn are one key contribution of this thesis.

In more detail, the EMI project has developed many IIRM concepts as part of the EMI-ES specification [283] as a proof of concept implementation and aligning UNICORE, gLite and ARC models. Many of these implementations bear the potential that middleware adoptions are proposed in Section 5.2 are able to use resources more efficiently that is another key impact of the thesis contributions. As open standard specifications of the emerging standards such as OGSA-BES 1.1, JSDL 1.1, and GLUE 2.1 are not likely to appear very soon, it might be not possible to develop them as part of the EMI project. It is further expected that either the collaboration of EMI is continued via other activities (e.g. joint open source software foundation like ScienceSoft [85]) or that the middleware provider will adopt the standards after EMI if funding is available. As traditionally standards have been important for UNICORE and GENESIS, it is very likely that these standards will appear in these systems.

Apart from the computational side, refinement concepts of the data and resource tracking domain are implemented as well [15]. For instance, the storage accounting record with the goal to extend the UR with storage information as proposed as part of Section 5.2. The storage accounting record [201] is another prototype that is currently developed and recently standardization activities have been started in OGF leading to an UR 1.1 specification in future.

Furthermore, there is also harmonization in terms of security meaning that the project develops aspects of the security pattern described in Chapter 5, including the work around SAML and XACML most notably as the 'description of work' reveals [15]. It is influenced with the thesis findings including the plan to adopt more open standards in the EU middleware gLite, ARC, and UNICORE being another impact of this thesis.

This thesis can only provide an architectural design in terms of the IIRM and its associated reference architecture including refinements for core building blocks and recommendations as well as prototypes. All these are valuable inputs to the standardization process that have been published over years in various occassions (cf. Section 1.4). But the detailed implementations of these concepts (i.e. source-code level) and their long-term standardization activities (i.e. new normative specifications) are beyond the scope (and timeline) of this thesis. This is not only because also other technologies may have other improvements in addition to the mentioned standards, but also because the *'standardization process'* itself is a community process that can take several years.

Many implementations of the IIRM concepts are considered as part of the EMI distribution implemented across the middleware systems gLite, ARC, and UNICORE [15]. The EMI distribution delivers middleware to existing production e-Science infrastructures (i.e. EGEE/EGI and DEISA/PRACE) with unique software that not many others in the Grid community are able to deliver. The emerging developments of EMI and its releases in EGI UMD [14] provide further evidence that the thesis influence existing production e-Science infrastructures by implementing major concepts and ideas of this thesis [15].

The EMI *'description of work'* [15] includes the IIRM core building blocks and its refinement concepts (i.e. PGI activities in [15]) that in turn illustrates the impact of this thesis in production technologies. Based on this evidence in particular and the EMI middleware in general, the requirement about *'supportability'* as defined in Definition 76 is satisfied.

But apart from the major involvement in Europe and through the process implementation described in the beginning of this Chapter, other reference architecture adoptions have been influenced with this thesis [235]. Most notably, US middleware providers are part of the activities within GIN and PGI as well [270]. Firstly, the US Grid middleware GENESIS [231] that is specifically optimized for campus Grids already adopts core building blocks of the reference architecture (e.g. OGSA-BES, JSDL, SAML, etc.) and is likely to adopt the refinement concepts as part of the PGI process as well [270]. This process is even supported outside of the standardization activities by deep collaboration in the XSEDE project where GENESIS alongside U NICORE are the middleware systems considered for the TeraGrid/XSEDE infrastructure that is revealed as part of other contributions in this chapter and published in [235].

Secondly, also the Globus middleware is constantly participating in PGI [270] via the IGE project [45] that already released OGSA-BES for Globus through integration with GridSAM [213] as part of the IGE 2.0 release [44]. The IIRM core building blocks and its refinements thus influence the European middleware adoptions, but also those in the US.

NAREGI/RENKEI and GridSAM are further middleware systems that are crucial to other international production infrastructures such as NAREGI in Japan and NGS in the UK. Evidence is provided through their constant participation in PGI [270, 84] and their involvement in the OGSA-BES related GIN demonstrations since many years [84]. Also these middleware systems plan to adopt the refinements of the core building blocks as part of the PGI standardization process [84], which outlines further evidence of international impact of the thesis results. The aforementioned facts contribute to another thesis contribution by performing numerous OGF activities to broaden the IIRM adoption beyond Europe (e.g. in GENESIS [235] and RENKEI [84]).

6.2.2 European e-Science Infrastructures Setup

The principle of differentiating between *'architecture and its implementation'* is relevant for middleware architectures, but also for e-Science infrastructures as Figure 6.6 illustrates. It enables the creation of concrete SOA-based production infrastructures (cf. Definition 5) that can be considered as a *'Network of interoperable Grid services'* according to Definition 15. Hence, e-Scientists are able to switch technologies that implement the same architecture core building blocks (e.g. standards) without being forced to use one particular implementation of the architecture.

'As yet there is no single unified computing system in the sense that there is a single World Wide Web; rather there is a (relatively small) number of grid, HPC and cloud services, each with different interfaces' [210]. Therefore, the major research question of this thesis is, *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined'*. This vision is llustrated in Chapter 2 in Figure 2.8 in the given problem domain. This section gives more answers to this question based on the IIRM defined in Chapter 5. Complementary, the previously mentioned emerging middleware adoptions of the IIRM and its assocatiated architecture work are put in context.

The first concrete architecture of European production e-Science infrastructures presented in this section is the EGEE/EGI infrastructure as shown in Figure 6.6. The transition process from a project based EGEE infrastructure towards an NGI-based sustainable infrastructure with a governing EGI.eu organization is finished. But the majority of NGI sites still run the gLite middleware with only a limited increase in UNICORE or ARC deployments. The flexibility of NGIs to choose the specific middleware they want motivates the thesis since the situation in EGEE/EGI gets even more fragmented when NGIs partly decide which software to be deployed. EGEE/EGI deploys middleware systems with the Unified Middleware Distribution (UMD) and thus the infrastructure will face a wide variety of middleware technologies that need to be interoperable when using resources of more than one NGI. The underlying compu-

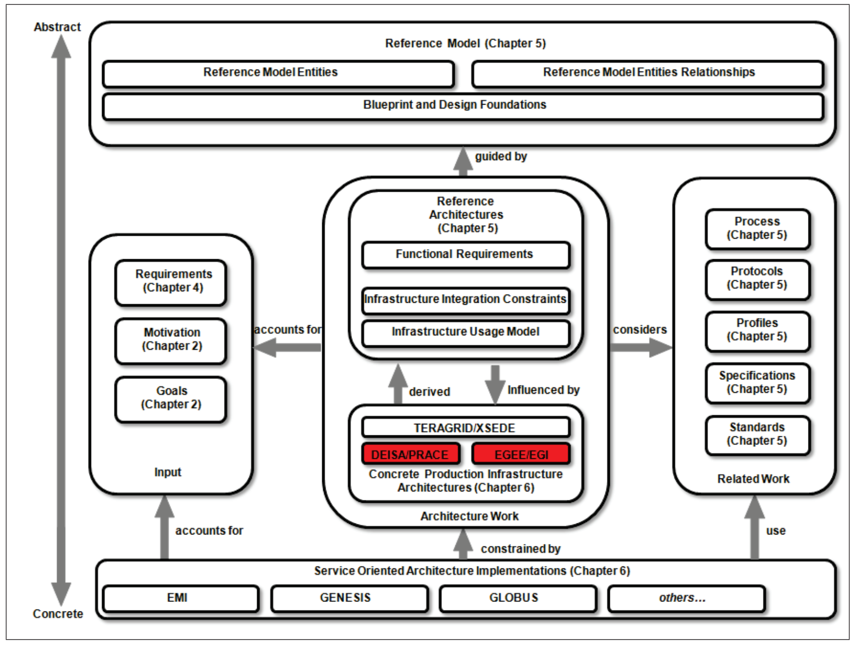


Figure 6.6: European e-Science Infrastructures Setup Overview.

tational paradigm in this infrastructure has not changed and thus EGEE/EGI is still considered as a HTC-driven infrastructure (cf. Definition 17). The requirements for interoperability within EGEE/EGI and with other infrastructures is even higher than before the transition since there is a demand for interoperability across the variety of middlewares deployed by many different NGIs. Some NGIs (e.g. the UK NGS) provide other technologies (e.g. GridSAM) in addition to those provided by UMD (e.g. UNICORE, ARC, gLite via EMI) and over time aim to get into the UMD among others (e.g. Globus).

The concrete architecture of the EGEE/EGI production e-Science infrastructure is influenced by the software within UMD. UMD in turn consists of middleware systems delivered by many technology providers while the EMI project is one of its major ones. The UMD 1.0 release consisted only of middleware components of the EMI 1 distribution getting more and more other software components. The influenced middleware adoptions with IIRM concepts and refinements thus also influence the shape of the EGEE/EGI production e-Science infrastructure via EMI and UMD. Using EMI middleware UNICORE, ARC, and gLite, EGEE/EGI is interoperable in the sense of Definition 21 among its NGIs as well as with other European infrastructures that are compliant with IIRM and its concepts such as DEISA/PRACE as the next paragraph will reveal. The aforementioned facts contribute to one key impact of this thesis with having IIRM core building blocks included in the EGI UMD mainly through EMI and its different middleware systems. As EMI and its middleware adopts a wide variety of open standards (e.g. OGSA-BES in UNICORE), the thesis further contributed to an increased interoperability via standards-based software in EGI UMD.

The second concrete architecture of an European production e-Science infrastructure is the DEISA/PRACE infrastructure that is still in the process of its transition. DEISA sites are considered to become part of the PRACE infrastructure and as such UNICORE is already installed on the supercomputer sites that offer access to resources in the resulting DEISA/PRACE infrastructure. The underlying computational paradigm HPC will not change and thus it remains a HPC-driven infrastructure (cf. Definition 16). The IIRM core building blocks and its refinement concepts are adopted by UNICORE through EMI. As EMI in general and UNICORE in particular is considered to remain the middleware provider for DEISA/PRACE, further evidence is thus provided that the IIRM influence the concrete architecture of DEISA/PRACE. Over the years, the thesis findings contributed via standards-based software UNICORE elements to the DEISA/PRACE infrastructure thus enhancing its interoperability capabilities.

Based on the aforementioned concrete infrastructure architecture setups of the EGEE/EGI and DEISA/PRACE, more insights into how core building blocks work with these setups are presented in the following paragraphs. With this, the IIRM satisfies the required production infrastructure integration constraints raised in Chapter 4. Figure 6.7 illustrates how the concepts published in [260] and described in Chapter 5 are used in context of existing constraints in the context of case studies like EUFORIA and WISDOM.

Figure 6.7 illustrates the concrete SOA-based production infrastructures EGEE/EGI and DEISA/PRACE with an emphasis on those services that are relevant for the invariants. The significance of the invariants in IIRM adoptions is visible in the computational parts when a detailed step-wise job submission process in context of concrete infrastructure architectures is described. The invariant implementations and applied methods are all also relevant for other reference architecture core building blocks (e.g. SRM), but the focus here is on computation. As shown in Figure 6.7, several core building blocks and their invariant implementations form a relatively complex architecture with essentially four layers.

The *'resource layer'* stands for the e-Science infrastructure resources that are available in EGEE/EGI and DEISA/PRACE. Examples are HPC- and HTC-computing resources while on this layer also data storage resources can be found in concrete production setups, but that are neglected to focus on the invariant description. On top of this layer, several IIRM reference architecture services are shown that offer access to the underlying resource layer and its functionalities realizing middleware functions. These services together form a *'network of interoperable services'* as defined in Definition 15 and are as such independent from the *'real physical e-Science infrastructures'* underneath (e.g. EGEE/EGI or DEISA/PRACE). The reason for this is because of the interoperability among the servies based on the IIRM and conformance to the invariants described in the next paragraphs.

One dedicated layer for the production infrastructures is added on top of these interoperable services to model the *'infrastructure boundaries'* as best as possible in Figure 6.7. This layer would include policies that govern access methods to the infrastructure (e.g. computational time grants, VO membership, etc.). Another layer of the architecture stands for *'scientific clients'* that can be any form of a *'scientific gateway'* providing easy access to e-Science infrastructure services used by scientists on a daily basis.

In this complex architecture, the IIRM invariants and its impact on the different levels are described in a step-wise fashion during cross-infrastructure application runs based on the *'design pattern'* introduced in Chapter 5. Each step marked as (n) corresponds to one step (n) within the provided production architecture setups illustrated in Figure 6.7.

An e-Scientist that works with a particular aforementioned scientific gateway (0) of his field (e.g. bio-informatics portal) integrates clients that access numerous IIRM services (i.e. OGSA-BES, SRM, etc.). Complementary to the functional services, there must be an integrated client

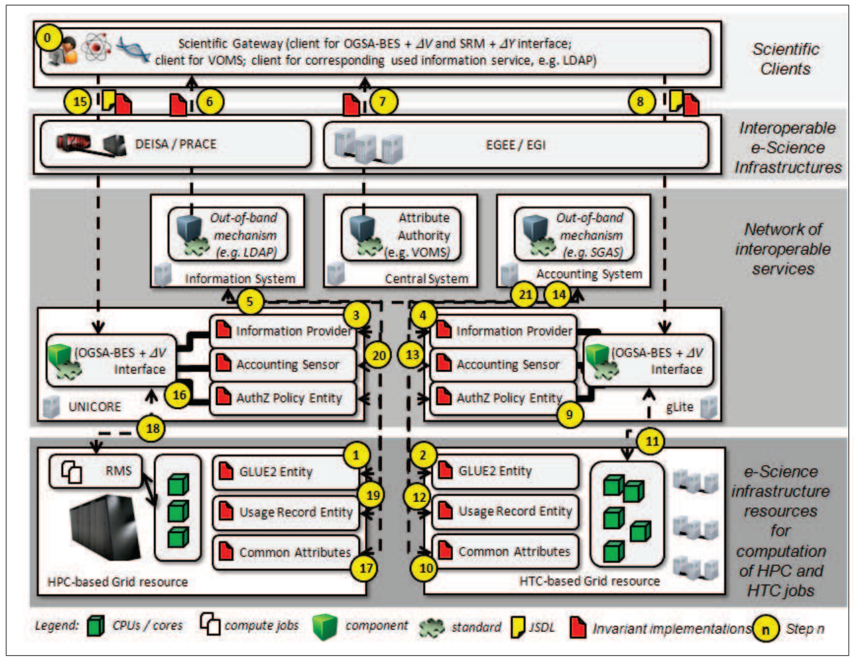


Figure 6.7: IIRM Infrastructure Integration Constraints applied to EGEE/EGI and DEISA/PRACE.

that obtains valid credentials from an AA like the VOMS system. The client require methods to query an information service of interest (e.g. via LDAP) in order to obtain the status of the e-Science infrastructures and their offered services (and underlying resource) capabilities.

Before the e-Scientist is even able to work with the interoperable infrastructures, the existence of computational resources and their capabilities are exposed in a consistent manner. The global information invariant comes into play that raises the demand to expose such properties via GLUE2. As the example shows, a HPC resource is described with a GLUE2 entity (1) and another HTC resource with another GLUE2 entity (2). These entities are exposed via corresponding information providers (3 and 4) to an information system (5). Due to scalability reasons we do not force specific WS-based interfaces as part of the IIRM and rather refer to a mechanism that is not further specified but that provides a scalable solution (e.g. based on LDAP). This satisfies the first invariant so that *'basic semantic interoperability'* is in place in the e-Science infrastructure ecosystem between EGEE/EGI and DEISA/PRACE. This basic semantic interoperability is essential since it clarifies the use of terms across infrastructures that previously have described their resources with different terms, languages, or information models.

The constraint of having one common information model enables interoperability in terms of fundamental information exchange and gives clarity thus avoiding semantic loss of information where one proprietary model might be transferred into another proprietary model and vice versa (i.e. transformation logic, cf. Chapter 3). Based on this semantic interoperability, the e-Scientist is able to use his client to query the information system (6) in order to search for required computational resources (across multiple infrastructures) that are accurately described with GLUE2 entities.

When the systems have been identified, the right credentials can be obtained from the AA that in our case is a VOMS server (7) augmented with a SAML interface as published in [302]. This server releases signed attribute statements about end-users (e.g. project or VO membership, role possession, etc.) either encoded in SAML or in X.509 proxies depending on the relevant systems of interest found in the query of the information service. This security setup information is part of the GLUE2-based information about each service and thus available before the call to the relevant AA in context. The interoperability of the interface for the AA is not as important as the agreement of the attribute formats stating the security content. It is essential that the attribute statements are in the same common security attribute format in order to achieve *'basic authorization interoperability'*. This lays the foundation to enforce the global authorization attributes invariant later during authorization decisions within the service layer.

After the aforementioned basic steps are taken, the e-Scientist uses a HTC resource with an embarrassingly parallel job. According to the IIRM design, the e-Scientist re-uses elements of obtained resource description elements based on GLUE2 to form an enhanced JSDL document (8) that is submitted to an enhanced OGSA-BES implementation within gLite. A missing link between GLUE2 and JSDL is used to form more accurate resource requirement statements in JSDL elements actually based on GLUE2 elements. One example of the enhancements of the JSDL document is the use of GLUE2 elements in its resource description section. Along with this submission is the security credential obtained from the VOMS server, such as an X.509 proxy credential with attribute statements that conforms to the common set of attributes stating the VO and role of the e-Scientist. Before the execution of the job is performed, the authorization policy framework of gLite (i.e. gJAF, or more recently Argus) (9) is responsible to extract the attributes from the credential and to perform a check of the attributes against the security policy. The constraints is set that also the policy definitions must be in the same common attribute format (10), the access is granted (11) when the e-Scientist obtained the credential from a trusted VOMS server that also then satisfies the global authorization attribute invariant. When the execution of the HTC job is finished a usage record entity (12) is created and forwarded

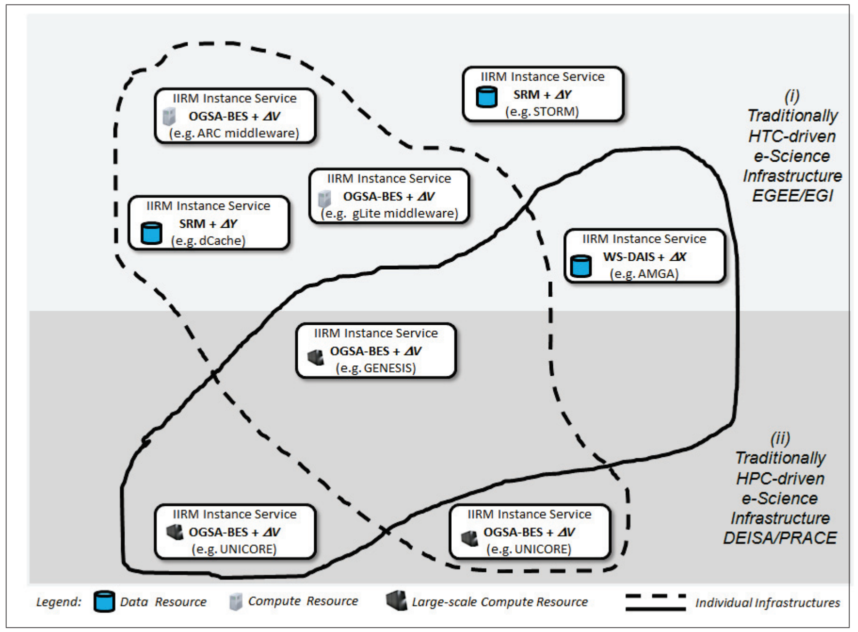


Figure 6.8: IIRM adoptions enable individually formed infrastructures across EGEE/EGI and DEISA/PRACE.

via a dedicated accounting sensor (13) to a central accounting system (14) such as SGAS [279]. This basically concludes the HTC execution part of the greater scientific workflow following the design pattern algorithm introduced in Chapter 5.

As [259] reveals, the results of the HTC-based workflows are evaluated, often also manually, before the best results are considered to be used in application submissions to HPC-based workflow elements. As these steps are in many cases very similar like the HTC steps due to the interoperability and common invariants usage, these parts are described shorter to avoid the repetition of details. Starting with the HPC-based workflow, the scientific gateway is used to submit an enhanced JSDL (15) with a different credential set (i.e. SAML assertion obtained from AA) but that encodes the same attribute statements as used in the HTC setup, including the same security information. The same context-based identity is used, but encoded in a different way while the real security-relevant information (i.e. security attributes) remains the same. This is an important aspect since it is related to the trust of end-users into the infrastructure and the feedback about their identity that remains the same independent of which infrastructure is actually used underneath.

Afterwards, the authorization policy entity (16) of the UNICORE middleware extracts the attributes encoded in a signed SAML assertion and performs an authorization decision based on the common set of attributes (17). This satisfies the *'global authorization attribute invariant'* that here avoids the need to (often manually) setup dedicated authorization policy elements for end-users of other e-Science infrastructures. Such a manual setup is often applied as in the GIN setups [256] that reveal that they are often hard to maintain, time-consuming and in many cases also error-prone leaving even security risks if users that are not allowed to use the infrastructure anymore are not removed.

When the access is granted (18) based on a match of the provided end-user attributes in the SAML assertion with the defined XACML security policy, the HPC execution is started. In many cases, a resource management system (cf. Definition 11) is used to execute in this step an MPI-based parallel HPC application. Finally, after the job run, a usage record entity (19) is created and forwarded via an accounting sensor (20) to the same relevant accounting system (21) as stated above (e.g. SGAS). This finally concludes the HPC workflow part and as such the whole scientific workflow that covers the concrete production e-Science infrastructures EGEE/EGI as well as DEISA/PRACE. The IIRM invariants and infrastructure integration constraints verify that the IIRM and its reference architecture is applicable to the concrete architectures of EGEE/EGI and DEISA/PRACE.

'Since many of the ESFRI projects have stated that they have a clear need to make use of several of the existing European e-Infrastructures, improving the interoperability between these structures will have a definite added-value for all the user communities' [161]. As shown in Figure 6.8, EGEE/EGI and DEISA/PRACE thus have an emerging interoperability between them that in turn is getting closer to the vision so that scientists can actually create *'individually formed infrastructures'* as defined in Definition 20 across both of them. Both Figure 6.7 and Figure 6.8 and its aforementioned descriptions provide evidence about one of the key impacts of that thesis enabling technically individual infrastructures between EGEE/EGI and DEISA/PRACE.

But interoperability is in this context only achieved on the technical level. Common usage policies need to be also defined while Figure 6.8 present one possible technical deployment example. Their clients use seamlessly the heterogenous resources for scientific research as illustrated in Figure 6.8. A comparison with this situation to the ideal situation, provides the following results. An ideal situation might be that there is only one middleware across both European infrastructures that in turn would make the problem of interoperability irrelevant. That is not the case and and this ideal situation will not emerge in the near term, mostly because many end-users are already using technologies in the past and keep using them instead of switching technologies.

Finally, further evidence on e-Science impact is provided by analyzing to which extend the thesis satisfies the following requirements. Firstly, the technical foundation for end-users is provided by the IIRM to choose resources from EGEE/EGI or DEISA/PRACE as defined in Definition 73. The refinement concepts also have thus satisfied Definition 74. The thesis approach avoids the use of transformation logic, which normally leads to performance reductions especially in cross-infrastructure use cases. End-users can use the HPC extensions for efficient executions on HPC resources essentially not having performance reductions, because optimizations (e.g. shapes, network topologies) can be used. This satisfies the requirement as defined in Definition 75.

6.2.3 US and other e-Science Infrastructures Setups

The last section focused on the thesis results that affect Europe while this section describes how the IIRM activities contribute to interoperability beyond Europe meaning mainly US as well as Japan. This directly addresses one critical 6th e-Concertation meeting outcome: *'Links to the rest of the world-leading e-Infrastructures need to be streamlined'* [133]. This section aims to offer some solutions for such an international link in the particular context of concrete production e-Science infrastructures (aka e-Infrastructures) as shown in Figure 6.9.

That this is important is best reflected in the e-IRG white paper 2009 with *'The engagement in worldwide collaborations between the various e-Infrastructures will increase cross fertilization of* *novel ideas across large scientific communities and harmonise policies and best practices between transcontinental large scale e-Infrastructures towards the worldwide Knowledge Society'* [206].

The principle of differentiating between *'architecture and its implementation'* is also followed as part of the XSEDE project [29]. XSEDE is the follow-on project of the known TeraGrid infrastructure in the US that was traditionally a rather HPC-driven infrastructure (cf. Definition 16). The precise architecture of the XSEDE infrastructure will evolve over time, but large elements of the architecture are considered to be specified by open standards that are all in-line with the IIRM and its concepts [235]. With the segment-based process implementation described in earlier parts of this chapter, the close collaboration with members of GENESIS via OGF GIN and PGI [270] lead to the collaborative XSEDE proposal including the adoption of open standards in the architectural design.

The IIRM thus influenced the XSEDE architecture design starting from the use of SAML and XACML for security and the use of OGSA-BES and JSDL in terms of Grid job submission as illustrated in Figure 6.10 published in [235]. The *'XSEDE Enterprise Services'* are based on open-standards and mandatory to be installed at every major XSEDE resource. The *'Community Provided Services'* are complementary optional installations. As the author is part of the architectural design team, the architecture will be influenced to a large extend by the IIRM design, including its reference architecture. The impact on the architecture is another contribution of this thesis to the open standards-based XSEDE according to IIRM as [235] reveals. But XSEDE also aims to adopt some standards that have been intentionally left out of this thesis scope, but that can be augmented to the IIRM without breaking its concepts over time. Examples include the Resource Namespace Specification (RNS) [243] that can be used in conjunction with the core building blocks of the IIRM reference architecture.

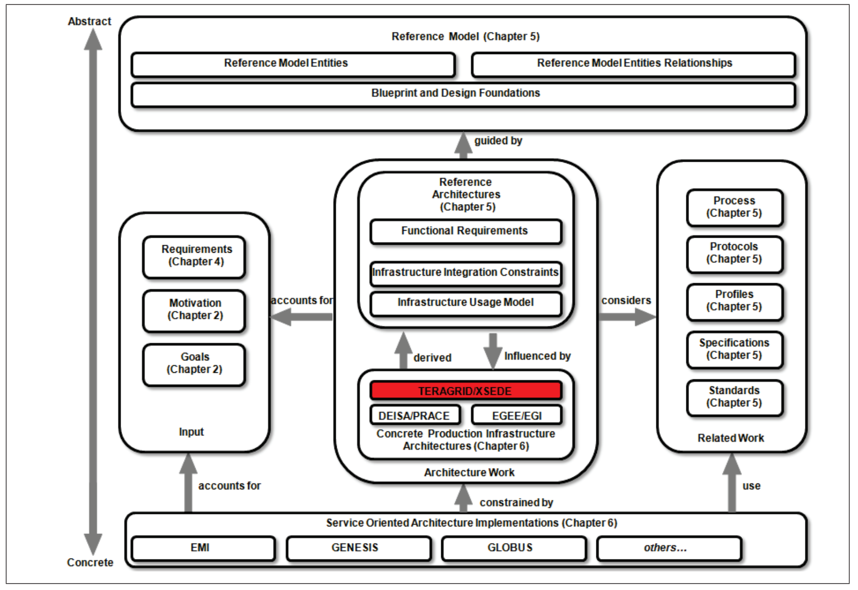


Figure 6.9: US and other e-Science infrastructures setup overview.

In terms of US-based infrastructures, the OSG, as a traditional HTC-driven infrastructure, is also important. Plans exist that OSG will become a status like *'sub-contractor of XSEDE'* such as other possible resource providers in the US. Mechanisms will be then established that enable the use of XSEDE but inherently forward jobs to OSG for computation. XSEDE thus becomes a hybrid infrastructure (cf. Definition 18) in the next years, since also HTC-driven campus resources will be integrated over time. It is important to acknowledge that this thesis can only partly influence the deployment situation of such infrastructures from a technical perspective. The final deployments, also in terms of which technologies will be deployed might differ, but here we outline the planned deployment situation as published in [235].

The key middleware technologies that are currently considered to implement the architecture are GENESIS and UNICORE augmented with several other tools like Globus Online data transfers [39]. As [235] reveals, GENESIS is planned to be deployed on campusses while UNICORE is planned to be used at major US supercomputing sites. Another key impact of this thesis is therefore the numerous contributions to the standards-based software UNICORE [257, 220, 302, 284] (cf. Section 1.4) that are considered for the XSEDE architecture for this particular reason. The aforementioned section about adoptions reveal, both GENESIS and UNICORE adopt the core building blocks of the IIRM and considering the adoptions of the concept refinements via PGI participation [270]. This evidence points to the fact that the IIRM and its concepts also influence the concrete architecture of the production e-Science infrastructure TeraGrid/XSEDE. Sites that in parallel deploy Globus might use its emerging OGSA-BES implementation by IGE via GridSAM [44] in order to achieve interoperability with TeraGrid/XSEDE sites and with European infrastructures DEISA/PRACE and even EGEE/EGI as illustrated in Figure 6.11.

The European interoperability is thus extended to TeraGrid/XSEDE as another concrete production e-Science infrastructure (cf. Definition 5) that is enabling a *'network of interoperable Grid services'* according to Definition 15. Based on the fact that Europe and US adopt middle-

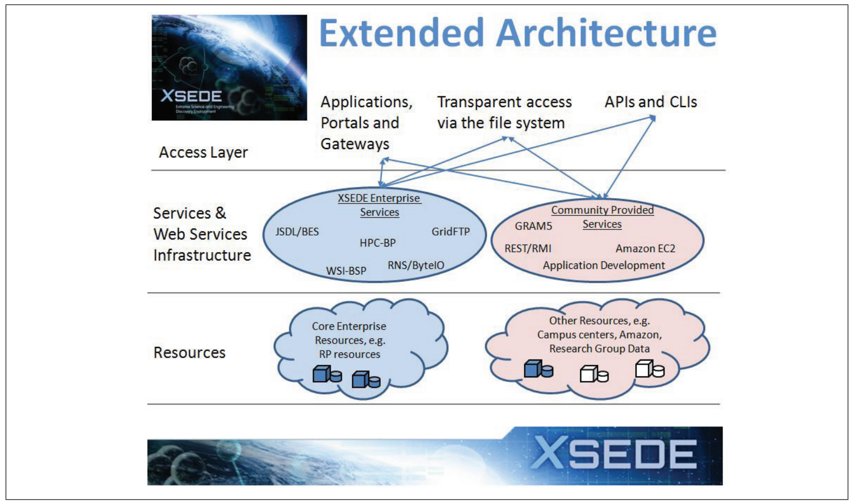


Figure 6.10: Emerging US XSEDE infrastructure architecture with IIRM core building blocks [29].

ware that adopts the IIRM core building blocks, e-Scientists are able to create an individually formed infrastructure (cf. Definition 20). Such an infrastructure increases the effectiveness of e-Scientists (cf. Defnition 10) in an unprecedented manner enabling seamless technical resource usage across TeraGrid/XSEDE, DEISA/PRACE, as well as EGEE/EGI. This ideal situation is illustrated in Figure 6.11 while the deployment in US is in progress and the situation in Europe is getting much more fragmented (e.g. UNICORE on EGEE/EGI in various NGIs). Given the secure environments of DEISA/PRACE, there is currently only UNICORE and some tools of Globus planned to be deployed such as an implementation of GridFTP [109] that is also in-line with the core building blocks of this thesis. With all the aforementioned key thesis contributions interoperability is ensured and the thesis contributes to the enabling of individually formed infrastructures across EU and US.

The IIRM and its associated reference architecture elements are already partly deployed on the US and European production e-Science infrastructures. Based on this technical foundation, the middleware boundaries are lowedred and e-Scientists are able to form individual infrastructures as needed using the resources they want. Nevertheless, it is important to mention that usage policies must be set in place to enable them with the seamless use (e.g. computing time) of all the illustrated infrastructures or the submission of a peer-reviewed grant in DEISA/PRACE has been succesful.

Apart from the US and as part of the seven segment-based process described earlier in this chapter, a collaboration with the Japanese infrastructure activities initially called NAREGI [221] and then Resources liNKage for E-scIence (RENKEI) [83] lead to IIRM adoptions as described in

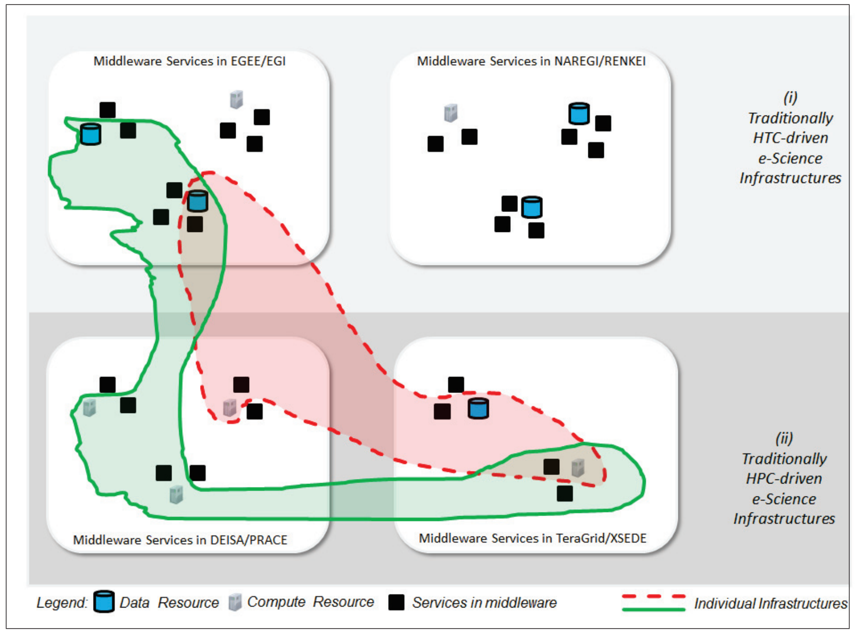


Figure 6.11: IIRM-enabled individual infrastructures across the US and Europe.

[84]. NAREGI/RENKEI members have been participating in GIN and PGI activities [270] while also their infrastructure is able to work together with the European and US infrastructures as described as follows. Based on the IIRM core building blocks adoptions within RENKEI [84], e-Scientists are able to create *'individually formed infrastructures'* across Europe, US, and Japan. The latter interoperability is still emerging and many sites of the infrastructure are not yet fully interoperable thus providing only a limited adoption of the IIRM that will be broaden during the course of the next years following the plan outlined in [84]. Recent activities in RENKEI reveal [84] that it is majorly orientated towards the use of HTC (but with more and more HPC resources over time) thus influencing the Figure 6.11 how this infrastructure is classified.

This evidence on the e-Science infrastructure impact is underpinned to which extend the thesis satisfies the requirements in the context of the aforementioned e-Science infrastructure architectures. The technical foundation for scientific end-users to choose resources from TeraGrid/XSEDE in the US or from the DEISA/PRACE infrastructure as defined in Definition 73 is provided thus *'enhancing users choice'*. The infrastructure interoperability setup is extended to HTC-driven infrastructures (cf. Definition 17) such as EGEE/EGI with the UMD deployments [14]. The thesis thus contributed to the vision where technical infrastructure boundaries, apart from the usage policies, do not exist. Comparing this architecture setup to the *'optimal setup'*, there must be still work on harmonizing the usage policies since computational time on TeraGrid/XSEDE as well as DEISA/PRACE is still subject of a peer-review process while the access to the EGEE/EGI infrastructure is different by joining a dedicated scientific VO.

In the particular interoperability setup between DEISA/PRACE (i.e. UNICORE) and TeraGrid/XSEDE (i.e. GENESIS and UNICORE) both infrastructures aim to use the refinement concepts of Chapter 5 with adoptions in GENESIS and UNICORE after PGI standardization [270] thus also in-line with Definition 74. This is because the approach around the reference architecture (e.g. using OGSA-BES) avoids the use of transformation logic (cf. Definition 27), which normally leads to performance reductions especially in cross-infrastructure use cases.

End-users can use then the HPC extensions for efficient executions on HPC resources as part of TeraGrid/XSEDE as well as DEISA/PRACE that are both rather HPC-driven infrastructures (cf. Definition 16). End-users that use this interoperability setup have not performance reductions, because they can take advantage of optimizations (e.g. shapes, task/core mappings, etc.). These optimizations, which are commonly performed on HPC resources, satisfy the requirement as defined in Definition 75.

6.2.4 Related Models for e-Science Applications

Figure 6.12 provides an overview of the approach throughout the thesis and illustrates elements related to concrete architectures and SOA-based architecture implementations. But the figure also reveals another element named as *'Related Models'* (marked in red) that is addressed in this brief section with a particular focus on those models that support e-Science applications and that have some relationships with the core building blocks described in this thesis.

Many related models can be directly used with the reference model and its associated architecture work, because the thesis provides concrete basic specifications as part of its reference architecture. The detailed foundational reference architecture can be considered by related models and is as such another contribution of this thesis. The wide variety of e-Science applications (cf. Definition 9) can also be applications of the IIRM and its associated reference architecture when the core building blocks described in Chapter 5 are used. The foundational character of the key areas (i.e. compute, data, security, and information) of the core building blocks, enables their use by related models such as *'higher level frameworks'*.

In the given context of production e-Science infrastructures (cf. Definition 5), the *'Simple*

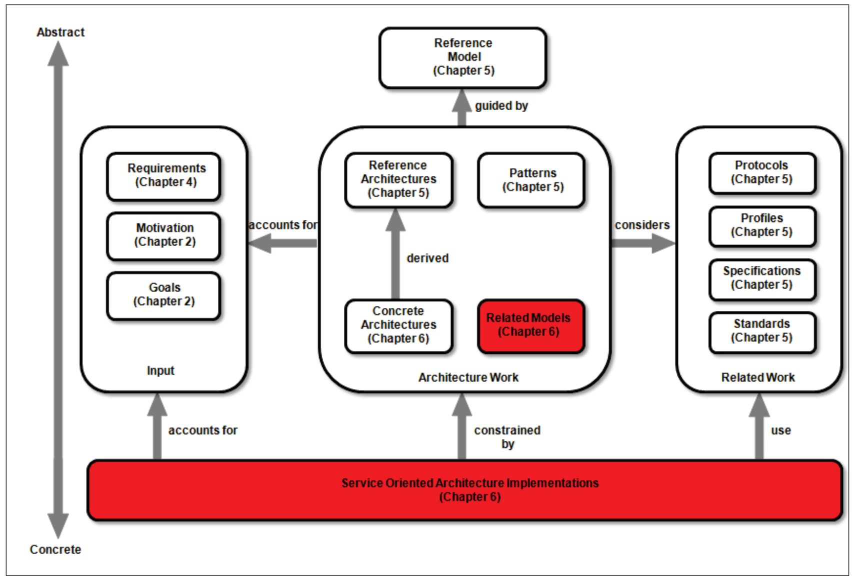


Figure 6.12: Related Models for e-Science Applications Overview.

API for Grid Applications (SAGA) Framework' [202] represents one concrete example of such a related model to the IIRM and its associated reference architecture. The term SAGA stands for two aspects. Firstly, SAGA is an open standard [186] to promote *'Grid interoperability on the application level'* developed by the OGF, but is not considered as an IIRM core building block in this thesis. Secondly, the so-called *'SAGA framework'* is actively developed by a team that adopts the SAGA standard as described in [202].

There are different reasons why the SAGA framework is relevant in the context of the thesis. Most notably, it is a framework that enables a high-level programming abstraction, which significantly facilitates the development and deployment of e-Science applications (cf. Definition 9). As such it provides a lot of application patterns (e.g. map-reduce, replication, parameter studies, etc. [287]) that are useful for e-Scientists thus lowering the barrier in using production e-Science infrastructures in general.

Apart from being a high-level application framework, [287] reveals that it can work on top of the foundational core building blocks of the IIRM reference architecture (e.g. OGSA-BES, JSDL, etc.). Although being previously bounded to proprietary interfaces of Grid middleware systems as described in [202], in [287], the SAGA framework is thus extended towards OGSA-BES and other related open standards that are in-line with the IIRM core building blocks. The idea of SAGA is to expose the same functionality as standards like OGSA-BES or JSDL, but to provide additional program level simplifications and abstractions to the e-Scientists in order to hide the complexity of these underlying interfaces [287]. SAGA is not only a standard (cf. Definition 14), but also an application framework that provides many features that in turn reflects the classification of this work as one of the related models (cf. Figure 6.4).

The SAGA framework benefits from the IIRM reference architecture by requiring to support open standard protocols (and possibly their refinements over time) for different middleware systems instead of numerous different adapters for each middleware (cf. adapter approach in Chapter 3) used in production e-Science infrastructures. User of the IIRM can benefit from using the wide variety of SAGA application patterns with scientific applications on top of the rather basic IIRM architecture.

Collaborations within GIN lead to initial prototypes, exploiting the mutual relationships with e-Science applications in order to provide scientific innovation through the IIRM. The IIRM and its architecture definition is a first step to provide a foundational architecture for higher-level application frameworks such as SAGA. The aforedescribed facts point to another contribution of this thesis that is the established foundation for higher-level application frameworks such as SAGA [287]. More related models (e.g. workflow engines) are likely to appear from the wide variety of application tools and models of ESFRI RIs using domain-specific tools. One concrete example is the WebLicht [192] workflow tool used in computational linguistics within the ESFRI Common Language Resources and Technology Infrastructure (CLARIN) infrastructure [306] that can be augmented with clients for the core building blocks in the reference architecture to take advantage of the computational power available in EGEE/EGI and DEISA/PRACE today.

6.3 Architecture Implementation for the WISDOM Applications

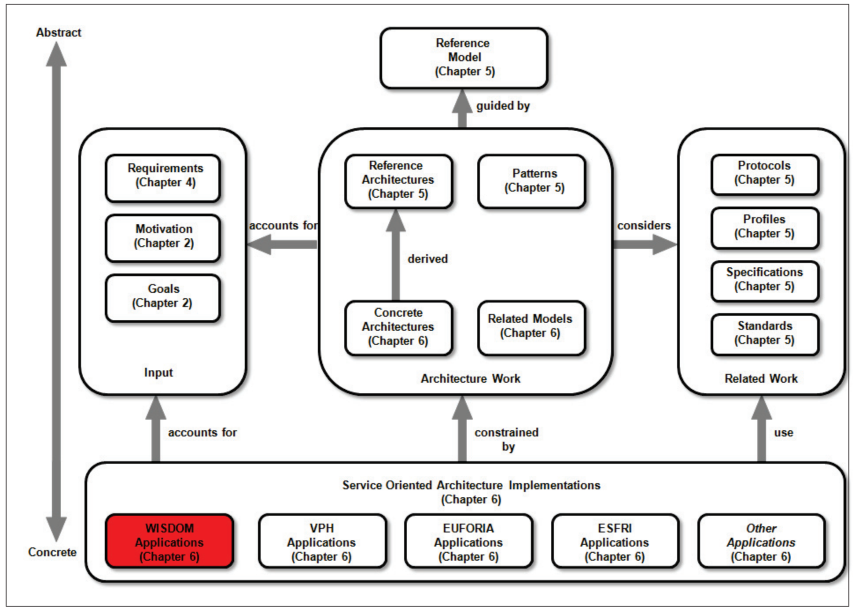


Figure 6.13: SOA implementation for WISDOM applications constrained by the concrete reference architecture.

This section describes concrete SOA-based implementation details about the scientific case study named *Wide In Silico Docking on Malaria (WISDOM)* [135] that was conducted during the definition of the proposed reference model. It contributed enormously to the findings of this thesis and many aspects of it are published in [259] and [272]. Figure 6.13 provides an overview of how the case study fits into the overall reference model approach and its implementations.

The path of the scientific investigation throughout the case study is also represented in the structure of this section. An introduction to the WISDOM initiative and its basic framework is given in order to understand how WISDOM e-Scientists use production e-Science infrastructures. The overall scientific WISDOM application workflows requires more than one e-Science infrastructure.

Both the architectural analysis and the scientific application analysis set the foundation for the critical academic analysis to explain the limitations of the basic WISDOM framework. As a key contribution of this section insights into how those identified limitations are solved by using the IIRM and its associated reference architecture implementations are given. The necessary IIRM reference architecture core building blocks are already emerging on production e-Science infrastructures as the previous sections revealed.

This section aims to verify that the reference model is applicable to the WISDOM scientific workflows that seek to take advantage of more than one production e-Science infrastructure. Many prototype developments and studies contributed to the case study thanks to the fruitful collaboration between the WISDOM community and the OMII-Europe project [69]. These results have been presented at various events, such as the Supercomputing 2007 in Reno [94] at the JSC booth as part of OGF GIN demonstrations.

6.3.1 Basic Framework of the WISDOM Initiative

The fundamental goal of the WISDOM initiative [135] is to support a broad drug discovery process illustrated in Figure 6.14 using specific computational methods (i.e. in silico). The approach is based on the so-called *'in silico drug discovery process'* [135], which uses computational simulations to speed up the identification and characterization of potential new drugs. The focus here is on the docking of the 3D structures of plasmepsin and small molecules [135], including their simulation over time once docked.

The SHARE project [86] indicates that pooling knowledge and computer technology to do in silico drug discovery can correspond to savings of about 300 million US dollars. It further states that this approach can reduce the development time of a new drug approximately by two years per drug, which also represents a crucial scientific innovation from the case study. Pharmaceutical research is constantly looking for ways of reducing the time and costs involved in drug development. The interoperability of e-Science infrastructures breaks institutional boundaries to help achieve these goals by providing access to more computational capabilities than one single infrastructure can provide. Such interoperable e-Science infrastructures bear a lot of potential to perform cheaper and faster drug discovery using in silico methods.

In contrast to the broader SHARE project, the WISDOM initiative has a more focused strategy aimed at developing new drugs for malaria. WISDOM scientists used only the EGEE/EGI e-Science infrastructure via gLite for large-scale in silico docking methods, but mechanisms as described in [259] have been established that make use of the DEISA/PRACE infrastructure, too. The basic WISDOM framework was created to enable the interoperation between EGEE/EGI and DEISA/PRACE illustrated in Figure 6.15.

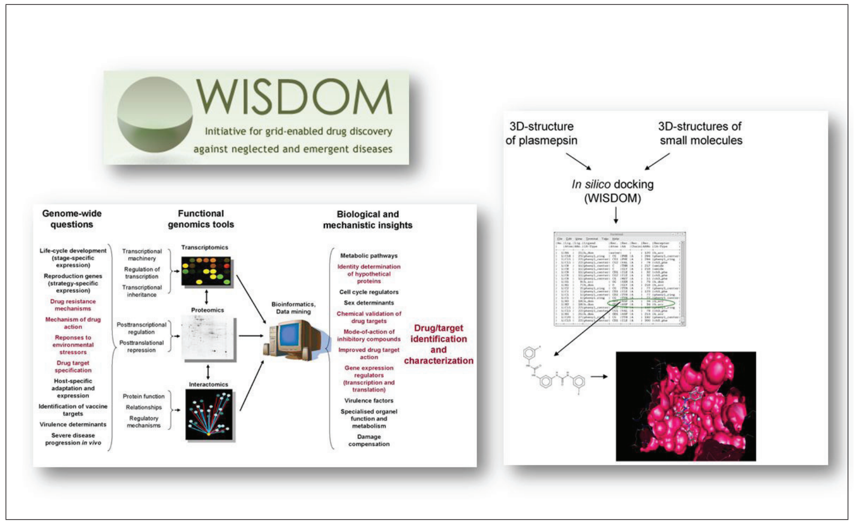


Figure 6.14: The WISDOM initiative focusses on drug/target identification and characterization [135].

As shown in Figure 6.15, the WISDOM e-Scientists use a technology called GridSphere that in turn is internally using the Vine Toolkit [278]. This toolkit provides adapter for various middleware technologies and is based on Java representing a high-level API for managing jobs on specific target sites. The environments in which Vine can be deployed are Java Web Start, Java Servlet 2.3, and Java Portlet 1.0. It is optimized to run in Browser-based setups that acts as the client tool for the GridSphere portal. The Vine version supports gLite and UNICORE (e.g. SSL key generation, and job-based management, etc.), but also VOMS (e.g. proxies, registering and un-registering users, etc.). In order to send jobs to a middleware, Vine has client classes for each middleware. As part of a production installation, it has to be deployed as a WS application deployed in a Web container. The URI of this container and the specific WISDOM portal is then a fixed location that e-Scientist can use to perform their daily scientific application runs. Some features that are used for the use case applications are the proprietary UNICORE interface for job management, queries of up to date job statuses, VOMS and proxy generation.

Also the Arda Metadata Grid Application (AMGA) metadata catalogue [280] is used as shown in Figure 6.15. AMGA is part of the EMI project, representing a metadata service that enables users to attach metadata information to files. AMGA is used as a general access tool to relational databases on the Grid, and has been also significantly used in WISDOM to attach metadata for scientific results from computational Grid jobs.

Figure 6.15 summarizes the pre-production setup of the case study enabling GridSphere to submit jobs, in this particular setup, to the EGEE/EGI and DEISA/PRACE infrastructures.

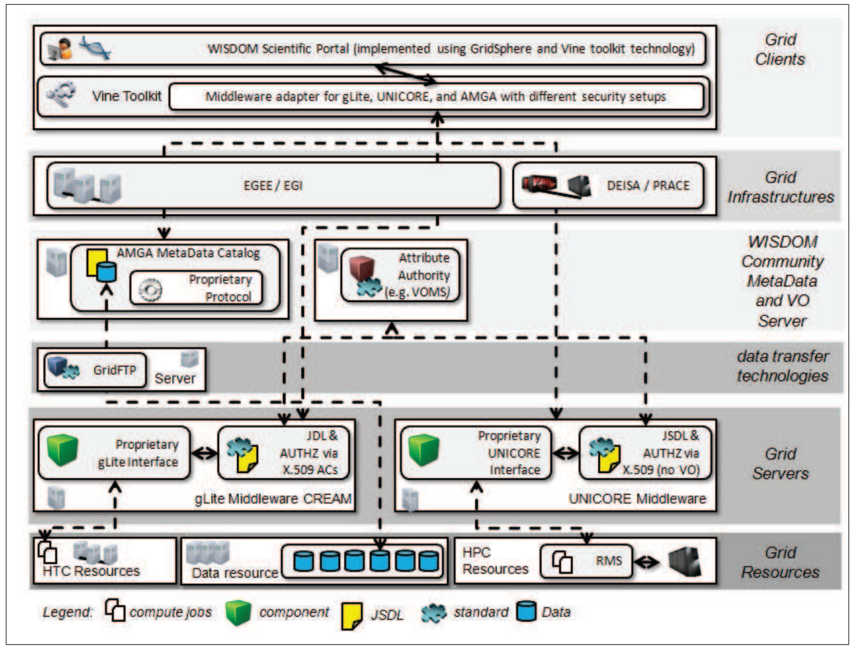


Figure 6.15: The basic WISDOM framework uses technologies of EGEE/EGI and DEISA/PRACE.

6.3.2 Scientific Applications of the Bio-informatics Domain

Whereas the previous section focused on the initial WISDOM framework from a technical perspective, this section reveals insights about the WISDOM scientific workflow from an end-user perspective. The overall scientific bio-informatics workflow is illustrated in Figure 6.16 (using elements of [259]) and illustrates three fundamental steps using two different infrastructures.

The first step uses the scientific application package FlexX [30] depending on which licenses are available at which sites. The commercial company BioSolveIT [30] also kindly provided free licenses for FlexX to enable parts of this case study. But sometimes also AutoDock [187] is used that is a free software application package. These software packages are molecular docking applications used to check whether one molecule is able to bind to another. The whole process is described in [135], including information on its greater impact on the whole drug discovery process. Important for the case study of this thesis is that the EGEE/EGI infrastructure is used with the aformentioned applications with the HTC computational paradigm. At the time of the studies relevant EGEE/EGI resources were accessible by gLite only using mostly HTC-based resources.

The output of the aforementioned step 1 is only an intermediate result as shown in Figure 6.16. It is a list of best chemical compounds that are potential drugs and thus not the final solution to performing the in vitro (i.e. real laboratory tests) and subsequent in vivo (i.e. living organism tests) steps in laboratories. The results of the HTC workflow part executions need

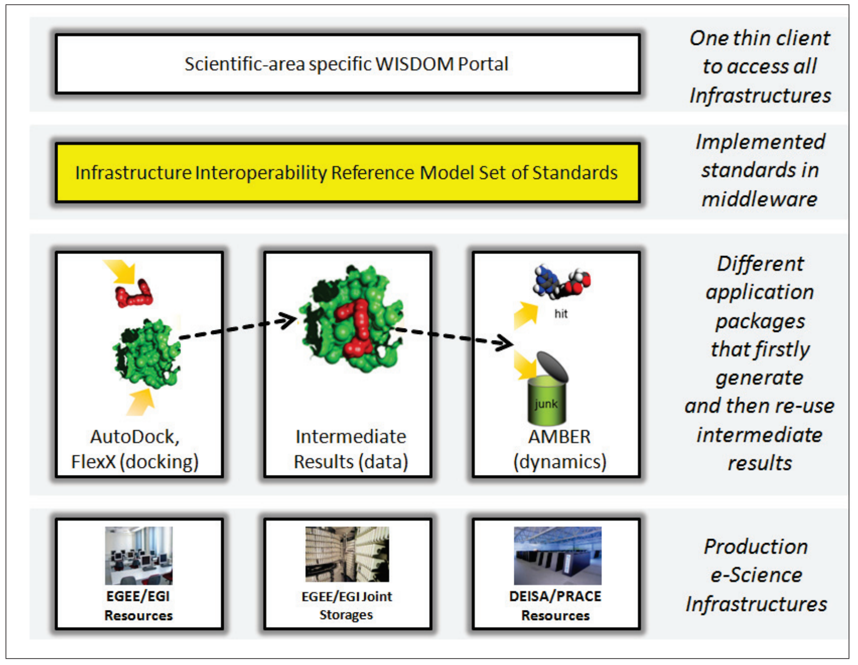


Figure 6.16: Accelerate Drug Discovery by using the core building blocks of the IIRM reference architecture.

No	e-Science Infrastructure	Setup	e-Science Application Demands
1	EGEE/EGI	Section 6.2.2	autodock and FlexX application embarassingly parallel runs requiring trivial computational power; use of HTC resources feasible;
2	EGEE/EGI	Section 6.2.2	data storage to store intermediate results; for each result metadata needs to be stored;
3	DEISA/PRACE	Section 6.2.2	massively parallelized AMBER code psander using HPC resources; required are data structure tranformations before and after the job run;

Table 6.4: Summary of the e-Science infrastructure setups of the WISDOM case study.

to be stored on an infrastructure while EGEE/EGI provides several data storages for the WISDOM VO. There are a massive amount of HTC runs in order to find potential drugs on a year perspective and thus each result need to be properly described with metadata. This metadata needs to describe which intermediate results are stored and which input data has been used.

A scientific method developed by Rastelli et al. [162] is used with molecular dynamics (MD) computations to refine this best compound list. But MD computations are very computationally intensive, so there is a lot of potential in using the Assisted Model Building with Energy Refinement (AMBER) [242] MD package within DEISA/PRACE. As computational time on rare HPC resources is limited, manual scientific investigation is required to pick those promising input data out of our intermediate result sets that actually make sense to be computed from a scientific perspective.

AMBER itself consists of roughly 80 programs with different executables, parallel programs using MPI and even smaller programs that are data format converters. The setup of AMBER is non-trivial especially since several different AMBER applications are usually executed as a sequence (not a Grid workflow) as published in [193].

Discussions with WISDOM e-Scientists revealed the high potential of using this workflow with the IIRM and its reference architecture accessible via one dedicated WISDOM GridSphere portal. This case study focuses on a very thin client layer access to the IIRM, including still proper security mechanisms required to use real existing production e-Science infrastructures. In contrast to subsequent case studies, the focus relies here on the requirement of manual selection of input data for data transfer and the complexities that are experienced with broad application packages such as AMBER.

Many scientific bio-informatics applications are very similiar to AMBER. Instead of AMBER, also other MD application suites that take advantage of parallel programming can be used with HPC Grid resources such as NAMD [236] or GROMACS [214].

Table 6.4 summarizes the production e-Science infrastructure setup required by this specific bio-informatics case study. It clarifies the question which e-Science infrastructure setup is used with what concrete types of e-Science applications while the previous sections are referenced for general infrastructure setup information in context. In this case study, the WISDOM e-Scientists use various types of executables with different infrastructure resources (cf. Definition 6).

6.3.3 Academic Analysis and Production Infrastructure Setup Experience

This paragraph analyses the basic architectural WISDOM framework using the previously mentioned scientific applications in order to understand limitations of the current approach.

No.	Limitation short description
(a)	additional Layer Approach (transformation logic)
(b)	no manual or client-initiated data-staging
(c)	no job sequences (not workflows) support

Table 6.5: Overview of limitations after academic analysis of the WISDOM framework.

from WISDOM e-Scientists. The basic framework was used in production e-Science infrastructure use cases and the outcome of the analysis points to several limitations of the basic framework that are summarized in Table 6.5.

The analysis of the general approach of the framework is the first step pointing to overall drawbacks of the approach before other details are presented of how specific concepts are needed but not supported. Chapter 3 provides a classification of general approaches to achieve interoperability. Among concepts like Adapter, Gateway, Mediator, and Neutral Bridge, there is also the Additional Layer approach.

The first major limitation in the basic WISDOM framework is the use of the *'(a) additional layer approach'* to enable interoperability between e-Science infrastructures in general and their corresponding Grid middleware gLite and UNICORE in particular. In the WISDOM case study, the additional layer is represented by the Vine toolkit that provides specific adapters for all Grid middleware systems and AMGA.

Apart from this general approach, a more fine-grained academic analysis of lessons learned provides more clarity and points to specific challenges that are later addressed to improve the efficiency of the WISDOM framework using the IIRM and its concepts. When applying the WISDOM scientific workflow, a missing concept named as *'(b) no manual or client-initiated data-staging'* was identified. This appeared after the first application run was performed on the EGEE/EGI infrastructure and the intermediate results are kept in a joint storage. E-Scientists would like to use only parts of the intermediate data after their evaluation, but there is no way in which it can seamlessly be transfered into the job directory for further DEISA/PRACE refinements using MD. As there is no convenient method, e-Scientists often copied the same intermediate results to another location that in turn was used with automatic data-staging capabilities with UNICORE on DEISA/PRACE. The duplication of even intermediate results in storage and the selection process are cumbersome and complicated. This limitation requires one major improvement of their work process.

Another identified limitation is that middleware provides *'(c) no job sequences support'*. The scientific application package AMBER consists of roughly 80 different programs of which a few are used only to transform input and output formats between different programs that need to run before or after specific programs, but in the same working directory. This is different from coarse-grained workflows where the job can reside in different locations and must not be executed in the same working directory. During the analysis, there was no way other than writing huge error-prone manual UNIX scripts that call different programs while in between the MPI program was executed. Job sequences that allow for pre- and post-processing of the main executable (that could run in parallel) were considered here as a major improvement.

Although the aforementioned limitations have been the most important ones identified there are other limitations (e.g. better HPC support) that overlap with those that will be explained in other use case studies (e.g. VPH). The focus in this case study is thus on those listed in Table 6.5.

6.3.4 Reference Model Impact and Applicability

The goal of the here presented concrete reference architecture instance is to improve the e-Science methods in the WISDOM initiative and thus significantly accelerate the drug discovery process by seamlessly using EGEE/EGI and DEISA/PRACE, as shown in Figure 6.17. This illustration represents a specific *'architecture instance'* of the IIRM and provides insights into the core building blocks of the reference architecture. As the subsequent case studies will reveal, several concept refinements (i.e. Δ) for the core building blocks are needed in order to run scientific applications on resources either more efficiently in terms of resource usage, or more seamlessly in terms of security. Improved open standards of the IIRM lead to a maintainable and thus sustainable WISDOM framework for bio-informatic scientists providing HTC and HPC resources.

The case study uses emerging IIRM core building blocks providing lessons learned that contributed significantly to the concepts documented in Chapter 5. An overview of which concrete core building blocks are needed as part of the WISDOM case study is in Table 6.6, following which concrete refinement concepts can be applied in order to overcome the limitations identified in the previous section. This table and Figure 6.17 illustrates another key contribution of this thesis by using IIRM core building blocks and their refinements with a real scientific use case.

While Table 6.6 provides a general overview of the used core building blocks of the reference architecture, the next paragraphs provide a step-wise walkthrough of Figure 6.17 in order

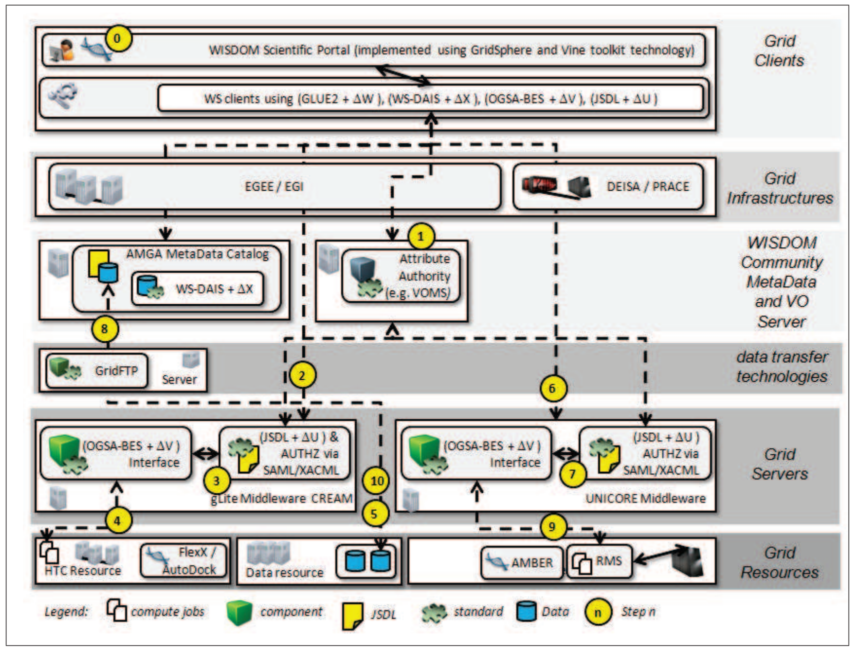


Figure 6.17: Enhanced WISDOM framework using the IIRM with bio-informatic applications.

Core building blocks	Specific usage in WISDOM
WS-DAIS + ΔX	Storing/retrieving metadata and file locations in database
OGSA-BES + ΔV	WS submission to create a Grid job using a middleware systems
JSDL + ΔU	FlexX, AutoDock, and AMBER application job description; sequences
GLUE2 + ΔW	Information about installed modules on HPC resources
SAML	Encoding (i.e. SAML assertions) of WISDOM end-user security attributes
WS-Security	Mechanism to transport SAML assertions in OGSA-BES WS messages
XACML	Security policy for end-users (also from WISDOM)
GridFTP	Transfer from computed data to be used by AutoDock, FlexX, and AMBER
X.509	Security X.509 certificate provided by each WISDOM end-user
GIN Exe Env	Module for AMBER used by psander during execution

Table 6.6: IIRM core building blocks that are used in the WISDOM case study.

to understand which refinement concepts are used in detail. Table 6.7 provides a summary of those refinements that matter most in the WISDOM case study, while it also takes advantage of refinements described in subsequent case studies in more detail (e.g. HPC extensions).

The yellow numbers indicate different steps explaining the usage of the architecture while at the same time it is described how limitations from the previous section are solved in the corresponding context. Step (0) indicates the desired work situation for WISDOM e-Scientists, i.e. their use of the GridSphere WISDOM portal with a configured identity (i.e. X.509 certificate). In step (1), this certificate is used to contact the VOMS system obtaining a SAML assertion using the SAML interface to the VOMS system published in [302]. The SAML assertion includes role possession, VO or project membership and it is signed with the identity of the VOMS server.

As shown in Figure 6.17, in workflow step (2), a GridSphere portal uses the SAML assertion during an OGSA-BES compliant job submit using JSDL. The use of open standards in this context of the IIRM avoids the need for transformation logic to be maintained in the Vine toolkit that is another contribution of this thesis reducing the maintenance of the overall framework.

The JSDL describes the invocation of applications that are defined by the e-Scientists via the portal GUI. The parameters for the applications are also encoded within the JSDL description. In the case study, the scientific applications FlexX and AutoDock are used in this particular step. As part of gLite, the CREAM-BES [220] service uses the JSDL document to invoke the FlexX and AutoDock applications with the CREAM backend on the EGEE/EGI e-Infrastructure. The `CreateActivity()` operation of the CREAM-BES OGSA-BES implementation takes a JSDL as input that is then further analysed for job execution. This step is indicated as (3) within Figure 6.17, including authorization of WISDOM e-Scientists. Immediately before the job execution, the authorization layer within CREAM-BES checks whether the SAML assertion allows the corresponding e-Scientist to execute applications on the infrastructure. A corresponding XACML policy is in place that could be implemented via the Argus system [303]. As CREAM-BES relies on X.509 proxies parts of the security pattern are used as described in Section 5.1.6.

In workflow step (4), FlexX and/or AutoDock are computed on the EGEE/EGI infrastructure as embarrassingly parallel scientific applications that require no interactions between the processes on different CPUs. The outcome of this computational intensive job is an intermediary result in terms of a compound list that represents potential drugs. In step (5), these results are transferred to a storage using GridFTP while its metadata and the link to the exact storage locations (i.e. GridFTP URIs) are put into a relational database. This database is accessible via a WS-DAIS-compliant specification implementation and is recognized as being part of the AMGA metadata catalogue framework.

On the right side in Figure 6.18, the manual data-staging (blue) complements the already existing concept auf automated data-staging (red) and thus realizes more control over the data transfers during Grid job submissions. With this functionality enhancement the thesis provided a mechanism for manual data-staging and better job control that illustrated the impact of the IIRM in practice.

After storing results, the WISDOM e-Scientist again uses the portal to submit another JSDL-compliant job to the UNICORE OGSA-BES interface implementation installed at some HPC resource within DEISA/PRACE. As shown in workflow step (6) in Figure 6.17, the JSDL describes the execution of a highly scalable AMBER c/fortran script. Not shown in the illustration, is that the WISDOM portal firstly obtains GLUE2 elements from an information service that exposes the AMBER module and its characteristics (cf. Section 5.2.2 (c)) available on this particular site. In step (6), these GLUE2 elements are simply re-used as part of the JSDL thus enabling execution of module AMBER (cf. Section 5.2.2 (b)). In order to indicate that a manual data-staging needs to be performed in the job directory, the e-Scientist set a corresponding hold point as part of our concept of manual data staging (cf. Section 5.2.5 (a)). In terms of security, the (not necessarily the same) SAML assertion must be transferred during the job submit [302] to ensure the authorization of the e-Scientist later within the UNICORE authorization layer.

Workflow step (7) in Figure 6.17 shows that the UNICORE OGSA-BES implementation uses JSDL to describe the AMBER application execution, taking the JSDL parameters (with GLUE2) into account. As part of this process, the Grid job sandbox is formed and its location is exposed to the end-user via GLUE2 according to our concept for enabling manual data staging (cf. Section 5.2.5 (d)) within the OGSA-BES service. The OGSA-BES factory attributes (with GLUE2 refinements) are obtained using the `GetFactoryAttributes` operation. UNICORE OGSA-BES is using its execution backend to enable a job submit to a RMS of a chosen supercomputer. But before the application can be started on DEISA/PRACE, authorization of the e-Scientist must be performed by using the SAML assertion in conjunction with XACML policy checks.

In order to use intermediary results of the EGEE/EGI job outcome within the DEISA infrastructure, the data must be transferred to the DEISA/PRACE storage systems at the corresponding supercomputer site. The significant improvements compared to OGSA-BES is that the progress of execution is not automatically started, because we used the aforementioned holdpoint concept (cf. Section 5.2.5 (a) and (b)). Using AMGA and GridFTP [109], step (8) requires a manual intervention by e-Scientists. Since getting and analyzing result queries from AMGA is a manual process, the concept of more scalable query results is required for its WS-DAIS interface usage (cf. Section 5.2.6). The scientific process requires e-Scientists to analyse the outcome of the EGEE/EGI jobs and take particular data sets as input for the DEISA/PRACE

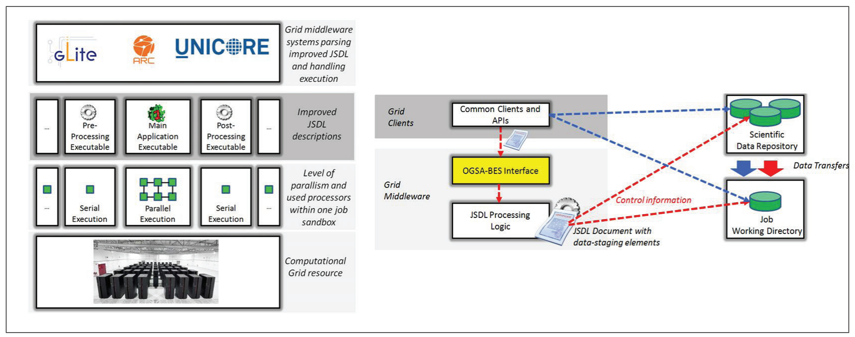


Figure 6.18: The WISDOM case study uses the sequence (left) and manual data staging concepts (right).

Δ	**Functionality Extensions and Improvements**	**Specific usage in WISDOM**
ΔW	5.2.2 (c) Execution Module Characteristics	AMBER module exposure
ΔX	5.2.6 (f) More scalable query results	AMGA responses
ΔV	5.2.5 (b) New Hold states in addition to HPC FSP states	Feedback that run is hold
ΔV	5.2.5 (c) Manual manipulation of job states	Continue psander execution
ΔV	5.2.5 (d) Job sandbox location exposure	Manual input data transfer
ΔU	5.2.1 (g) Application output joins	Merge outputs of AMBER
ΔU	5.2.2 (b) Common Execution Modules	Execution of AMBER module
ΔU	5.2.4 (a) Pre-job sequences (pre-processing)	AMBER data conversions
ΔU	5.2.4 (b) Post-job sequences	AMBER data conversions
ΔU	5.2.5 (a) Pre-defined hold points	Hold until manual transfer

Table 6.7: Functionality improvements used in the WISDOM case study.

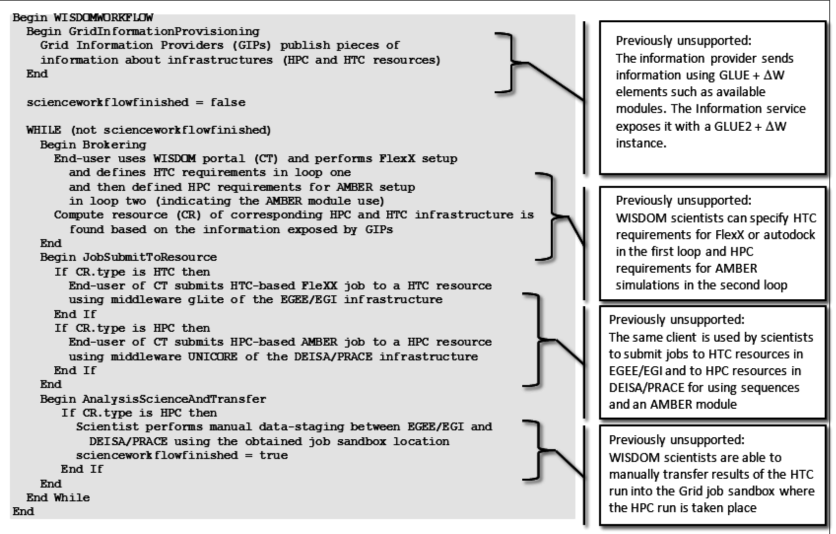
jobs. Assuming that the data of the intermediary results are reachable within DEISA/PRACE after the manual transfer, the concept of manual data-staging (cf. 5.2.5 (c)) allows for the continuation of the execution. With the aforementioned concepts within the reference architecture limitation (a) of the previous section is solved.

After the authorization and the manual data-staging activities by the e-Scientist, the AMBER application is computed on the DEISA/PRACE infrastructure on massively parallel supercomputers as illustrated in step (9). Since AMBER is a huge set of executables that need to be carefully setup, this execution takes advantage of the module concept (cf. Section 5.2.2 (b)) meaning that AMBER executables do not need to be configured anymore before executions. As aforementioned, the MD script was developed by G. Rastelli et al. [162] and is executed by using JSDL descriptions. The script itself uses several different programs of the AMBER molecular dynamics package (e.g. ptray, psander, etc.), some in a serial mode and some in a parallel mode as showns in Figure 6.18. The left parts of the figure shows how the WISDOM case study can take advantage of the sequence concept with the AMBER set of executables. The WISDOM e-Scientists raised the demand of sequences (cf. Section 5.2.4 (a) and (b)) that are part of the reference architecture, and thus overcome limitation (c) identified in the previous section. To provide an example, one small program of AMBER is just used to transfer the input data structure in a format that is understandable by psander and thus is executed before the real parallel psander production run. In a similiar manner, after the execution of psander, another conversation of data structures takes place with another small executable of AMBER that makes the results suitable to be stored as final results in a database. More insights into sequences with AMBER are given in [193]. With the functionality enhancement of sequences, the thesis provided a mechanism that illustrates the impact of the IIRM in practice.

All the previously described activities happen as part of step (9) in Figure 6.17, where the resource management system schedules different end-users on one Grid resource. During the complex AMBER set of executions and for a more convenient analysis of the e-Scientists, application output joins (cf. Section 5.2.1 (g)) are needed.

The overall computation is now significantly faster than without the interoperability between EGEE/EGI and DEISA/PRACE, because the AMBER code is scalable and thus capable of leveraging the massive number of CPUs available on resources within DEISA/PRACE. The outcome of this job is the final result for the laboratory experiments accessible via AMGA and GridFTP indicated in step (10).

The overall aforementioned workflow steps describe a scientific solution that has been computed within EGEE/EGI and DEISA/PRACE that would not be seamlessly possible without the interoperable components of the IIRM. The work in this case study was mainly carried out in collaboration with members of the WISDOM initiative and as part of the work of the author



(i) Listing: Pseudo-code of basic design pattern algorithm applied to the WISDOM case study

Figure 6.19: The WISDOM scientific workflow can be mapped to the IIRM run-time pattern algorithm.

in the OMII-Europe project following several subsequent activities. Some details about the work in WISDOM is illustrated in Figure 6.18 to increase the understanding of key concepts used in this particular case study. More insights into the scientific workflow are described in [259] that lead to the more formal definition of the multi-Grid algorithm as illustrated in Figure 6.19. It also highlights the previously unsupported elements within this workflow that are realizable by using the concepts of the IIRM. The pseudo-code notation maps the WISDOM workflow to the general design pattern defined in Chapter 5 alongside the architecture work. The reference architecture is thus applicable to the WISDOM workflow and its different steps that use different key IIRM architecture aspects and concepts. Evidence that supports this claim is the possibility to map the workflow to the general design pattern. This in turn highlights another key impact of this thesis by enabling the use of WISDOM applications with EGEE/EGI and DEISA/PRACE. As a consequence, the proposed framework with IIRM elements contributes to cheaper and faster drug discovery as scientific innovation as another key impact of this thesis.

Another collaboration with e-Scientists lead to activities appling for a DECI project [301] in order to obtain computational time on the HPC-driven Grid infrastructure DEISA/PRACE as WISDOM is already organized as one VO in EGEE/EGI. This higlights an important aspect, because even if the technical foundations are in place the policies for obtaining computational time in DEISA/PRACE are still valid and thus access needs to be requestes for CPU allocations and need to be granted by the corresponding peer-review committee. The technical interoperability achieved with the IIRM in this context is important, but also the work around resource access policies to streamline interoperability can be improved by the corresponding infrastructures (e.g. policy groups as described by segment 5 in the aligned process).

6.4 Architecture Implementation for the VPH Applications

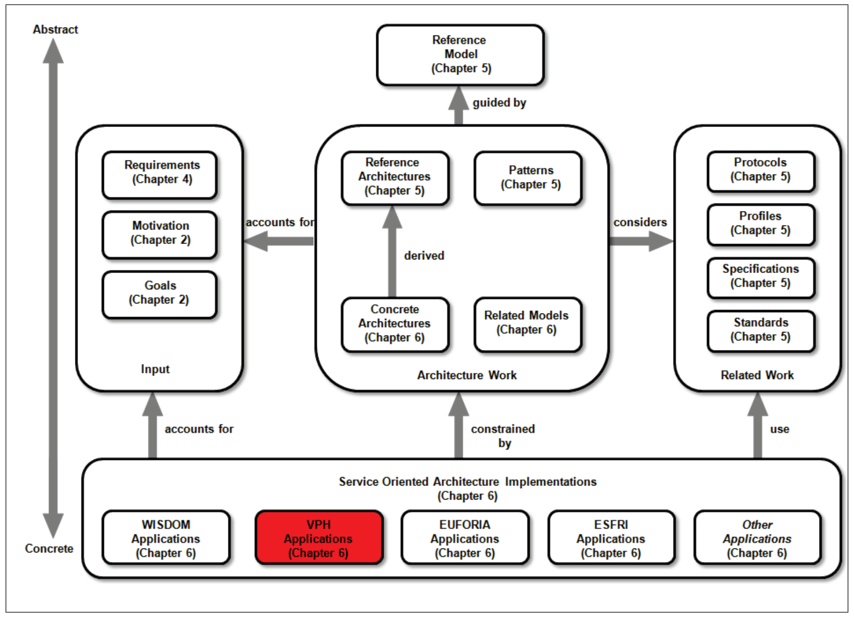


Figure 6.20: SOA implementation for VPH applications constrained by the concrete reference architecture.

This section describes the concrete SOA-based implementation of the scientific case study named *Virtual Physiological Human (VPH)* [92] that was conducted during the definition of the proposed reference model. It contributed enormously to the findings of this thesis and is published in [263] and [272]. Figure 6.20 provides an overview how the case study fits in the overall reference model approach and its implementations.

The path of the scientific investigation throughout the case study is represented in the structure of this section. A short introduction to the VPH roadmap and its basic framework is given in order to understand how VPH e-Scientits use production e-Science infrastructures. Then scientific applications used in VPH are introduced that require more than one e-Science infrastructure.

The architectural analysis and the scientific application analysis set the foundation for a critical academic analysis to explain the limitations of the basic VPH framework. As a key contribution of this section, insights are given into how those identified limitations can be solved by using the IIRM and its associated reference architecture implementations. These implementations are already emerging in production e-Science infrastructures as previous sections revealed.

This section aims to verify that the reference model is applicable to a wide variety of VPH applications that seek to take advantage of more than one production e-Science infrastructure. Many prototype developments and studies contributed to the case study based on a fruitful collaboration between the VPH community, OMII-Europe [69], and later also the DEISA2 project

[184]. Results have been presented at the Supercomputing 2008 in Austin [95] at the JSC booth as part of the OGF GIN demonstrations.

6.4.1 The STEP Roadmap and the Basic VPH Framework

Today, e-Health can be considered as one of the most important research fields. This includes the use of information and communication tools as well as using computational methods to support these tools behind the scene or to support the understanding of health fundamentals. It thus plays a significant role in improving the health of world-wide citizens.

While e-Health is a large scientific research field, the focus in this section is on recent work towards the VPH that is part of a greater roadmap organized from the *Strategy for the Euro-Physiome (STEP)* consortium [92]. VPH needs computational Grid resources in order to realize the three major working cycles illustrated in Figure 6.21. As part of the academic studies, the interest in this thesis is in having scientific applications to gather insights using models and data and to validate them in a cyclic fashion as illustrated in Figure 6.21.

Since several years, VPH e-Scientists take already advantage of single e-Science infrastructures to perform computationally-intensive investigations of the human body. But a known limitation of these approaches are that they tend to consider each of the constituent parts separately without taking into account the multiple important interactions between those parts. Subdivisions make it impossible to investigate the systematic nature in which the body functions, however, many e-Science applications in this area are limited by the computational power provided in the respective e-Science infrastructures while sharing it with other applications of science and engineering. The VPH vision is a methodological and technological framework that enables collaborative investigations of the human body as a unique complex system. In order to achieve this, the basic VPH framework was created to enable the interoper-

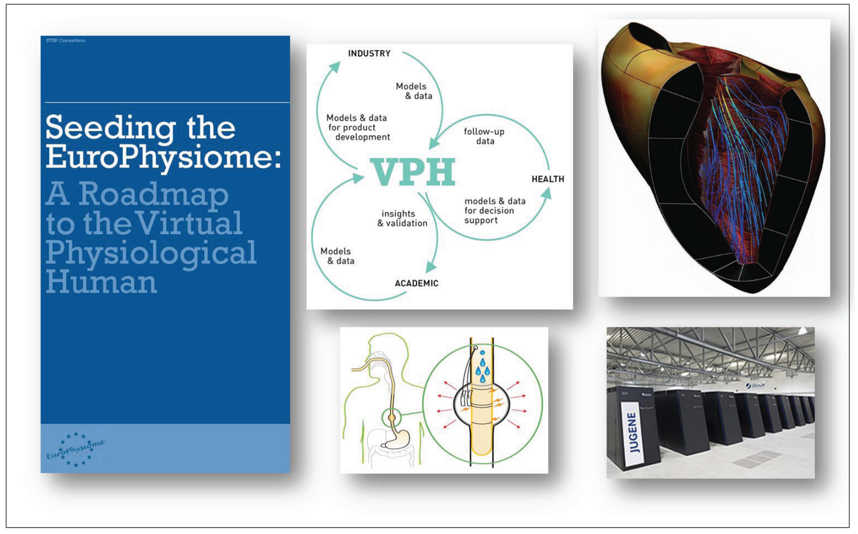


Figure 6.21: The VPH vision includes three major cycles in the health, industry, and academic domain [92].

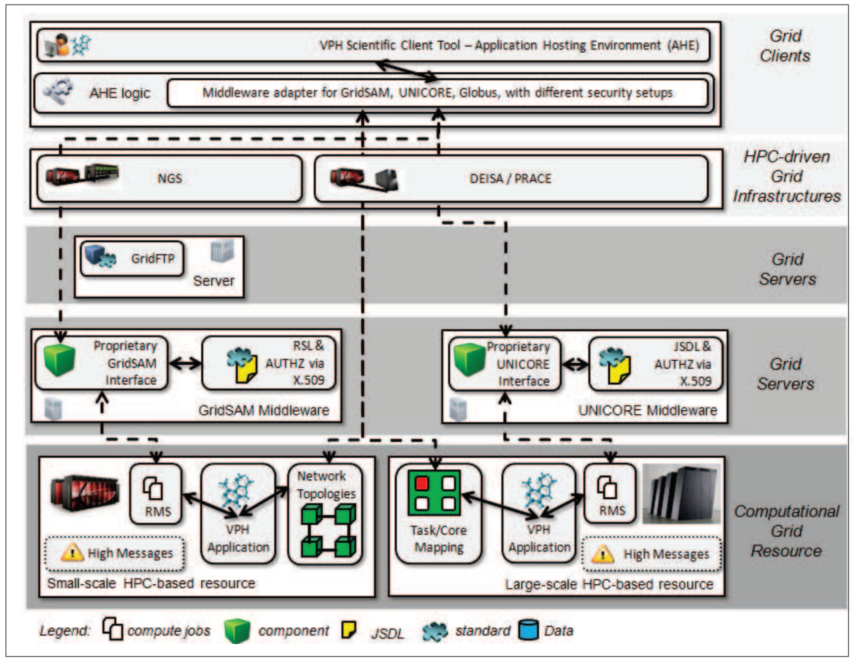


Figure 6.22: The basic VPH framework uses technologies of NGS and DEISA/PRACE.

ation between the National Grid Service (NGS) [59] in the UK and the European HPC-driven infrastructure DEISA/PRACE as illustrated in Figure 6.22.

Figure 6.22 indicates a key requirement of the VPH community that is related to the seamless access of different infrastructures. The e-Scientists often use their own specific clients that in this case is named as the Application Hosting Environment (AHE) [308]. The fundamental goal of this tool is to allow clinicians to seamlessly interact with a large amount of computational power available in different e-Science infrastructures even from within their operation theatre.

This specific use case implies the interactive access to resources of these infastructures as well in order to perform computational steering in real-time. Computational steering refers to the change of application parameters on the fly during the application execution on one of the resources within an e-Science infrastructure. The goal of these real time visualization and computational steering is to allow clinicians to interact with the simulations as they run in order to review the possible effects of various surgical investigations. This leaves the scope of this thesis, but approaches are given in [139]. In [255] it is shown that foundational concepts (e.g. job submission, etc.) are used to establish an interactive steering channel that can work seamlessly together with the IIRM core building blocks being part of this thesis as they are very similiar (e.g. job submission interfaces).

Figure 6.22 illustrates the pre-production setup of the case study enabling the AHE to submit jobs, in this particular example, to the NGS and DEISA/PRACE.

6.4.2 Scientific Applications of the e-Health Domain

The previous section focussed on the basic VPH framework from a technical perspective. This section reveals insights about one particular e-Health VPH application from an end-user perspective. The overall idea is illustrated in Figure 6.23 (using elements of [263] and [272]) that basically illustrates the use of three e-Science infrastructures.

The VPH initiative is part of the larger international Physiome Project that clearly raise the demand for aligning world-wide interoperable e-Science infrastructures for collaborative research within its roadmap [92]. This is needed in order to tackle the computational-intensive challenges that the simulation of the VPH implies. The VPH community seeks to serve the development and integration of multi-scale models, which have different computational requirements. The range is from single processor desktop machines to the largest supercomputers available in different kinds of e-Science infrastructures as shown in Figure 6.23. The STEP in general and the VPH community in particular provides a lot of scientific applications that can take advantage of interoperable e-Science infrastructures. Here, one application is picked for the case study that is very similiar to all the others. They basically all share in common that the majority of codes are parallel computing techniques using MPI while the VPH framework makes also partly use of non-parallel codes on HTC-driven infrastructures such as EGEE/EGI.

The particular pre-production setup from the case study is using one specific application in the research field of cardio-vascular diseases that are the cause of a large number of deaths in the developed world. The problems of patients are often due to anomalous blood flow behavior in the neighborhood of bifurcations and aneurysms within the brain. Cerebral blood flow behaviour plays a crucial role in the understanding, diagnosis, and treatment of this disease. The central goal of this application is to simulate the blood flow behaviour using the computer

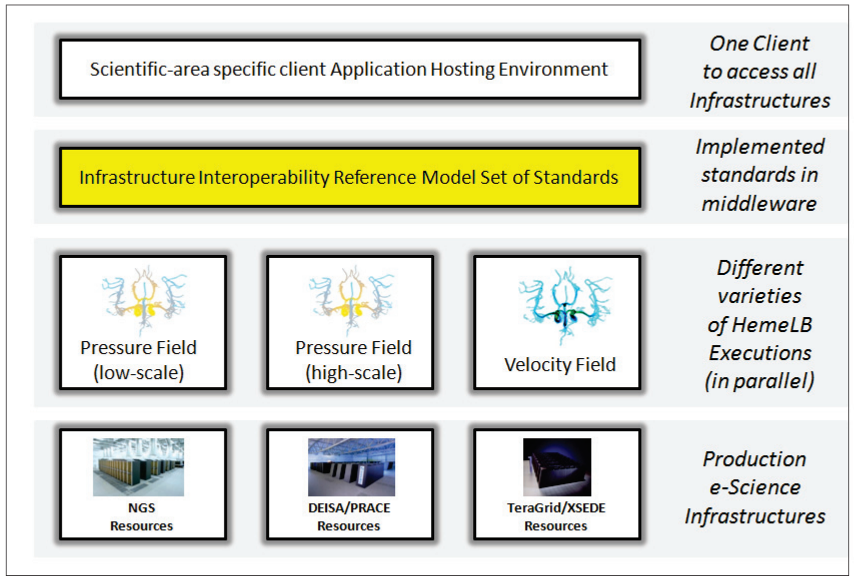


Figure 6.23: HemeLB brain bloodflow simulations with core building blocks of the IIRM reference architecture.

No	e-Science Infrastructure	Setup	e-Science Application Demands
1	NGS	Section 6.2.2	moderate parallelized HemeLB blood flow pressure field code (low-scale) requires small to medium-sized computational Grid resources;
2	DEISA/PRACE	Section 6.2.2	massively parallelized HemeLB blood flow pressure field code (high-scale) requires large-scale HPC resources with optimized BlueGene code;
3	TeraGrid/XSEDE	Section 6.2.3	massively parallized HemeLB blood flow velocity field code requires modern large-scale computational Grid resources;

Table 6.8: Summary of the e-Science infrastructure setups of the VPH case study.

power available on multiple production e-Science infrastructures today. The application thus raises a demand for a large amount of computing resources offering different scales, because simulating a whole blood flow in a brain is computational-intensive.

The Grid Enabled Neurosurgical Imaging Using Simulation (GENIUS) project [36] is working in this particular field and is mainly concerned with performing neurovascular blood flow simulations in support of clinical neurosurgery. It uses a lattice-boltzmann scientific code named HemeLB [219] designed to simulate fluid flow in the sparse topologies of the patient brains. According to [219], the simulation models are derived from patient-specific brain x-ray angiography scans that are in turn used as input to the simulation. The infrastructure setup thus requires the possibility for large and effective file transfers that are able to transport the x-ray-based data to the computing resources across boundaries of existing e-Science infrastructures. This circumvents a duplicate storage of these large datasets in each different e-Science infrastructure that is used.

There is a high potential to use the VPH workflow with the IIRM and its reference architecture, but e-Scientists also require the seamless access with their already existing tool named AHE introduced in the previous section. The difference to the first case study is that the IIRM is used with a community-specific integrated framework that already uses a scientific-area specific client instead of a thin Web-based GridSphere portal.

Many scientific e-Health applications are very similiar to the HemeLB code that is used as part of the studies. All these applications have thus a very similiar setup and are not even restricted to the VPH scientific field, but can be actually used throughout the whole e-Health domain and beyond using the same infrastructure setup.

Table 6.8 summarizes the production e-Science infrastructure setup required by the specific case study. It clarifies the question which e-Science infrastructure setup is used with what concrete types of e-Science applications while references to previous sections are given for infrastructure setup information. In this case study, the VPH e-Scientists use the HemeLB application with different computational Grid infrastructure resources (cf. Definition 6).

6.4.3 Academic Analysis and Production Infrastructure Setup Experience

This sections analyses the basic architectural VPH framework using the scientific application HemeLB. The basic framework has been already used in production e-Science infrastructure use cases, but the outcome of this analysis points to limitations of the basic framework that are summarized in Table 6.9.

No.	Limitation short description
(a)	additional Layer Approach (transformation logic)
(b)	no large-scale and recent HPC resource architecture support
(c)	no common information model
(d)	no support for sequential compiling applications before the Grid job

Table 6.9: Overview of limitations after academic analysis of the VPH framework.

The high-level analysis of the general approach of the framework points to drawbacks of the overall approach. According to Chapter 3, the first major limitation identified in the basic VPH framework is the use of the *'(a) additional layer approach'* to enable interoperability between e-Science infrastructures in general and their corresponding Grid middleware GridSAM and UNICORE in particular. In the VPH case study, the additional layer is represented by the AHE tool that provides specific adapters for all Grid middleware systems such as GridSAM, UNICORE, Globus, and others.

Apart from this general approach, a more fine grained academic analysis of the lessons learned provides more clarity and points to specific challenges that need to be addressed to improve the efficiency of the VPH framework using the IIRM reference architecture. There is a limitation that *'(b) no large-scale and recent HPC resource architecture support'* is present in the basic framework. Also when using OGSA-BES and JSDL (i.e. core building blocks) in context point to limitations. The HemeLB application is an MPI code that is able to run more efficiently when several missing HPC resource aspects that modern HPC architectures provide are specified in the job description. Some of these advanced features have been the support for network topologies, shape support, and task/core mapping support. In addition, there is a demand for more easier access to the high messages of a particular HPC resource.

Closely related is the limitation of having *'(c) no common information model'* in order to express the resource information in a coherent way so that it can be easily parsed by the AHE and transformed in subsequent resource requests during job submission. Each HPC system offers distinct functionality (e.g. network topologies) and capabilities (e.g. shapes) and each of those are not part of one common information model. The information must be transformed (i.e. transformation logic is used) in a way that the AHE is able to understand it. In several cases resoure information gathering is even performed manually obtained by e-Scientists looking on partly outdated Websites leading to error-prone job submissions. In context of running the above described HemeLB application on different HPC machines in different versions, there was *'(d) no support for sequential compiling applications before the Grid job'*.

While the aforementioned limitations have been the most important ones identified in this case study, there have been also other limitations (e.g. manual data-stagings) that overlap with those that have been already explained in other use case studies (e.g. WISDOM). The focus in this case study is on those listed in Table 6.9 complementary to those of other use cases.

6.4.4 Reference Model Impact and Applicability

The fundamental goal of the concrete reference architecture instance described in this section is to improve the VPH e-Science methods thus significantly contributing to the VPH vision of seamlessly using NGS and DEISA/PRACE as shown in Figure 6.24. As the usage of DEISA/PRACE is similiar to TeraGrid/XSEDE in this particular usage, the description of the use case focus only on NGS and DEISA/PRACE workflow steps. The usage of TeraGrid/XSEDE resources can be seen in analogy to the DEISA/PRACE usage and thus Figure 6.24 shows only two infrastructures. This figure represents a specific *'architecture instance'* of

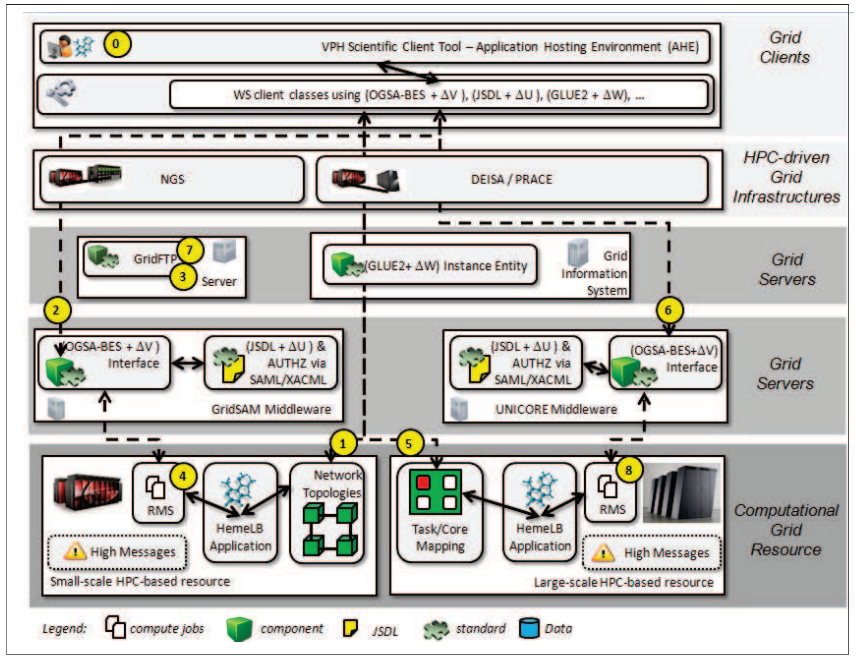


Figure 6.24: Enhanced VPH framework using the IIRM with e-Health applications.

the IIRM and provides insights in the used core building blocks. Several concept refinements (i.e. Δ) to the core buildung block functionality have been necessary to do production runs on DEISA/PRACE and partly also on TeraGrid/XSEDE. The production runs on DEISA/PRACE have been supported by computational time of the *DEISA virtual community VPH* [301].

The case study with using emerging IIRM core building blocks created lessons learned that contributed significantly to the findings documented in Chapter 5. An overview of which concrete core building blocks are provided as part of the case study is presented in Table 6.10. The concrete refinement concepts (e.g. indicated with Δ) are used in order to overcome the

Core building blocks	**Specific usage in VPH**
OGSA-BES + ΔV	WS submission to create a Grid job using a middleware systems
JSDL + ΔU	HemeLB application job description
GLUE2 + ΔW	Computational Grid resource information details
SAML	Encoding (i.e. SAML assertions) of VPH end-user security attributes
WS-Security	Mechanism to transport SAML assertions in OGSA-BES WS messages
XACML	Security policy for end-users (also from VPH)
GridFTP	Transfer from patient-specific data to be used by HemeLB
X.509	Security X.509 certificate provided by each VPH end-user
GIN Exe Env	Common environment variables used by HemeLB during execution

Table 6.10: IIRM core building blocks that are used in the VPH case study.

limitations listed in the previous section. While Table 6.10 provides an overview of the used core building blocks, the next paragraphs provide a step-wise walkthrough of Figure 6.24 in order to understand which refinement concepts are used in detail. This table and Figure 6.24 illustrates another key contribution of this thesis by using IIRM core building blocks and their refinements with another real scientific use case.

Table 6.11 summarizes those refinements that matter most in this particular case study, while also this case study partly takes advantage of refinements already described in our previous case study (e.g. manual data-staging and sequences).

The yellow numbers in Figure 6.24 indicate steps explaining the usage of the concrete architecture while at the same time the identified limitations are addressed. Step (0) indicates the usual work situation for VPH e-Scientist meaning that they are using their AHE tool with their

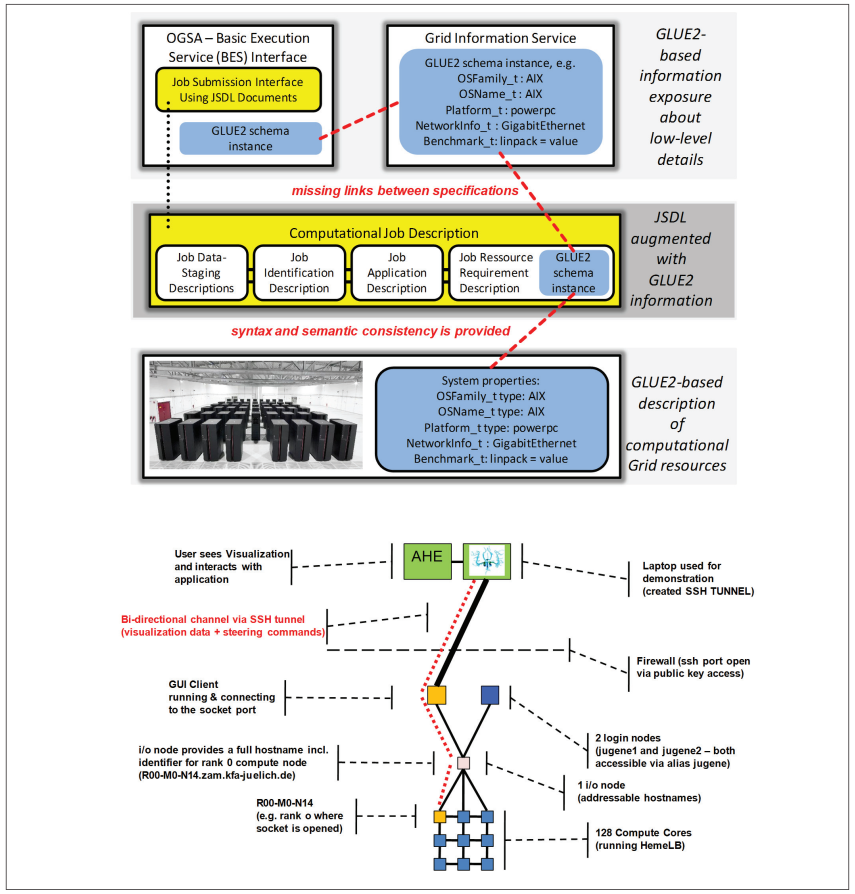


Figure 6.25: The VPH case study uses improvements of Chapter 5 (top) and difficult HPC setups (bottom).

Δ	Functionality Extensions and Improvements	Specific usage in VPH
ΔU	5.2.3 (a) Network topology (torus, global tree, etc.)	Describe BlueGene-specific capabilities
ΔU	5.2.3 (b) Shape reservation (x X y X z)	Describe BlueGene-specific capabilities
ΔU	5.2.3 (f) Task/Core Mapping Definition	Describe BlueGene-specific capabilities
ΔU	5.2.4 (a) Pre-job sequences (compilation)	Compile machine-specific HemeLB
ΔW	5.2.3 (c) Network information enhancements	BlueGene capabilities exposure
ΔW	5.2.3 (d) Available shape characteristics	BlueGene capabilities exposure
ΔW	5.2.3 (e) High message support	Computational resource status
ΔW	5.2.3 (g) Available task/core mappings	BlueGene capabilities exposure

Table 6.11: Functionality improvements used in the VPH case study.

configured identity (i.e. X.509 certificate). In step (1), the end-user is able to obtain accurate HPC resource information from an information service using the common information model GLUE2, including some refinements in order to address low-level HPC resource capabilities as described in Chapter 5. Obtaining information in a common format overcomes the identified limitation (b) and enables a common understanding of HPC functionalities (e.g. available CPU/core/threads setups, network topologies, etc.) across different Grid infrastructures. This step in particular is using the HPC refinement concepts (cf. Section 5.2.3 (c), (d), (e), and (g)). The use of this enhanced functionality to increase the HemeLB efficiency on recent HPC resources illustrates the impact of the IIRM in practice.

In step (2), the AHE is then able to use these pieces of information as part of an OGSA-BES call to GridSAM that includes the refined JSDL (cf. Section 5.2.3 (a), (b), and (f)) meaning that initially obtained GLUE2 elements are simply re-used for the JSDL job submission. Using open standards with IIRM core building blocks avoids the need for transformation logic to be maintained in VPH AHE that is another key contribution of this thesis. This solves the identified limitation (c) since the refined JSDL is able to express support for recent HPC systems such as the used HECTOR system within NGS. The application described in the JSDL is the HemeLB application variant for lower scale systems in this specific context. Using the JSDL data-staging concept, GridFTP is used to transfer the patient-specific data into the job directory on the Hector system in step (3).

After that, step (4) indicates the low-level machine level where the sequence concept is used (cf. Section 5.2.4 (a)) in order to overcome our identified limitation (d). The HemeLB application is compiled for the corresponding machine type while the sourcecode can be provided as another staged-in data or accessible via a global file system. Immediately after the compilation within the 'same' Grid job sandbox, the run of the HemeLB application is started simulating a low-scale pressure flow of a human brain based on previously transferred data. The provisioning of this mechanism to ease the machine-specific compiling of applications illustrates the impact of this thesis in practice.

After the job submission to the NGS, the e-Scientist uses the AHE in order to obtain also standardized information about HPC resource information available in DEISA/PRACE via GLUE2 and the refinements (cf. Section 5.2.3 (c), (d), (e), and (g)) in step (5). Pieces of information in a common format are present that expresses details about modern large-scale HPC resources such as the BlueGene system in Juelich as part of DEISA/PRACE. This pieces of information can consist of recent HPC features such as task/core mapping definitions for the particular machine or available shape setups. Such very accurate pieces of information are necessary to be part of the refined JSDL (cf. Section 5.2.3 (a), (b), and (f)) that is submitted to the OGSA-BES implementation in step (6) decribing another HemeLB execution specifically optimized to perform large-scale simulations. Within DEISA/PRACE, GridFTP is used to transfer patient-specific data orginally obtained from an MRT into the working directory of the job indicated as step (7). Step (8) indicates the execution of HemeLB on the Juelich machine within DEISA/PRACE being compiled specifically for the BlueGene architecture beforehand (cf. Section 5.2.4 (a)). It efficiently simulates the pressure field of a human brain bloodflow in high-scale. TeraGrid/XSEDE executions are kept out for clarity and to reduce duplication, since it follows essentially the same steps while HemeLB might be used with such resources to simulate, for instance, the bloodflow velocity field.

All in all, the steps described a scientific solution that has been computed within NGI, DEISA/PRACE, (and partly TeraGrid/XSEDE) that would not be seamlessly possible without interoperable components of the IIRM reference architecture. The work in this case study was mainly carried out in collaboration with the VPH community and as part of the work in the OMII-Europe and DEISA2 project. Some details about the work in VPH is illustrated in Figure 6.25 to increase the overall understanding of the concepts used in the case study. As shown in Figure 6.25, the VPH case study started using resources like the Supercomputer JUMP (left) the study then continued on BlueGene supercomputers JUBL and JUGENE (right, including steering setup). As part of this process of bringing the HemeLB application from one architecture to these others, several insights have been gathered for the functionality improvements in Chapter 5. More insights into both computational steps are described in [263] that lead to the more formal definition of the multi-Grid algorithm as illustrated in Figure 6.26. It also highlights the previously unsupported elements within this workflow but that are possible by using the concepts of the IIRM. The pseudo-code provides evidence that it is possible to map the VPH workflow to the general design pattern defined in Chapter 5 algongside the architecture work. The reference architecture is thus applicable to the VPH workflow and its different parallel steps that in turn use different application scales on different HPC resources. Evidence that

```
Begin VPHWORKFLOW
  Begin GridInformationProvisioning
    Grid Information Providers (GIPs) publish pieces of
    information about infrastructures (small-/large-scale HPC resources)
  End

  scienceworkflowfinished = false

  WHILE (not scienceworkflowfinished)
    Begin Brokering
      End-user uses AHE Client (CT) and performs HemeLB setup
        and defines small-scale HPC requirements in loop one
        and then defined large-scale HPC requirements for HemeLB setup
        in loop two
      Compute resource (CR) of corresponding HPC infrastructures are
        found based on the information exposed by GIPs
    End
    Begin JobSubmitToResource
      If CR.type is small-scale HPC then
        End-user of CT submits small-scale HPC-based HemeLB job
        to a HPC resource using GridSAM of the NGS infrastructure
      End If
      If CR.type is large-scale HPC then
        End-user of CT submits large-scale HPC-based HemeLB job
        to a HPC resource using UNICORE of the DEISA/PRACE infrastructure
      End If
    End
    Begin AnalysisScience
       If CR.type is small-scale or large-scale HPC then
         Scientists perform manual data-staging with GridFTP to repository
         scienceworkflowfinished = true
       End
    End
  End While
End
```

Previously unsupported:
The information provider sends information using GLUE2 elements such as network topologies, shape setups or task/core mappings. The Information service exposes it with a GLUE2 Instance

Previously unsupported:
VPH scientists can specify small-scale and large-scale HemeLB simulation runs that can be even submitted in parallel or sequentially as needed

Previously unsupported:
Using the same client the scientists can submit jobs to small-scale resources in the UK NGS while also they are able to submit to large-scale resources of DEISA/PRACE

Previously unsupported:
VPH scientists are actually able to manually transfer analyzed results manually to their patient-specific organized repositories

(i) Listing: Pseudo-code of basic design pattern algorithm applied to the VPH case study

Figure 6.26: The VPH scientific workflow can be mapped to the IIRM run-time pattern algorithm.

supports this claim is the possibility to map the workflow to the general design pattern. This highlights another key impact of this thesis by enabling the use of VPH applications with NGS, DEISA/PRACE, and TeraGrid/XSEDE.

The presented use case was a collaboration between e-Scientists in the context of the DEISA VPH virtual community [301] that was specifically getting computational time on the HPC-driven Grid infrastructure DEISA/PRACE. The access to the NGS system was achieved without the direct involvement of the author. During the preparation period it turned out that the policies to apply for large scientific endeavours is still cumbersome for the e-Scientists that can not rely on using these systems in general since their proposal may be rejected. Technical interoperability is achieved with the IIRM in this context. But the usage policies of getting computational time on the rarely available HPC resources is pointing to the importance of solving the policy issues raised as part of the segmented process in Chapter 5.

6.5 Architecture Implementation for the EUFORIA Applications

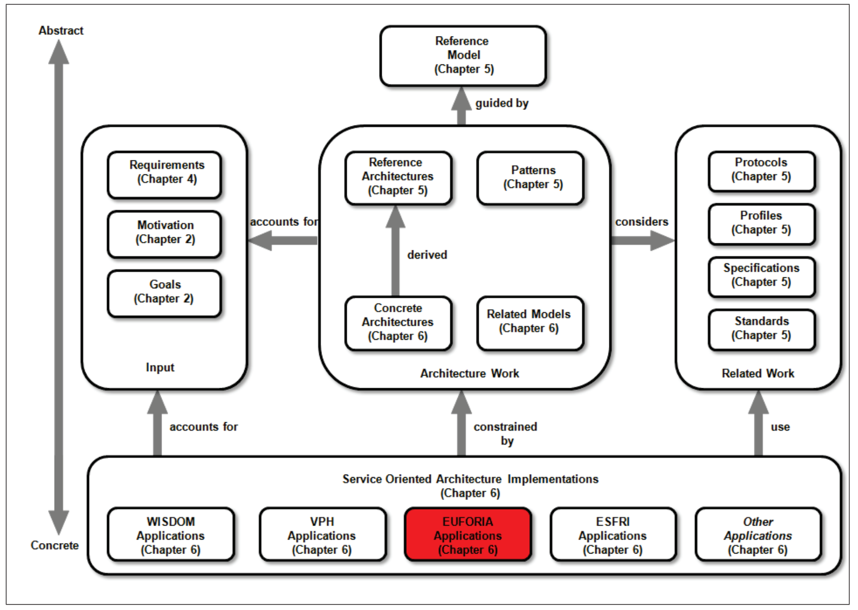


Figure 6.27: SOA implementation for EUFORIA applications constrained by the concrete reference architecture.

This section describes the concrete SOA-based implementation of the *EU Fusion for ITER Applications (EUFORIA)* [247] case study that was conducted during the definition of the reference model. This study contributed enormously to the findings of this thesis and is published in [225] and [272]. Figure 6.27 provides an overview of how the case study fits with the overall reference model approach and its implementations.

The scientific investigation throughout the case study is also represented in the structure of this section. A short introduction to the EUFORIA project [247] and its basic framework is given in order to explain how e-Scientists from the fusion domain want to use production e-Science infrastructures. Then scientific EUFORIA application workflows are explained that require more than one production e-Science infrastructure.

The architectural analysis and the scientific application analysis in turn are the foundation for the critical academic analysis explaining the limitations of the basic EUFORIA framework. As a key contribution of this third case study, insights into how such identified limitations are solved are given by using the IIRM and the associated reference architecture implementations. These implementations are already used be fusion e-Scientists and thus emerging in production e-Science infrastructures today.

This section aims to verify that the reference model is applicable to the EUFORIA scientific workflows that seek to take advantage of more than one production e-Science infrastructure. Many prototype developments and studies contributed to the case study which was based on a fruitful collaboration between the EUFORIA project [247], the fusion community, and the

DEISA2 project [184]. The results have been presented at the Supercomputing 2009 in Portland [96] at the JSC booth as part of the OGF GIN demonstrations.

6.5.1 The EUFORIA Framework

The fundamental goal of the EU Fusion for ITER Applications (EUFORIA) project [247] is to exploit production e-Science infrastructures for fusion science as illustrated in Figure 6.28. A wide variety of scientific applications are used to simulate elements of the International Thermonuclear Experimental Research (ITER) [121] tokamak, which is a fusion device that may become the basis for future fusion power-generating power-plants. It aims to demonstrate the scientific and technical feasibility of fusion as a sustainable energy source for the future.

To exploit the full potential of the device shown in Figure 6.28 (using elements of [40] and [247]), and to guarantee its optimal operation, a high degree of physics modelling and simulation is required, even in the current construction-related phase of the ITER project. Detailed modelling tools that are required for an adequate description of the underlying physics, are in general very demanding from a computational point of view.

The EEF report 2010 summarizes the requirements of the ITER community as *'Compute resources requirements range from small Linux clusters to supercomputers'* [161]. As a consequence, this case study requires resources of the HPC-driven infrastructure DEISA/PRACE but also resources of the HTC-oriented EGEE/EGI infrastructure. Many fusion applications form scientific workflows across heterogeneous Grid resource types. Again, the EEF report summarizes the requirements of this community as follows: *'Depending on the requirements, the work-flow is executed on local resources, on a compute grid or on remote HPC. Standardized ways are needed for code coupling and execution, for monitoring and for data management* [161]. While ITER has many activities, the focus in this case study is on leveraging the potential of DEISA/PRACE and EGEE/EGI with the EUFORIA framework as illustrated in Figure 6.29.

As part of the client layer, the client tool is a GUI responsible for end-user interaction and managing multi-site, distributed, and heterogeneous workflows. For managing workflows, the architecture uses the Kepler [215] tool, which is a generic scientific workflow engine. This tool is capable of supporting general job executions using so-called *'actors'* [225], and therefore the EUFORIA project defined HTC- as well as HPC-specific additional actors enabling job management and data-staging functions. By combining a meaningful sequence of such actors, client

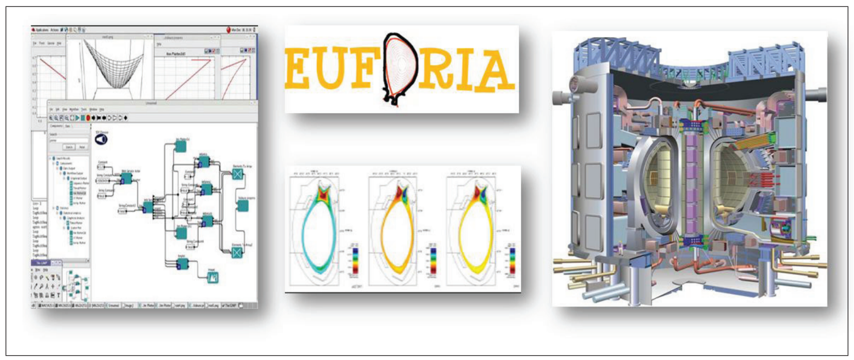


Figure 6.28: The EUFORIA project enables fusion research on production e-Science infrastructures [247].

applications can create a scientific workflow in a very flexible and convenient manner. Once the workflow is verified, the request of each actor is forwarded to the so-called Resource Access Server (RAS) [225].

The RAS in turn provides general functions of job management and monitoring as well as certain data management functions. As shown in Figure 6.29, the RAS interacts with HTC as well as HPC resources through Grid middleware-based adapters implementing transformation logic (cf. Chapter 3). For HTC-based job requests RAS communicates with gLite services, whereas for HPC services it interacts with UNICORE services. The gLite middleware is used in the EGEE/EGI infrastructure, whereas UNICORE provides access to resources of the DEISA/PRACE infrastructure. Using these types of services and thus HPC or HTC resources, fusion e-Scientists can conveniently execute serial and massively parallel jobs. The e-Scientists decide an application launch only within Kepler, but they need to know low-level details of execution in the basic framework in order to understand which fusion codes can run on which resources. The knowledge of which libraries are installed is necessary to efficiently use the EUFORIA framework. In the production setup, UNICORE is used as an interface to HPC, because of its wide range of support for modern resource management systems. As shown in Figure 6.29, RAS interacts with UNICORE using the Vine toolkit [278].

Figure 6.29 reveals that a fusion e-Scientist uses a scientific client that is implemented with the NX technology [62], which handles remote X window connections to the NX Server installed on the fusion community Kepler server and thus in turn Kepler can be locally displayed

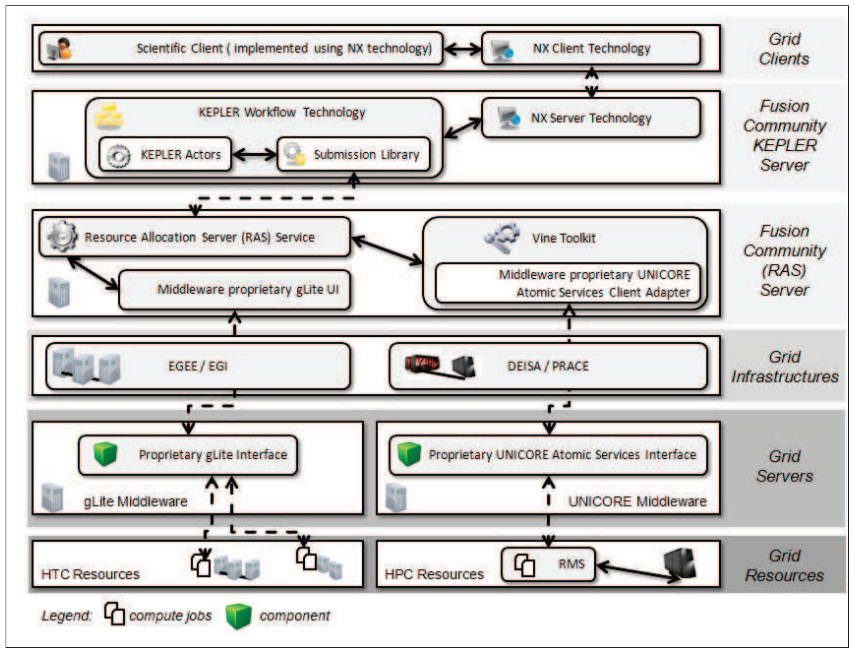


Figure 6.29: The basic EUFORIA framework uses technologies of EGEE/EGI and DEISA/PRACE.

on the scientist's desktop. Based on these connections and as illustrated in Figure 6.29, the Kepler workflow technology is then used to set up fusion application workflows that use several middleware-specific Kepler actors (e.g. gLite Actor, UNICORE Actor, etc.) and related submission libraries to access the RAS server of the fusion community. Figure 6.29, illustrated two adapters that map the functionality of the middleware-specific Kepler actors within the fusion-specific scientific workflow to the corresponding middleware-based requests of the corresponding middleware systems. A proprietary gLite UI adapter is used to access HTC-oriented resources within EGEE/EGI, but also the Vine toolkit supports gLite. It is used as another adapter with the proprietary UNICORE Atomic Services (UAS) [293] and thus provides access to HPC-driven Grid resources within DEISA/PRACE.

6.5.2 Scientific Applications of the Fusion Domain

Whereas the previous section focused on the basic EUFORIA framework from a technical perspective, this section reveals insights about some of the scientific fusion codes from an end-user perspective. The overall scientific workflow is illustrated in Figure 6.30 (using elements from [91] and [40]) which shows two fundamental computational steps.

The EUFORIA project enhances modelling capabilities for ITER through the joint use of HTC and HPC resources together with the fusion modelling community by adaptation, optimization and integration of a set of critical applications for edge and core transport modelling

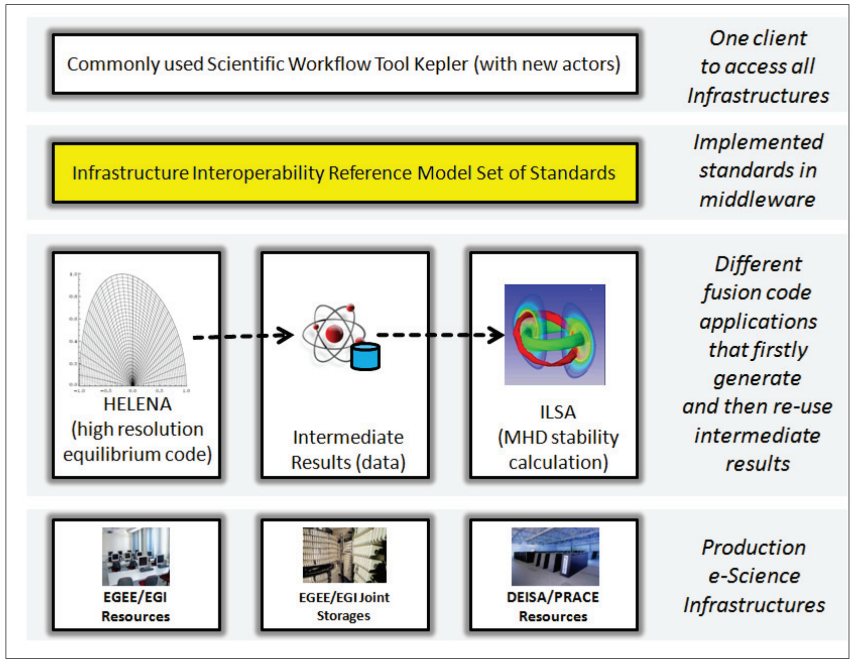


Figure 6.30: HELENA and ILSA fusion simulations with core building blocks of the IIRM reference architecture.

as well as turbulence simulations. Most of these application codes allow construction of complex, combined workflows that produce advanced physical results. The number of codes of such combined workflows depends on their characteristics and a wide variety of them have been enabled and optimized to be used partly with HPC as well as HTC resources.

The particular fusion use case application of the case study is HELENA-ILSA, which is one scientific fusion workflow, illustrated in Figure 6.30 that includes parameter scan applications. HELENA [198] is a high-resolution fixed boundary equilibrium code used to calculate the magnetic flux surfaces in a tokamak by solving the Grad-Shafranov equation. This involves solving a large sparse band matrix equation iteratively.

ILSA [199] is a linear magnetohydrodynamics (MHD) code. It is used to compute linearly unstable MHD modes (on a mesoscale) in tokamak plasma. Typical modes that can be calculated with ILSA are ballooning modes, peeling modes, kink modes, sawteeth, and neoclassical tearing modes. ILSA also solves linearized set of coupled MHD PDEs using various solvers (QR algorithm, inverse vector iteration, Lanczos algorithm, Nyquist method). The major use of the framework is the inverse vector iteration and the calculation of the stability of a ballooning mode in ideal MHD. Both of the aforementioned codes are used with the framework to provide the end-user with a seamless access method to HTC and HPC resources based on Kepler, as illustrated in Figure 6.30. The HELENA code is first computed on HTC resources (using EGEE/EGI resources) and its outcome is further used by the ILSA application code on HPC resources (using DEISA/PRACE).

The resources of the corresponding infrastructures are accessible via the EUFORIA VO of EGEE/EGI and the DEISA EUFORIA virtual community [301]. Whereas the usage of computing resources in EGEE/EGI is essentially free for the EUFORIA VO, the EUFORIA VC has a dedicated number of agreed compute cycles distributed over the DEISA/PRACE production infrastructure.

After discussions with EUFORIA e-Scientists, a high potential of using this workflow with the IIRM and reference architecture was revealed when using Kepler. This case study thus focuses on a an integration into a workflow engine in order to get access to the IIRM model, including proper security mechanisms required to use real existing production e-Science infrastructures. In contrast to subsequent case studies, the requirement of providing a more seamless access to pre-installed software packages and libraries is presented. Also tracking resource usage is highlighted in this use case.

Many scientific fusion applications are very similiar to those that are part of this case study. Other codes of the fusion community such as the fully kinetic massively parallel BIT1 described in [225] have similiar requirements, including the need for necessary libraries [247].

Table 6.12 summarizes the production e-Science infrastructure setup required by the specific fusion case study. It clarifies the question which e-Science infrastructure setup is used with what concrete types of e-Science applications while previous sections reveal general infrastructure setup information in context. The EUFORIA e-Scientists use various types of executables with different computational Grid infrastructure resources (cf. Definition 6).

6.5.3 Academic Analysis and Production Infrastructure Setup Experience

The architectural framework and the use case application experience is analysed in this section to find out how the efficiency of fusion e-Scientists can be improved. The framework runs in production e-Science infrastructure use cases, but the outcome of the analysis points to limitations of the framework and these are summarized in Table 6.13.

The analysis of the general approach of the framework is the first step. Chapter 3 presents a classification of general approaches to interoperability. Among concepts like Additional Layer,

No	e-Science Infrastructure	Setup	e-Science Application Demands
1	EGEE/EGI	Section 6.2.2	HELENA application; embarassingly parallel runs requiring trivial computational power; use of HTC resources feasible;
2	EGEE/EGI	Section 6.2.2	data storage to store intermediate results;
3	DEISA/PRACE	Section 6.2.2	massively parallelized ILSA code using HPC resources; easy access to pre-installed libraries and software necessary;

Table 6.12: Summary of the e-Science infrastructure setups of the EUFORIA case study.

Adapter, Gateway, or Mediator, the Neutral Bridge approach is defined. The first major limitation in the basic EUFORIA framework is the use of the well-known *'(a) Neutral Bridge approach'* to enable interoperability between e-Science infrastructures in general and their corresponding Grid middlewares gLite and UNICORE in particular. In the EUFORIA case study, the neutral bridge is represented by the RAS server that is accessed with the neutral RAS protocol that channels the middleware-specific setups of the Kepler actors through to the bridge.

Apart from this general approach, a more fine-grained academic analysis of lessons learned provides more clarity and points to specific challenges that are addressed to improve the efficiency and supportability of the EUFORIA framework. The use case application HELENA-ILSA workflow revealed several challenges when accessing the different infrastructures. There were *'(b) no common ways of specifying Grid applications'* in a meaningful way (e.g. even without specifying the concrete application executable location) or using pre-installed software packages or libraries. In addition, there have been even limitations on the lowest level with *'(c) having no common way of Grid application execution environments'* (e.g. environment variables). This particular case study also raised the challenge for detailed tracking of computational resource usage. A *'(d) lack of detailed resource usage tracking througout different resources'* was identified when using computational Grid resources in EGEE/EGI, DEISA/PRACE, and also standalone resources such as the dedicated HPC-FF system [42] for the fusion community in Juelich. Tracking the resource usage on these machines including the information of which sub-project of the fusion community is using which resources in a very detailed manner is required in this case study.

Because of these aforementioned limitations and the unavailability of emerging standards in production on the infrastructures, there has been a constant struggle in using the emerging IIRM core building blocks in production setups compared with the proprietary interfaces. From time to time, the proprietary production interfaces have been used again, which was particularly the case in this particular case study. This is clearly not desirable, but once more again a motivational factor to make the standards stronger and more suitable for production.

No.	Limitation short description
(a)	Neutral Bridge (transformation logic)
(b)	No common way of specifying Grid applications
(c)	No common way of Grid application execution environments
(d)	Lack of detailed resource usage tracking througout different resources

Table 6.13: Overview of limitations after academic analysis of the EUFORIA framework.

6.5.4 Reference Model Impact and Applicability

The goal of the concrete reference architecture instance is to improve the work of fusion e-Scientists when using the EUFORIA framework with EGEE/EGI and DEISA/PRACE, as shown in Figure 6.31. This illustration represents a specific *'architecture instance'* of the IIRM and provides insights into the core building blocks of the reference architecture. Several concept refinements (i.e. Δ) to the core building blocks are proposed in order to run scientific fusion applications on Grid resources more easily for EUFORIA e-Scientists. This is important since fusion research uses many different software codes requiring many libraries [247].

This case study uses emerging IIRM core building blocks created from lessons learned that contributed significantly to the findings documented in Chapter 5. An overview of which concrete core building blocks are interesting as part of the EUFORIA case study are provided in Table 6.14. Those concrete refinement concepts that are used in this case in order to overcome several identified limitations of the previous section are indicated with Δ. While Table 6.14 provides a general overview of the used core building blocks of the reference architecture, the next paragraphs provide a step-wise walkthrough of Figure 6.31 in order to explain which refinement concepts are used in detail. Figure 6.31 and Table 6.14 highlight one key impact of the thesis by using IIRM core building blocks and their refinements with another real scientific use case. Table 6.15 summarizes those refinements that matter most in the EUFORIA case study, while it also takes advantage of refinements described in previous case studies (e.g. HPC extensions).

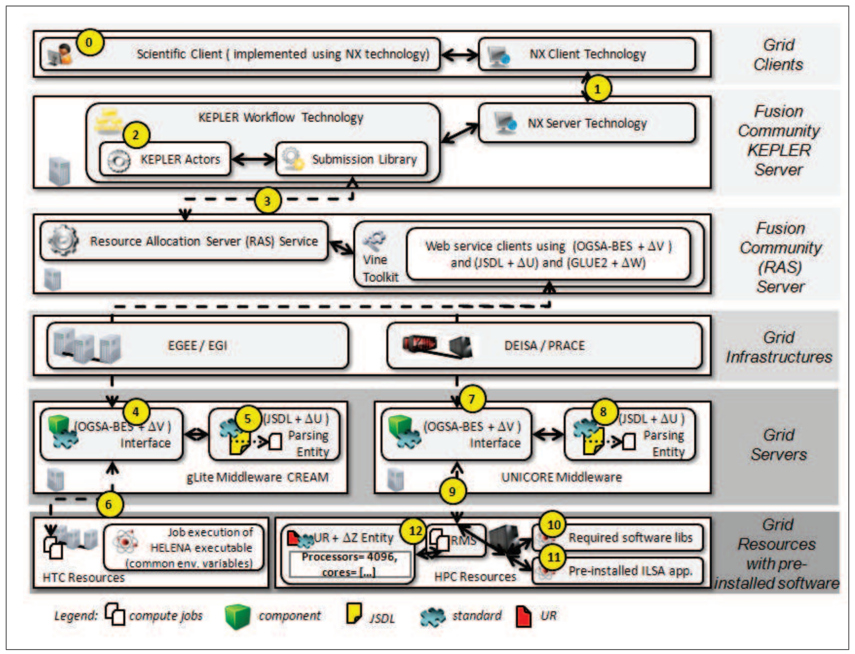


Figure 6.31: Enhanced EUFORIA framework using the IIRM with fusion science applications.

Core building blocks	Specific usage in EUFORIA
UR + ΔZ	Detailed usage record tracking across infrastructures
OGSA-BES + ΔV	WS submission to create a Grid job using middleware systems
JSDL + ΔU	HELENA and ILSA job description; software application concepts
GLUE2 + ΔW	Information about pre-installed fusion software
SAML	Encoding (i.e. SAML assertions) of EUFORIA end-user security attributes
WS-Security	Mechanism to transport SAML assertions in OGSA-BES WS messages
XACML	Security policy for EUFORIA end-users
X.509	Security X.509 certificate provided by each EUFORIA end-user
GIN Exe Env	Environment variables used during application executions

Table 6.14: IIRM core building blocks that are used in the EUFORIA case study.

The yellow numbers indicate the steps explaining the usage of the architecture while at the same time they address how limitations from the previous section are solved. This paragraph focus on the concepts listed in Table 6.15 and thus omits security details since they have been described already in previous case studies whereas this case study is also using the security pattern (cf. Section 5.1.6). Step (0) indicates the usual work situation for EUFORIA e-Scientists by opening a connection to the fusion community server via the NX technology illustrated in step (1). In step (2), the e-Scientist use the Kepler workflow tool with dedicated Kepler actors and their configured identity (i.e. X.509 certificate). The Kepler actors include a submission library that uses via the RAS server the Vine toolkit technology, as illustrated in step (3).

The Vine toolkit in turn uses Web service client classes in order to use core building blocks of the IIRM reference architecture. In step (4), Vine uses the CREAM-BES [220] of gLite in order to submit the HELENA application on a HTC resource within the EGEE/EGI infrastructure. Using open standards and IIRM core building blocks points to another key contribution of this thesis thus avoiding the need for transformation logic to be maintained in Vine toolkit for this particular framework.

In step (5), the JSDL is analysed and the concept of application types classification (cf. Section 5.2.1 (a)) is used to prepare the job submission in an optimized way. HELENA is considered to be a serial application running on an available HTC resource. But the HELENA code needs to be called within a certain location and with various parameters thus making use of our proposed revised application executable definition (cf. Section 5.2.1 (c)). After the end-user processing of the JSDL is finished and the end-user is authorized, the HELENA job is executed on a HTC resource. HELENA as a high resolution equilibrium code can take advantage of the concept of common environment variables (cf. Section 5.2.2 (a)) to optimize its execution. This step is marked as (6) in Figure 6.31 and the concept can be thus used to overcome the identified limitation (c) in the previous section. The intermediate results from the HELENA run is stored in EGEE/EGI storages that are accessible for DEISA/PRACE using different data transfer methods like GridFTP or simple HTTP.

Step (7) stands for various WS message exchanges. This includes the communication with OGSA-BES of UNICORE in order to obtain information about the chosen DEISA/PRACE resources using the `GetFactoryAttributes` operation. In addition to OGSA-BES factory attributes, this operation also exposes the GLUE2 refinements as published in [224]. As Figure 6.31 illustrates, the refinement concept of obtaining information of what software (e.g. pre-installed ILSA) and fusion libraries are installed (cf. Section 5.2.1 (b) and (e)) is used too. This information is re-used with the software requirements and software statements concept in JSDL (cf. Section 5.2.1 (d) and(f)). It enables a precise description of the execution run within JSDL defining the pre-installed ILSA application (including specified required libraries). This overcomes the previously identified limitation (b) by having a common, more detailed, mechanism

Δ	**Functionality Extensions and Improvements**	**Specific usage in EUFORIA**
ΔZ	5.2.6 (a) Tracking of more HPC resource details	ILSA run on HPC
ΔZ	5.2.6 (b) Tracking of VO information	HELENA and ILSA record
ΔZ	5.2.6 (c) Re-use elements of GLUE2	ILSA on modern HPC
ΔU	5.2.1 (a) Application types classification	HELENA as serial; ILSA as parallel
ΔU	5.2.1 (c) Revised application executable definition	in JSDL for HELENA
ΔU	5.2.1 (d) Application software statement	JSDL with pre-installed ILSA
ΔU	5.2.1 (f) Application software requirements	ILSA software requirements
ΔW	5.2.1 (b) Application type refinements	pre-installed ILSA exposure
ΔW	5.2.1 (e) Application Family extension	fusion code libraries exposure
ΔW	5.2.2 (a) Common Environment Variables	ILSA use environment variables

Table 6.15: Functionality improvements used in the EUFORIA case study.

of defining Grid applications. This GLUE2-refined JSDL in turn is used with the second WS message exchange using the OGSA-BES `CreateActivity()` operation. Providing this mechanism illustrates another key impact of this thesis by having easier software and library usage supported by the IIRM in practice.

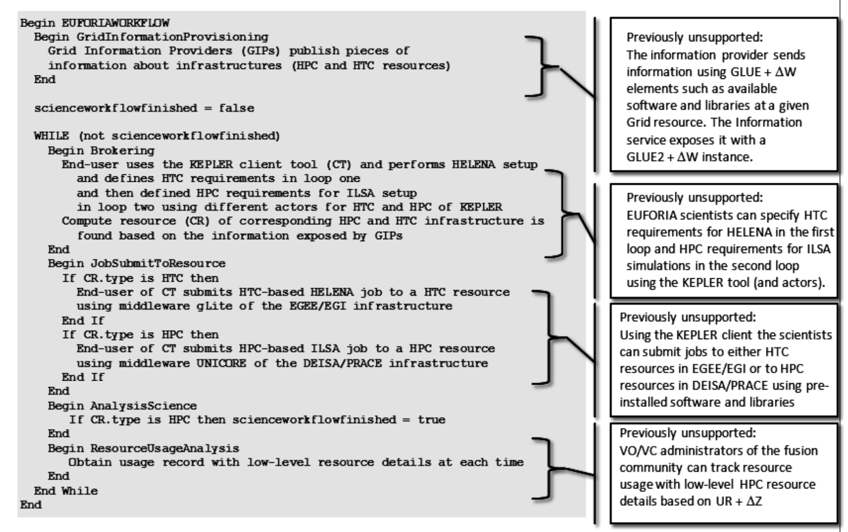
After receiving the JSDL, the OGSA-BES implementation of UNICORE analyses the JSDL in Step (8). It setups the execution run taking advantage of clear definitions about necessary required libraries and the definition of using a pre-installed application definition. The benefit is that end-users do not need to know the exact executable path (e.g. in the HELENA example) and thus calling ILSA remotely is signficantly reduced especially when different libraries are used. In step (9), the well-defined job is forwarded to the RMS for execution making available required libraries in step (10) and schedules in step (11) the ILSA application for its execution on the HPC resource.

In order to overcome limitation (d), step (12) illustrates the refinement concepts of tracking resource usage. This step (12) is highlighted on the HPC resource, because the concept significantly extends the granularity of the UR format for this particular purpose (cf. Section 5.2.6 (a) and (c)) by re-using GLUE2 elements that are also used in other concepts (e.g. amount of cores, etc.). For accounting purposes, the concept of adding the information about VOs (cf. Section 5.2.6 (b)) is used in the particular case in DEISA/PRACE to use the identity of the EUFORIA VC. The VO is thus set as VC since EUFORIA is a VO in EGEE/EGI and also a known VC in DEISA/PRACE. This is in particular crucial, since the same resource might be used without using DEISA/PRACE meaning a local access with the same user id and another computing time contingent.

This case study thus demonstrates an important functionality in terms of accurate resource usage tracking that is also relevant when pool accounts are used (cf. Section 5.2.6). Tracking inside the UR the VO/VC identity (here EUFORIA VC) makes accounting easier, especially when systems are not only locally available at a site, but also shared in production e-Science infrastructures such as DEISA/PRACE.

The overall computation is now significantly easier to setup and to be used with the EUFORIA framework, because e-Scientists are able to simply choose as part of their actors in Kepler which applications and libraries they need for the job run. The framework hides the complexity of the infrastructure setups so that fusion e-Scientists can concentrate on their science rather than knowing the low-level details beyond the RAS server and Kepler.

The overall aforementioned workflow decribes a scientific solution that has been computed within EGEE/EGI and DEISA/PRACE that would not be seamlessly possible without interoperable components of the IIRM. But also the collaborative work of the EUFORIA project was important in terms of Kepler actors (e.g. BES Kepler actor) and Vine toolkit changes. This case



(i) Listing: Pseudo-code of basic design pattern algorithm applied to the EUFORIA case study

Figure 6.32: The EUFORIA scientific workflow can be mapped to the IIRM run-time pattern algorithm.

study relied substantially on the results of the EUFORIA project and its members that have worked in parallel with the author to realize the described use case.

More insights into the aforementioned workflow steps are described in [225] that lead to the more formal definition of the multi-Grid algorithm as illustrated in Figure 6.32. It highlights the previously unsupported elements within this workflow but that are available by using the concepts of the IIRM reference architecture. The pseudo-code notation maps this particular fusion workflow to the the run-time pattern and general algorithm defined in Chapter 5. The IIRM is thus applicable to the fusion workflow and as this workflow is very similiar to others in the fusion science community, the IIRM is also useful there. Evidence that supports this claim is the possibility of mapping the workflow to the general design pattern. This represents another key impact of this thesis by enabling the use of EUFORIA applications with EGEE/EGI and DEISA/PRACE. As a consequence, another key contribution of this thesis is supporting the fusion modelling process as scientific innovation.

The case study benefitted enormously from a collaboration with the DEISA EUFORIA VC [301]. Computational time was available on the HPC-driven Grid infrastructure DEISA/PRACE as well as the dedicated HPC-FF resource in Juelich, whereas EUFORIA is already organized as one VO in EGEE/EGI. Technical interoperability is achieved with the IIRM, but the usage policies for obtaining computational time more conveniently on the rarely available HPC resources remain to be solved by others (e.g. policy groups as described by segment 5 in the aligned process).

6.6 Architecture Implementations for ESFRI and other Applications

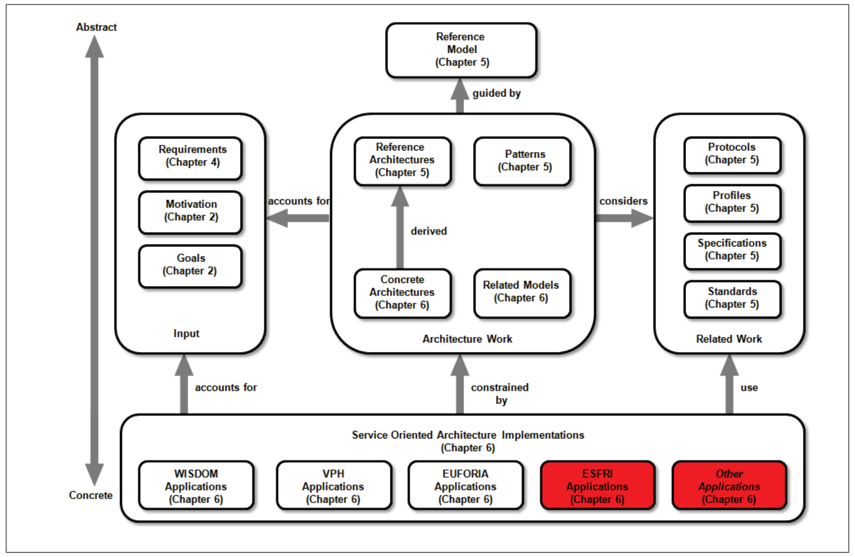


Figure 6.33: SOA implementation for ESFRI and other applications using the concrete reference architecture.

The previous sections describe concrete SOA-based implementation details about the scientific case studies that have been performed alongside the definition of the IIRM and its reference architecture. In contrast, this section reveals further e-Science applications that can take advantage of interoperable infrastructures and the IIRM. The focus is on emerging architectures that are very similiar to the conducted case studies. *'ESFRI and the e-IRG have identified the strong need of ESFRI-projects for ICT tools and access to e-Infrastructures'*[134]. Here, ICT tools can be considered as those provided by the different e-Science infrastructures (aka e-Infrastructures) in Europe. The focus is thus on some example applications that arise from RIs emerging from the ESFRI roadmap and its projects as introduced in Chapter 2.

Figure 6.33 provides an overview resulting from collaborations with some ESFRI projects illustrating the context to the reference model approach and its implementations. The structure of this section is thus different than in previous detailed case studies. One potential architectural framework is proposed in the context of several ESFRI projects, such as CLARIN and the Digital Research Infrastructure for Arts and Humanities (DARIAH) [8] with close relationships to EUDAT [22].

In general, the ESFRI projects have a broader scope as computation that is driving the findings of this thesis. However an analysis of ESFRI requirements performed [161] indicated that *'The sensitive nature of the data to be stored leads to a need for fine-grained Authentication and Authorization system, it also is imperative that such a system provides Single Sign On functionality'* [161]. Furthermore, *'the ability to use grid/cloud compute facilities for the processing of the stored data is also foreseen in some projects'* [161]. Both are just two examples of the synergies between emerging ESFRI RIs and the already established e-Science infrastructures presented in this thesis.

The aforementioned statements point to the need to access computational resources mainly for large-scale data processing where both HTC and HPC resources as needed.

As key contribution of this section insights are given how the IIRM can provide benefits to architectures arising from the ESFRI roadmap and its application requirements. This section thus aims to verify that the reference model is applicable to other application domains as those indicated in the three case studies.

6.6.1 Basic Framework for ESFRI Projects

This section is based on numerous discussions with members of ESFRI projects resulting from several workshops with a particular focus on the ESFRI projects DARIAH [8] and CLARIN [306]. Both CLARIN and DARIAH are emerging European RIs that are organized under the ESFRI umbrella and that consider to follow an architectural approach more in detail described in [271] and that is outlined in Figure 6.34. It describes how a virtual workspace with numerous features can be used by scientific-discipline specific Virtual Research Environments (VREs) following a Web 2.0 YouTube-like design [271]. The impact of this thesis in this particular context is the provisioning of a detailed reference architecture to DARIAH and CLARIN clarifying exactly which interfaces provide access to computational resources (with limited storage capabilities).

Figure 6.34 reveals that the ESFRI RIs are extremely complex but require among many other elements, also HTC and HPC resources that need to be seamlessly integrated into the architectural design. Open standards are a preferred method to access those resources and the IIRM provides a collective set of interfaces in its associated reference architecture as a viable solution

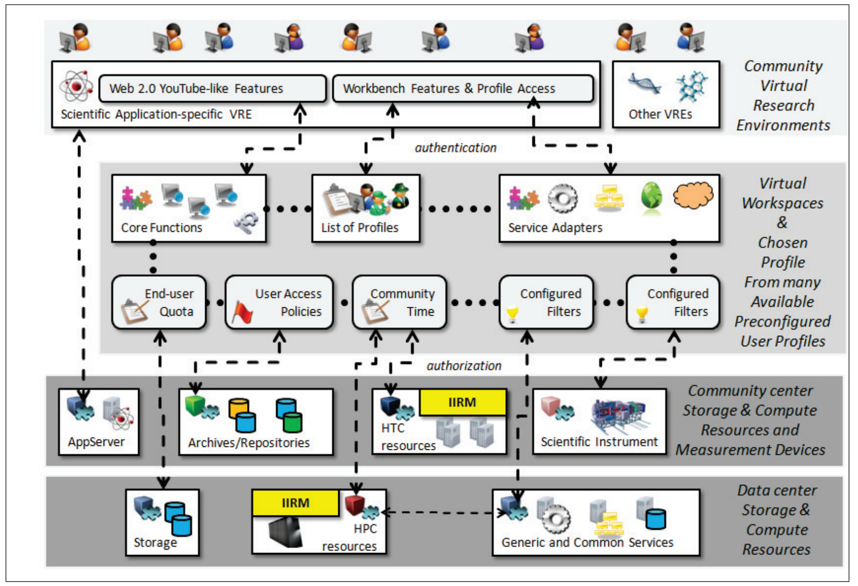


Figure 6.34: Emerging SOA implementations for ESFRI projects such as CLARIN and DARIAH (from [271]).

to seamlessly access those resources. For very computationally intensive problems, European production e-Science infrastructures such as EGEE/EGI or DEISA/PRACE are also required. The context of the proposed IIRM in context of the emerging architectures is illustrated in Figure 6.34.

Complementary elements of the architecture are thin clients that leverage the Web 2.0 technologies within Web browers. Such thin clients should be used to access so-called *'virtual workspaces'* that includes identity and profile membership management. ESFRI e-Scientsts use this workspace like a *'workbench'* to access all their prefered tools and data.

Apart from accessing computational resources with a clear set of interfaces (e.g. such as those provided by the IIRM), the e-Scientists need to access large data archives (e.g. provided by iRods [229]). In the case study CLARIN, such data archives store files and media data used for linguistic research that partly relies on computational methods. The difference to the work in the IIRM is that the digital repositories are directly linked with Persistent Identifiers (PIDs) such as those provided by European Persistent Identifier Consortium (EPIC) [21]. Microservices of iRods are used to update PIDs for each element in the digital repository.

The WebLicht [192] workflow tool within the ESFRI CLARIN infrastructure [306] is one preferred tool that consists of 180 Web services that call different linguistic tools in a chain. Some steps in this chain are very much computationally intensive and in this context the interfaces to the IIRM are used as part of this particular tool by using Web services communication with the IIRM core building blocks if needed. The WebLicht can be considered as one of the generic community services also illustrated as part of Figure 6.34.

In terms of authorization and authentication slightly different methods are envisaged in comparison with the IIRM, but that are still similiar to those provided by the IIRM. Many ESFRI research infrastructures aim to consider Shibboleth [232] as their preferred security mechanism. This security technology is based on the SAML standard that in turn is also one core building block of the IIRM reference architecture, but used in a sligthly different manner. Minor work needs to be done to agree on common end-user security attribute formats since they are different from those released from a VOMS server.

Other RI elements are less related to the thesis such as the access to scientific instruments illustrated in Figure 6.34 to enable the access to measurement devices and other form of physical device. Other services such as Persistent Identifier (PID) services [21] enable the reference to scientific data over decades to come. More detailed descriptions of the services and functionality are available in [271] and only interesting for this thesis to a limited degree. The infrastructures arising from the various ESFRI projects are not fundamentally different from those known as production e-Science infrastructures over years. But e-Science infrastructures have been majorly focussed on computation, while the ESFRI projects aim to have more belance between resources in general and data, instruments, and computations in particular.

Many aspects of the IIRM and its reference architecture are useful to ESFRI research infrastructures as well, because these infrastructures, like the production e-Science infrastructures, represent distributed systems facing similiar challenges in terms of technology provisioning, its sustainability, and most notably its interoperability. This is another key contribution of this thesis by providing a specific reference architecture currently usable by ESFRI projects when computational capabilities are required.

6.7 Conclusion

The proposed reference model was abstractly described in Chapter 5, while this Chapter describes the impact on production e-Science technologies and infrastructures using concrete examples and case studies. The proposed IIRM has an impact on European and US-based production e-Science infrastructures, including a limited extend to those in Asia through Japan. Further evidence of the scientific impact of the IIRM is provided through accompanying case studies from the bio-informatics, e-Health, and fusion domain.

The seven segment-based process implementation leads to major scientific innovation in terms of technologies through the reference model and in terms of applications through the means of our collaboratively performed case studies. Each implementation of the seven segments contributed to the overall impact of the thesis findings, including the important factor of being significant for real production e-Science infrastructures as the major research question required. The reference model in general and its applicability in particular in this chapter provides community-proven evidence (e.g. papers, deployments, etc.) that the IIRM and its concrete emerging deployments provide an answer to the given research question. Various publications of the concepts are available using the proposed reference architecture core building blocks that provide evidence that the approach of IIRM in e-Science is meaningful. Many prototype developments over years have been provided that led to the technical soundness of the IIRM today and to the funding of multi-million projects like EMI and XSEDE that are both delivering production implementations for SOA-based architectures that are based on the IIRM design and its associated reference architecture. The aligment of the EMI work plans with the ideas of the IIRM and the influence on XSEDE was a significant part to improve the interoperability between European and US-based e-Science infrastructures. The detailed refinements concepts of open standards that represent core building blocks are contributions of this thesis while several of the detailed implementations are the implementations of others for which the author takes no credit for.

Instead, to prove that the IIRM has a real scientific impact, several scientific case studies are provided with underpinning publications in both technology (i.e. concepts) and science (i.e. application use case papers). The WISDOM case study takes advantage of the established interoperability via the IIRM and several of its refinements concepts. It contributed enormously to the findings in Chapter 5, but the fact that the WISDOM workflow works with the IIRM provides evidence that the IIRM can work not only with WISDOM applications but also with a wide variety of applications from the bio-informatics domain (e.g. other MD simulation programs).

The VPH case study takes also advantage of the established interoperability, but with a different focus as WISDOM by using primarily HPC-driven e-Science infrastructures and as such the refinements of the core building blocks in order to enable more efficient runs on HPC resources. Also this case study provides valuable lessons learned to the IIRM design in Chapter 5, but the fact that this case study works with the IIRM also provides evidence that similiar HPC codes can be used than just those of the e-Health domain. The EUFORIA case study also takes advantage of multiple e-Science infrastructures, but in a manner of integrating the IIRM concepts into a larger framework driven by the fusion community. This use case, among with the VPH AHE evidence, prooves that the IIRM and its associate reference architecture is even applicable when its core building blocks need to be integrated with existing frameworks such as the AHE in VPH or, more motably, the complex RAS server environment of EUFORIA. Finally, an initial impact of the thesis work in the context of more complex RIs arising from the ESFRI roadmap are given in general and from those in the context of DARIAH and CLARIN in particular.

Chapter 7

Conclusion

In the first chapter of the thesis, objectives have been defined in terms of five major contributions and the major research question was formulated: *'How a reference model for a network of interoperable services in production e-Science infrastructures can be defined?'* In this chapter, the thesis work is reviewed and set it in the context of the desired results in order to understand how the objectives have been achieved. Evidence is provided in context of earlier Chapters that support the claims and the collective thesis contributions represent one possible answer to the major research question.

The first major contribution of the thesis is the Infrastructure Interoperability Reference Model and its associated architecture work, as described in Chapter 5, that altogether represent a trimmed-down version of the Open Grid Services Architecture in terms of functionality and complexity. The overall IIRM design follows the TCP/IP approach (not ISO/OSI) that has been proven to be succesful in the Web domain. The IIRM and its reference architecture is thus more compact, but more specific than OGSA being thus also more production-oriented to be used specifically with production technologies today. Evidence is provided in Chapter 6 where the IIRM is compared with OGSA according to the production reference model factors and indicators that have been defined after a thorough analysis of OGSA and related work in Chapter 3. Further evidence is provided in Chapter 6 indicating that the IIRM is being implemented in various Grid middleware technologies that are all already deployed in production e-Science infrastructures (i.e. EGEE/EGI, DEISA/PRACE, TeraGrid/XSEDE, etc.).

The history of computer science shows that often complex architectures (e.g. OGSA) are used less than their trimmed-down versions (e.g. IIRM). For instance, the complex Structured Generalized Markup Language (SGML) is used less than its smaller version well-known as Extensible Markup Language (XML). XML is less complex and therefore quickly became a de facto standard in Web data processing. The same principles can be exploited with OGSA by defining a more limited but more focussed infrastructure reference model leading to the IIRM and its associated architecture. This becomes increasingly important in the context of economic constraints since the huge OGSA service ecosystem requires massive amounts of maintenance whereas the reference model significantly reduces these maintenance costs by providing only a small, but required basic subset of functionality. This is done in a more well-defined manner tuned to today's production usage. The case studies of Chapter 6 revealed that the IIRM has already started to contribute to major research advances in science and engineering via developments from key technology providers such as EMI covering IIRM implementations of ARC, gLite, and UNICORE.

Complementary to the academic investigations, the seven segment-based process, was defined and implemented as a complementary applied research method that brought together inputs from various sources (e.g. e-IRG, EU e-Infrastructure concertation activities, EEF reports, EICTA experience, etc.). This process is another complementary contribution to the IIRM and its architecture work. As described in Chapter 5, the seven segment-based process deals with change and policy aspects of production e-Science infrastructures and was implemented through this thesis over years leading to real impacts of existing production e-Science infrastructures rather than on academic testbeds or prototype environments. By implementing this process, in addition to the academic results, further relevance and impact of this thesis in real production environments is given through collaborating with the right set of partners, projects, and infrastructures as described in Chapter 6.

The second major contribution of this thesis results from an academic analysis of lessons learned and field experience obtained by using the core building blocks (i.e. open standards) of the proposed IIRM reference architecture in production e-Science infrastructures. Over time, these results influenced the architectural design of the IIRM and the associated architecture work that was complemented with several *'refinement concepts'* of how the efficiency of the core building blocks could be improved to specifically support e-Science applications. These results also provide evidence that the IIRM goes far beyond what appears to be a combination of different Grid components such as taking SRM from gLite, SAML from UNICORE, OGSA-BES from GENESIS, and GridFTP from Globus. Instead, over several years, various inputs to the academic analysis have been gathered by using early production versions and best practices of emerging open standards and prototypes within production environments with scientific applications. Hence, these applications go beyond the *'bin-date'* demonstration use case and is more a production-oriented typical usage of an existing Grid middleware in one infrastructure. These results influenced the design of the reference model and afterwards also the standardization activities of OGF where a whole group ecosystem with GIN, PGI, and other groups (e.g. OGSA-BES, JSDL, etc.) was created. Here, the goal was to influence new emerging versions of open standards with lessons learned obtained from production and an academic critical review of existing approaches offering several concept improvements and refinements. As part of the seven segment-based process implementation, the feedback within PGI thus influences the next generation of open standards.

But the concrete results of the standardization process (i.e. concrete normative specifications) are not part of this thesis since this is a community process that is already ongoing and that this thesis only influences to a certain degree. But this thesis provides significant input to this standardization process via many publications in the field of how standards can be improved or work together as an ecosystem under one common *'reference model umbrella'*. This thesis thus not includes the final OGSA-BES 1.1 or JSDL 1.1 specifications although in practice the thesis results are one of the key contributions to such specifications with the proposed refinements (e.g. OGSA-BES + ΔV) resulting from academic studies.

The third major contribution is twofold. First, *'missing links'* between open standards specifications have been identified through the academic analysis of lessons learned and from the work with real scientific applications. For example, the GLUE2 and JSDL dependencies are shown throughout this thesis and represent one important missing link that needs to be filled. The use of security specifications with others (e.g. SAML with OGSA-BES) is another filled link. Various publications of this thesis concepts have been proposed in the past and as part of this thesis in this regard. Second, GIN and PGI have been created as another applied research process segment that is able to address the challenge to provide a feedback channel of standard experience to the right groups. This is difficult since the usage of standards are highly interlinked but still standardized in isolation from each other and neglecting their important relationships and interactions in many ways. The academic analysis revealed the necessity of a new *'interdisciplinary research method'* of how missing links between core building blocks of the IIRM reference architecture can be found going beyond only one technical area. One further example in this context is the work with SAML assertions instead of VOMS proxies to convey security attributes of end-users during an OGSA-BES-based job submission and deriving authorization decisions based on SAML with XACML policies. This is another missing link published as one of the first results of the thesis with members of OMII-Europe in 2007 where neither UNICORE 5 nor gLite used SAML in this specific way.

By aligning the IIRM and its reference architecture around open standards and by providing an associated *'standardization feedback ecosystem'* as part of the process, building blocks are constructed that are able to respond to *'dynamic changes'* over time. This is given only when the original concepts of the proposed standards are not completely changed, like for example, a strange setup where OGSA-BES would be used as usage record schema instead for job submission as it is used today.

The fourth major contribution is the work around the IIRM associated *'design pattern'* using HPC resources together with HTC resources to run a scientific workflow leading to scientific innovation by using resources across different infrastructures. In the majority of the scientific applications and case studies analysed, e-Scientists still take advantage of using one computational paradigm for their science, essentially being also limited to one production e-Science infrastructure. The science in the case studies is already extremely complex and thus e-Scientists should not deal with computational barriers in using HPC and HTC in order to focus on their scientific challenges, which means that they do not need to know the underlying interfaces and habits of different production e-Science infrastructures. The thesis thus in general and the IIRM in particular focused on providing a *'frame of reference'* for using HPC and HTC together with the same technologies, including missing concepts obtained from an academic analysis of production usage and constraints of real e-Science infrastructures.

The aforementioned aspect is evidence of changing the foundation of a reference model from intra-Grid to across infrastructures well focussed on the *'problem of interoperability'*. Another piece of evidence is that the seven segment-process is associated with the IIRM and it was presented several times in important community events organized by EGI, EMI, or SIENA in order to promote the idea of *'interoperability by design'* one level higher than on the purely technical level. The possibilities on the technical level are limited, especially when the Grid setup is compared with the electrical power Grid setup outlined at the beginning of this thesis. Over decades, even the powerful and financially well equipped technical community around electrical power Grids has still not significantly improved interoperability, so that different adapters are still required in countries world-wide today. This situation is similiar in the Grid community by indicating clear barriers for technical interoperability that can only be solved by solutions that are conducted by, for example, harmonizing technologies through means of close collaboration, in order to reach full interoperability in the long-term. The proposed seven segment-based process is a proposed *'guiding solution'* to this problem that was considered as a good example at the SIENA CloudScape-III event in Brussels where the segments were presented as input to the SIENA standardization roadmap. Because some segments of the process are not only technical in their nature, we acknowledge that their are other views on the segments and its impact. But the key contributions of this thesis remains the clearly technical work around the IIRM and its architecture as well as its validation with use case applications.

The fifth major contribution of this thesis delivers evidence for the claims that the IIRM and its associated architecture work not only has an impact on real production e-Science infrastructures, but also through the deployment leads to scientific innovation. This innovation, that goes beyond the traditional methods (e.g. shell-based scripts, etc.), is described via three accompanying case studies from different scientific fields covering bio-informatics, e-Health, and also fusion science. These three case studies of WISDOM, VPH, and EUFORIA provide evidence of the claim that the IIRM reference architecture works with different existing frameworks such as scientific community tools (Kepler, AHE, etc.) or *'thin client approaches'* such as Web browsers (e.g. GridSphere WISDOM portal). All these tools hide the complexity of the underlying production e-Science infrastructures using parts of the IIRM architecture towards the seamless execution of scientific workflows across the multiple infrastructures existing in Europe, the US, and Japan. With the contributions of this thesis, e-Scientists as part of these case studies, are able to create *'individually formed e-Science infrastructures'* according to their needs in terms of computational or storage resources.

The same is true in terms of middleware usage being basically not forced to use one specific Grid middleware technology enhancing thus users choice. From a technical perspective, they can just choose the technology they want as long as it is conforms with the IIRM reference architecture that at least cover the basic key infrastructure functionalities.

Further evidence is provided that some production Grid middleware already emerges that implement IIRM concepts and associated architecture elements such as within gLite, ARC, UNICORE via EMI in Europe, or others like GENESIS and Globus in the US, or even RENKEI in Japan.

The thesis findings do not cover the concrete and exact implementations within these middlewares that may defer in terms of specific realization while the overall concepts are still being adopted in one form or another as Chapter 6 revealed. General concepts with hints for possible realizations have been part of this thesis while in some cases several other realizations of the same concept can be achieved.

But these concepts and their prototypes verified the IIRM approach as well as emerging open standard implementations. But the production quality implementations are provided by the middleware consortia, which in turn is a process the author takes no credit for. It is the implementation of the seven segment-based process and the various publications as evidence that underpins that the thesis work in the context of production middleware and infrastructure matters is relevant mainly because of using thesis aspects in key Grid community projects like OMII-Europe, DEISA2, EMI, EGI, and XSEDE as well as the contributions to the OGF standardization activities via GIN, PGI, and other groups over years.

This supports the claim that the IIRM, its associated architecture work, and the aligned seven segment-based process is providing e-Scientists with an approach that has real impact on existing production e-Science infrastructures. This is true regardless of the fact that there is a period of *'infrastructure change'* such as the transition of EGEE to EGI, the integration of DEISA2 sites into PRACE, and the establishment of XSEDE integrating resources from the previous TeraGrid (indirectly including OSG as resource provider). All these are particularly not as important as the transition process is mainly an infrastructure evolution providing the same underlying resources according to the core computational paradigms HPC and HTC. As a consequence, and having verified in Chapter 6 that the thesis works with real scientific use cases, the overall conclusion is that the thesis provides one of more possible answers to the major research question: the IIRM and its associated architecture elements as well as its complementary process define a reference model (here in the sense of an umbrella term, cf. Chapter 3) leading to a network of interoperable services in production e-Science infrastructures. This enables scientific innovation and e-Science breakthroughs on an unprecedented scale and will continue over the next decade.

At the end of this thesis, some directions of future work are given. In Chapter 6, insights on ESFRI application projects such as CLARIN and DARIAH that can take advantage of the IIRM and its reference architecture have been given. Future work might include to obtain new requirements and concept refinements when the IIRM is used as part of other ESFRI RIs.

On the technical side, future work might be the academic analysis of the next generation of open standards that are not yet reflected as part of the reference architecture. One concrete example is DRMAA version 2.0 that is currently standardized and could be an interesting extension to the IIRM and its associated reference architecture. Another interesting area of future work might include the extension towards other standards used in TeraGrid/XSEDE that are currently not part of the reference architecture such as RNS for example. Another academic analysis might be worthwhile to understand whether and how cloud-based e-Science infrastructures can be used with the core building blocks (e.g. within virtual appliances of virtual machines). This not only include new research challenges such as dynamic deployment and security models, but also the exploration of whether the IIRM is useful for e-Business although being focussed on the scientific needs of the next decade in distributed computing.

List of Acronyms

AA - Attribute Authority
AHE - Application Hosting Environment
AJO - Abstract Job Object
AMBER - Assisted Model Building with Energy Refinement
API - Application Programming Interfaces
ARC - Advanced Resource Connector
ARIS - ARC Information System
AMGA - Arda Metadata Grid Application
BDII - Berkeley Database Information Index
BOINC - Berkeley Open Infrastructure for Network Computing
CCA - Common Component Architecture
CCM - Cluster Compatibility Mode
CE - Computing Element
CIM - Common Information Model
CIS - Common Information System
CLARIN - Common Language Resources and Technology Infrastructure
CPN - Colored Petri Nets
CR - Computational Resource
CREAM - Computing Resource Execution and Management
CSA - Service Component Architecture
DAG - Directed Acyclic Graph
DARIAH - Digital Research Infrastructure for Arts and Humanities
DCI - Distributed Computing Infrastructure
DECI - DEISA Extreme Computing Initiative
DEISA - Distributed European Infrastructure for Supercomputing Applications
DGAS - Distributed Grid Accounting System
DMTF - Distributed Management Task Force
DPM - Disk Pool Manager
DRMAA - Distributed Resource Management Application API
E-IRG - e-Infrastructure Reflection Group
EDGES - Enabling Desktop Grids for e-Science
EDGI - European Desktop Grid Initiative
EEF - European e-Infrastrucure Forum
EGEE - Enabling Grids for e-Science
EGA - Enterprise Grid Alliance
EGI - European Grid Initiative
EICTA - European Informations, Communications, and Consumer Electronics Industry Technology Association
EMI - European Middleware Initiative
EMI-ES - EMI Execution Service
EPIC - European Persistent Identifier Consortium
ESFRI - European Strategy Forum on Research Infrastructures
ESM - Extreme Scalability Mode
ETSI - European Telecommunications Standards Institute

ETTF - Education and Training Task Force
EUDAT - European Data Infrastructure
EUFORIA - EU Fusion for ITER Applications
FQAN - Fully Qualified Attribute Name
FSP - File Staging Profile
FTP - File Transfer Protocol
GAAI - Global Authorisation Attributes Invariant
GAI - Global Accounting Invariant
GAL - Grid Abstraction Layer
GENIUS - Grid Enabled Neurosurgical Imaging Using Simulation
GGF - Global Grid Forum
GII - Global Information Invariant
GIN - Grid Interoperation Now
GILDA - Grid INFN Laboratory for Dissemination Activities
GOS - Grid Operating System
GPE - Grid Programming Environment
GRIA - Grid Resources for Industrial Applications
GSI - Grid Security Infrastructure
HPC - High Performance Computing
HPC FSP - HPC File Staging Profile
HTC - High Throughput Computing
HTTP - Hypertext Transfer Protocol
IETF - Internet Engineering Task Force
IGE - Initiative for Globus in Europe
IGIIW - Int. Grid Interoperability and Interoperation Workshop
IIRM - Infrastructure Interoperability Reference Model
IP - Internet Protocol
ISO - International Standards Organization
ISV - Independent Software Vendor
ITER - International Thermonuclear Experimental Research
JDL - Job Definition Language
JSDL - Job Submissiona and Description Language
JSPG - Joint Security Policy Group
LDAP - Lightweight Directory Access Protocol
LHC - Large Hadron Collider
LL - LoadLeveler
LSF - Load Sharing Facility
MD - Molecular Dynamics
MHD - Magnetohydrodynamics
MPC - Multi-Particle Collision Dynamics
MPI - Message Passing Interface
MRT - Magnetic Resonance Tomograph
MWSG - Middleware Security Group
NATO - North Atlantic Treaty Organization
NDGF - Nordic DataGrid Federation
NGI - National Grid Initiative
NGS - National Grid Service
NJS - Network Job Supervisor
NSF - National Science Foundation
OASIS - Organization for the Advancement of Structured Information Standards
ODP - Open Distributed Processing
OGF - Open Grid Forum
OGSA - Open Grid Services Architecture
OGSA-BES - OGSA - Basic Execution Services

OGSA-DAI - OGSA - Database Access and Integration
OGSA-RUS - OGSA - Resource Usage Service
OGSI - Open Grid Services Infrastructure
OMII - Open Middleware Infrastructure Institute
OSG - Open Science Grid
OSI - Open Systems Interconnection
PEPC - Pretty Efficient Parallel Coulomb-Solver
PGI - Production Grid Infrastructure
PID - Persistent Identifier
PKI - Public Key Infrastructure
PRACE - Partnership for Advanced Computing in Europe
PTP - Parallel Tools Platform
QSAR - Quantitative Structure-Activity Relationships
RAS - Resource Allocation Server
REB - Roadmap Editorial Board
RENKEI - Resources liNKage for E-scIence
REST - Representational State Transfer
RI - Research Infrastructure
RM-ODP - Reference Model for Open Distributed Processing
RMS - Resource Management System
RNS - Resource Namespace Specfication
RPC - Remote Procedure Call
RSL - Resource Specification Language
RUS - Resource Usage Service
SAGA - Simple API for Grid Applications
SAML - Security Assertion Markup Language
SGAS - Swedish Grid Accounting System
SGML - Structured Generalized Markup Language
SCP - Secure Copy
SDK - Software Development Kit
SDO - Standard Development Organization
SDOB - Service Data Object
SIENA - Standards and Interoperability for e-Infrastructure Implementation Initiative
SLA - Service Level Agreements
SMS - Storage Management Service
SOA - Service Oriented Architecture
SOAP - Simple Object Access Protocol
SPG - Security Policy Group
SPMD - Single-Program-Multiple-Data
SRM - Storage Resource Manager
SSH - Secure Shell
SSL - Secure Socket Layer **STEP** - Strategy for the EuroPhysiome
SQL - Structured Query Language
TCP - Transmission Control Protocol
TSI - Transport Layer Security
TSI - Target System Interface
UAB - User Advisory Board
UAS - UNICORE Atomic Services
UDP - User Datagram Protocol
UMD - Unified Middleware Distribution
UNICORE - Uniform Interface to Computing Resources
UR - Usage Record
UVOS - UNICORE VO Service
VC - Virtual Community

VDT - Virtual Data Toolkit
VISIT - Visualization Interface Toolkit
VO - Virtual Organization
VOMS - Virtual Organization Membership Service
VPH - Virtual Physiological Human
VRE - Virtual Research Environment
W3C - World-Wide Web Consortium
WISDOM - Wide In-Silico Docking on Malaria
WLCG - Worldwide Large Hadron Collider (LHC) Computing Grid
WMS - Workload Management System
WS - Web Services
WS-BPEL - WS Business Process and Execution Language
WS-DAI - WS-Data Access and Integration
WSDL - Web Services Description Language
WS-I - Web Services Interoperability
WSRF - Web Services Resource Framework
WS-S - Web Services Security
XACML - Extensible Access Control Markup Language
XML - Extensible Markup Language
XSD - XML Schema Definition
XSEDE - eXtreme Science and Engineering Discovery Environment

References

[1] GEANT2, Infrastructure. http://www.geant2.net/.
[2] BISGRID. https://bi.offis.de/bisgrid/tiki-index.php, September 2012.
[3] CN - Grid. http://blog.sina.com.cn/webhpc, September 2012.
[4] Common Component Architecture Forum. http://www.cca-forum.org, September 2012.
[5] Common Information Model. http://www.dmtf.org/standards/cim/, September 2012.
[6] CRAY XE6 Supercomputer. www.cray.com/Products/XE/CrayXE6System.aspx, September 2012.
[7] CRAY XE6 Supercomputer Modes. www.cray.com/Products/XK6/SolutionPartners.aspx, September 2012.
[8] DARIAH. http://www.dariah.eu, September 2012.
[9] Database Access and Integration Services (DAIS). https://forge.gridforum.org/sf/go/proj1070, September 2012.
[10] DEISA Trouble Ticket System. http://tts.deisa.eu, September 2012.
[11] Disk Pool Manager. http://twiki.cern.ch/twiki/bin/view/LCG/DpmInformation, September 2012.
[12] Distributed Computing Infrastructure Interoperability Minisymposium at PARA 2010. http://yourhost.is/para2010/scientific-programme.html, September 2012.
[13] EGI Security Policy Group. www.egi.eu/wiki/SPG, September 2012.
[14] EGI UMD Repository. www.repository.egi.eu, September 2012.
[15] EMI Description of Work. https://twiki.cern.ch/twiki/bin/view/EMI/EmiDocuments#Public_Description_of_Work_DoW, September 2012.
[16] EMI FutureGrid Project. www.portal.futuregrid.org/projects/170, September 2012.
[17] EMI Security Token Service. www.eu-emi.eu/security-token, September 2012.
[18] Enabling Desktop Grids for e-Science. http://www.edges-grid.eu, September 2012.
[19] Enhanced-Science (e-Science) Definition. http://www.e-science.clrc.ac.uk, September 2012.
[20] Environment Modules. http://modules.sourceforge.net, September 2012.
[21] EPIC. www.pidconsortium.eu, September 2012.
[22] EUDAT. www.eudat.eu, September 2012.
[23] European - e-Infrastructure Forum. http://www.einfrastructure-forum.eu, September 2012.
[24] European Desktop Grid Initiative. http://www.edgi-project.eu, September 2012.
[25] European Grid Infrastructure. http://www.egi.org/, September 2012.
[26] European Grid Infrastructure - Technical Forum 2010. www.egi.eu/EGITF2010, September 2012.
[27] European Middleware Initiative. http://www.eu-emi.eu, September 2012.
[28] European Persistent Identifier Consortium (EPIC). www.pidconsortium.eu, September 2012.
[29] Extreme Science Engineering and Discovery Environment (XSEDE). www.xsede.org, September 2012.
[30] FlexX. http://www.biosolveit.de/FlexX, September 2012.
[31] From Open Grid Services Infrastructure to Web Services Resource Framework: Refactoring and Evolution. http://www.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf, September 2012.
[32] FUJITSU K Supercomputer. www.fujitsu.com/global/about/tech/k, September 2012.
[33] FUJITSU K Supercomputer - Network. www.fujitsu.com/global/about/tech/k/whatis/network, September 2012.
[34] FutureGrid. http://www.futuregrid.org, September 2012.
[35] Gaussian. http://www.gaussian.com, September 2012.
[36] GENIUS Project. wiki.realitygrid.org/wiki/GENIUS, September 2012.
[37] GGUS. http://www.ggus.eu, September 2012.
[38] GIN Execution Environment Draft. http://forge.gridforum.org/sf/docman/do/listDocuments/projects.gin/docman.root.current_drafts.execution_environment, September 2012.
[39] Globus Online. www.globusonline.org, September 2012.
[40] Grid interoperation now in euforia: Using egee and deisa for fusion science. http://zam581.zam.kfa-juelich.de/jsc/news/sc09/tuesday, September 2012.
[41] GRIDSITE Delegation Service. http://www.gridsite.org, September 2012.
[42] HPC-FF. www.fz-juelich.de/ias/jsc/EN/Expertise/Supercomputers/JUROPA/JUROPA_node.html, September 2012.
[43] Hyper Text Transfer Protocol - HTTP. http://www.w3.org/Protocols/, September 2012.
[44] IGE 2.0 Release. www.ige-project.eu/downloads/software/releases/200, September 2012.

[45] Initiative for Globus in Europe. http://www.ige-project.eu, September 2012.
[46] International Grid Interoperation and Interoperability Workshop (IGIIW) 2007. http://www.e-science2007.org/Workshops.htm, September 2012.
[47] International Grid Interoperation and Interoperability Workshop (IGIIW) 2008. http://www.e-science2008.iu.edu/workshops/international/index.shtml, September 2012.
[48] International Standards Organisation. http://www.iso.org, September 2012.
[49] International Telecommunication Union - Telecommunication Sector. www.itu.int/ITU-T, September 2012.
[50] Joint Security Policy Group. www.jspg.org, September 2012.
[51] JUGENE Supercomputer. http://www.fz-juelich.de/jsc/jugene, September 2012.
[52] JUMP Supercomputer. http://www.fz-juelich.de/jsc/jump, September 2012.
[53] Load Sharing Facility. http://www.platform.com/Products/Platform.LSF.Family/, September 2012.
[54] Media Video: PGI OGF25. http://www.youtube.com/watch?v=LAS1MCrtI_s, September 2012.
[55] MOLPRO Quantum Chemistry Package. http://www.molpro.net, September 2012.
[56] MWSG. www.technical.eu-egee.org/index.php?id=146, September 2012.
[57] MWSG Meeting Zuerich. www.indico.cern.ch/conferenceDisplay.py?confId=52862, September 2012.
[58] National Grid Initiatives. https://wiki.egi.eu/wiki/NGI, September 2012.
[59] National Grid Service. http://www.ngs.ac.org, September 2012.
[60] NGI - DE. http://www.ngi-de.eu, September 2012.
[61] Nordic Data Grid Facility. http://www.ndgf.org/, September 2012.
[62] NX - Technology. http://www.nomachine.com, September 2012.
[63] OASIS WS-Interoperability (merged with oasis). http://www.oasis-ws-i.org, September 2012.
[64] OGF - Europe Project. http://www.ogfeurope.eu, September 2012.
[65] OGF OGSA - Resource Usage Service (OGSA-RUS) Working Group. https://forge.gridforum.org/projects/rus-wg, September 2012.
[66] OGF Reference Model Working Group. https://forge.gridforum.org/sf/projects/rm-wg, September 2012.
[67] OGF25 PGI Session. www.ogf.org/gf/eventschedule/index.php?=1571, September 2012.
[68] Open Grid Forum. http://www.ogf.org/, September 2012.
[69] Open Middleware Infrastructure Institute - Europe. http://omii-europe.org/, September 2012.
[70] Open Middleware Infrastructure Institute - UK. http://www.epcc.ed.ac.uk/projects/omii-uk, September 2012.
[71] Open Service Component Architecture (CSA). http://www.oasis-opencsa.org, September 2012.
[72] Open Service Data Objects (SDO). http://www.oasis-opencsa.org/sdo, September 2012.
[73] Organization for the Advancement of Structured Information Standards. http://www.oasis-open.org/home/index.php, September 2012.
[74] Parallel Tools Platform. http://www.eclipse.org/ptp, September 2012.
[75] PBSPro. http://www.altair.com/software/pbspro.htm, September 2012.
[76] PGI JSDL Iinputs. www.forge.org.org/sf/go/doc16293, September 2012.
[77] PGI Requirements List. www.forge.org.org/sf/go/doc16080, September 2012.
[78] PRACE deploys UNICORE. www.prace-ri.eu/PRACE-partner-Julich-joins-XSEDE, September 2012.
[79] Press Article: International Science Grid This Week (isgtw) - Interoperability Reference Model. http://archive.isgtw.org/?pid=1001481, September 2012.
[80] Press Article: Juelicher Software fuer US-supercomputing. http://www.silicon.de/technologie/software/0,39044013,41555657,00/juelicher_software_fuer_us_supercomputing.htm, September 2012.
[81] Pressemitteilung: 121 Millionen Dollar Projekt mit Juelicher Beteiligung. http://www.fz-juelich.de/portal/DE/Presse/Kurznachrichten/2011/august2011.html#doc1040726bodyText4, September 2012.
[82] Reference Model for Open Distributed Processing. http://www.rm-odp.net, September 2012.
[83] RENKEI. http://www.e-scien.org/index-e.html, September 2012.
[84] REsources liNKage for E-scIence - RENKEI. www.e-sciren.org/publications/sc10.pdf, September 2012.
[85] SCIENCESOFT. www.sciencesoft.org, September 2012.
[86] SHARE. http://www.share-project.org, September 2012.
[87] SIENA. http://www.sienainitiative.eu, September 2012.
[88] SIENA - CloudScape III. http://www.sieneinitiative.eu/StaticPage/Cloudscape.aspx, September 2012.
[89] SIMDAT. http://www.simdat.org, September 2012.
[90] SMARTLM. http://www.smartlm.eu, September 2012.
[91] Solving Grad-Shafranov numerically. https://perswww.kuleuven.be/~u0016541/Talks/helena.pdf, September 2012.
[92] STEP - Constortium: Seeding the EuroPhysiome - A Roadmap to the Virtual Physiological Human (VPH). http://www.europhysiome.org/roadmap, September 2012.
[93] StratusLab. http://www.stratuslab.eu, September 2012.
[94] Supercomputing Conference 2007. http://sc07.supercomp.org, September 2012.

[95] Supercomputing Conference 2008. http://sc08.supercomp.org, September 2012.
[96] Supercomputing Conference 2009. http://sc09.supercomp.org, September 2012.
[97] Supercomputing Conference 2010. http://sc10.supercomp.org, September 2012.
[98] Torque Resource Manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php, September 2012.
[99] VENUS-C. http://www.venus-c.eu, September 2012.
[100] Virtual Data Toolkit. http://vdt.cs.wisc.edu/, September 2012.
[101] W3C - Web Services Activity. http://www.w3.org/2002/ws/, September 2012.
[102] WISDOM - Webpage. http://wisdom.healthgrid.org/, September 2012.
[103] World Wide Web Consortium. http://www.w3.org, September 2012.
[104] Worldwide Large Hadron Collider Computing Grid. http://lcg.web.cern.ch/lcg/, September 2012.
[105] G. Aad, E. Abat, J. Abdallah, A. Abdelalim, A. Abdesselam, et al. The ATLAS Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3, 2008.
[106] K. Aamodt, A. Quintana, R. Achenbach, S. Acounis, et al. The ALICE Experiment at the CERN Large Hadron Collider. *Journal of Instrumentation*, 3, 2008.
[107] J. Alexander, D. Box, L. Cabrera, D. Chappell, G. Daniels, C. Kaler, D. Orchard, I. Sedukhin, M. Simek, and M. Theimer. *WS - Enumeration*. W3C Member Submission, 2006.
[108] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lőrentey, and F. Spataro. From Gridmap-file to Voms: Managing Authorization in a Grid Environment. *Future Generation Computer Systems*, 21(4):549–558.
[109] W. Allcock. *GridFTP: Protocol Extensions to FTP for the Grid*. Open Grid Forum, Grid Final Document Nr. 20, 2006.
[110] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guizar, N. Kartha, C. Liu, R. Khalaf, D. Koenig, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu. *Web Services Business Process and Execution Languge Version 2.0*. Organization for the Advancement of Structured Information Standards, 2007.
[111] A. Alves, A. Filho, A. Barbosa, I. Bediaga, G. Cernicchiaro, et al. The LHCb Detector at the LHC. *Journal of Instrumentation*, 3, 2008.
[112] D. Anderson. BOINC: A System for Public-Resource Computing and Storage. In *Proceedings 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004), Pittsburgh, USA*, pages 4–10, 2004.
[113] S. Andreozzi, S. Burke, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, and J. Navarro. *GLUE Specification Version 2.0*. Open Grid Forum, Grid Final Document Nr. 147, 2009.
[114] A. Andrieux, K. Czaijkowski, A. Dan, K. Keahey, H. Ludwig, T. Kakata, J. Pruyne, J. Rofrano, S. Tuecke, and M. Xu. *Web Services Agreement Specification*. Open Grid Forum, Grid Final Document Nr. 107, 2007.
[115] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. *Job Submission Description Language (JSDL) Specification Version 1.0*. Open Grid Forum, Grid Final Document Nr. 136, 2008.
[116] A. Anjomshoaa, M. Drescher, D. Fellows, S. McGougha, D. Pulsipher, and A. Savva. *Job Submission Description Language (JSDL) - Specification Version 1.0*. Open Grid Forum, Grid Final Document Nr. 56 (Deprecated, new Document Nr. 136, 2008), 2006.
[117] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. C. Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead. The Design and Implementation of Grid Database Services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17:357–376, 2005.
[118] M. Antonioletti, B. Collins, A. Krause, S. Laws, J. Magowan, S. Malaika, and N. Paton. *Web Services Data Access and Integration - The Relational Realisation (WS-DAIR) Specification, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 76, 2006.
[119] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski. Toward a Common Component Architecture for High-Performance Scientific Computing. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing (HPDC), Redondo Beach, USA*, pages 13–20, 1999.
[120] I. Avery. *Polish Grid Infrastructure PL-Grid*. Cel Publishing, ISBN 6200697256, 2012.
[121] R. Aymar, V. Chuyanov, M. huguet, and Y. Shimomura. Overview of ITER-FEAT - The Future International Burning Plasma Experiment. *Journal of Nuclear Fusion*, 41(1301), 2001.
[122] K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C. Liu, M. Nottingham, and P. Yendluri. Basic Profile Version 1.1. In *Web Service Interoperability (WS-I) Organization Final Document*, 2004.
[123] T. Banks. *Web Services Resource Framework Primer*. Organization for the Advancement of Structured Information Standards, 2003.
[124] R. Barbera, M. Fargetta, and E. Giorgio. Multiple Middleware Co-Existence: Another Aspect of Grid Interoperability. In *Proceedings of the IGIIW Workshop, Third IEEE International Conference on eScience, Bangalore, India*, pages 577–583, 2007.
[125] A. Barbir, M. Gudgin, M. McIntosh, and K. Morrison. Basic Security Profile Version 1.0. In *Web Service Interoperability (WS-I) Organization Final Document*, 2005.
[126] W. Barth. *NAGIOS: System and Network Monitoring*. Open Source Press GmbH, ISBN 1-59327-179-4, 2008.

[127] M. Beckerle, M. Buddhikot, D. Chatterjee, A. Clark, T. Coulter, M. Enescu, B. Goyal, J. Hammersley, M. Kataoka, R. Kumar, D. Pearson, P. Peiravi, S. Reddy, T. Rodriguez, J. Saiyed, R. Schibler, S. Schleimer, M. Schmitz, R. Sheen, H. Skardal, B. Souder, P. Strong, K. Sudo, B. Thome, and V. Viswanathan. *EGA - Reference Model and Use Cases Version 1.5 - Part 1 of 2*. Enterprise Grid Alliance Draft Document, 2006.
[128] M. Beckerle, M. Buddhikot, D. Chatterjee, A. Clark, T. Coulter, M. Enescu, B. Goyal, J. Hammersley, M. Kataoka, R. Kumar, D. Pearson, P. Peiravi, S. Reddy, T. Rodriguez, J. Saiyed, R. Schibler, S. Schleimer, M. Schmitz, R. Sheen, H. Skardal, B. Souder, P. Strong, K. Sudo, B. Thome, and V. Viswanathan. *EGA - Reference Model and Use Cases Version 1.5 - Part 2 of 2*. Enterprise Grid Alliance Draft Document, 2006.
[129] M. Beisiegel, H. Blohm, D. Booz, M. Edwards, O. Hurley, S. Ielceanu, A. Miller, A. Karmarkar, A. Malhotra, J. Marino, M. Nally, E. Newcomer, S. Patil, G. Pavlik, M. Raepple, M. Rowley, K. Tam, S. Vorthmann, P. Walker, and L. Waterman. *Service Component Architecture - Assembly Model Specification*. Organization for the Advancement of Structured Information Standards, 2007.
[130] M. Beisiegel, D. Booz, C. Chao, M. Edwards, S. Ielceanu, A. Karmarkar, A. Malhotra, E. Newcomer, S. Patil, M. Rowley, C. Sharp, and U. Yalcinalp. *Service Component Architecture - Policy Framework*. Organization for the Advancement of Structured Information Standards, 2007.
[131] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni, and S. Sundarrajan. *Distributed Systems Security - Issues, Processes and Solutions*. Wiley, ISBN 978-0-470-51988-2, 2009.
[132] BELIEF Project and European Commission. *5th e-Infrastructure Concertation Meeting 2008 Report*, 2009.
[133] BELIEF Project and European Commission. *6th e-Infrastructure Concertation Meeting 2008 Report - Looking to the Future: Sustainable e-Infrastructures*, 2009.
[134] BELIEF Project and European Commission. *7th e-Infrastructure Concertation Meeting 2009 Report*, 2009.
[135] L.-M. Birkholtz, O. Bastien, G. Wells, D. Grando, F. Joubert, V. Kasam, M. Zimmermann, P. Ortet, N. Jacq, N. Saidani, S. Roy, M. Hofmann-Apitius, V. Breton, A. I. Louw, and E. Marechal. Integration and Mining of Malaria Molecular, Functional and Pharmacological Data: How Far are we from a Chemogenomic Knowledge Space? *Malaria Journal, 5:110*, 2006.
[136] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. *Web Services Architecture*. W3C Note, 2004.
[137] D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, M. Hadley, C. Kaler, D. Langworthy, F. Leymann, B. Lovering, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, E. Sindambiwe, T. Storey, S. Weerawarana, and S. Winkler. *Web Services Addressing*. W3C Member Submission, 2004.
[138] C. Bratosin, W. Aalst, N. Sidorova, and N. Trcka. A Reference Model for Grid Architectures and its Analysis. In *Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico*, pages 898–913, 2008.
[139] J. Brooke, T. Eickermann, W. Frings, P. Gibbon, L. Kirtchakova, U. Lang, D. Mallmann, M. McKeown, S. Pickles, A. Porter, M. Riding, M. Romberg, A. Visser, and U. Woessner. Application Steering in a Collaborative Environment. In *Proceedings of the ACME/IEEE SC2003 Conference, Phoenix, USA*, pages 61–68, 2003.
[140] S. Burke, S. Andreozzi, and L. Field. Experiences with the GLUE Information Schema in the LCG/EGEE Production Grid. In *In Proceedings of the International Conference on Computing in High Energy and Nuclear Physics 2007 (CHEP 2007), Victoria, Canada*, 2007.
[141] S. Canada. *ISO Reference Model - Open Distributed Processing - Architecture*. ISO/IEC JTC1/SC7/WG19 HYD-015, Final Draft International Standard, 2010.
[142] S. Cantor, J. Kemp, R. Philpott, and E. Maler. *Assertions and Protocols for the OASIS Security Assertion Markup Language*. Organization for the Advancement of Structured Information Standards, 2005.
[143] A. Carbone, L. Agnello, A. Forti, A. Ghiselli, E. Lanciotti, L. Magnoni, M. Mazzucato, R. Santinelli, V. Sapunenko, V. Vagnoni, and R. Zappi. Performance Studies of the StoRM Storage Resource Manager. In *Proceedings of the Third IEEE International Conference on e-Science, Bangalore, India*, pages 423–430, 2007.
[144] M. Casalegno, G. Sello, and E. Benfenati. Top-Priority Fragment QSAR Approach in Predicting Pesticide Aquatic Toxicity. *Chemical Research in Toxicology*, 19(11):1533–1539, 2006.
[145] C. Catlett. *HPC and Grids in Action Amsterdam*, chapter TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications, pages 225–249. IOP Publishing Ltd., 2007.
[146] C. Catlett, C. de Laat, D. Martin, G. Newby, and D. Skow. *Open Grid Forum Document Process and Requirements*. Open Grid Forum, Grid Final Document Nr. 152, 2009.
[147] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *Parallel Programming in OpenMP*. Morgan Kaufmann, ISBN 1-55860-671-8, 2001.
[148] S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, et al. The CMS Experiment at the CERN LHC. *Journal of Instrumentation*, 3, 2008.
[149] V. Chopra, S. Li, and J. Genende. *Professional Apache Tomcat 6*. John Wiley and Sons, ISBN 0471753610, 2007.
[150] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. *Web Services Description Language - Version 1.1*. W3C Note, 2001.
[151] S. Covey. *The Seven Habits of Highly Effective People*. Free Press , ISBN 0-7432-6951-9, 1989.
[152] B. Demuth, B. Schuller, S. Holl, J. Daivandy, A. Giesler, V. Huber, and S. Sild. The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows. In *Proceedings of the 6th IEEE International Conference on e-Science 2010, Brisbane, Australia*, pages 238–245, 2010.
[153] T. Dierks and C. Allen. *The TLS Protocol Version 1.0*. Internet Engineering Task Force, RFC 2246, 1999.

[154] B. Dillaway, M. Humphrey, C. Smith, M. Theimer, and G. Wasson. *HPC Basic Profile Version 1.0*. Open Grid Forum, Grid Final Document Nr. 114, 2007.
[155] M. Drescher, A. Anjomshoaa, G. Williams, and D. Meredith. *JSDL - Parameter Sweep Job Extension*. Open Grid Forum, Grid Final Document Nr. 149, 2009.
[156] e-ScienceTalk and European Commission. *8th e-Infrastructure Concertation Meeting 2010 Report*, 2010.
[157] P. Eerola, T. Elof, M. Ellert, J. Hansen, A. Konstantinov, B. Konya, J. Nielsen, F. Ould-Saada, O. Smirnova, and A. Waananen. The NorduGrid Architecture and Tools. In *In Proceedings of the International Conference on Computing in High Energy and Nuclear Physics 2003 (CHEP 2003), La Jolla, USA*, 2003.
[158] EICTA. *EICTA White Paper on Standardisation and Interoperability*, 2006.
[159] J. Elgeti and G. Gompper. *NIC Symposium 2008*, volume 39 of *NIC series*, chapter Hydrodynamics of Active Mesoscopic Systems, pages 53–62. John von Neumann Institute for Computing, Juelich, ISBN 978-3-9810843-5-1, 2008.
[160] M. Ellert, M. Gronager, A. Konstantinov, B. Konya, J. Lindemann, I. Livenson, J. Nielsen, M. Niinimaeki, O. Smirnova, and A. Waeaenaenen. Advanced Resource Connector Middleware for Lightweight Computational Grids. *Future Generation Computer Systems*, 23(2):219–240, 2007.
[161] European e-Infrastructure Forum. *Euroepan E-Infrastructure Forum: ESFRI Requirements for Pan-European e-Infrastructure resources and facilities*, 2010.
[162] A. Ferrari, G. Degliesposti, M. Sgobba, and G. Rastelli. Validation of an Automated Procedure for the Prediction of Relative Free Energies of Binding on a set of Aldose Reductase Inhibitors. *Bioorg Med Chem*, 12(24), 2007.
[163] L. Field, S. Andreozzi, and B. Konya. Grid Information System Interoperability: The Need For A Common Information Model. In *Proceedings of the IGIIW Workshop, Fourth IEEE International Conference on eScience, Indianapolis, USA*, pages 501–507, 2008.
[164] L. Field and M. Schulz. Grid interoperability: the interoperation cookbook. In *In Proceedings of theInternational Conference on Computing in High Energy and Nuclear Physics 2007 (CHEP 2007, Victoria, Canada*, 2008.
[165] L. Field, M. Schulz, and E. Laure. Grid Deployment Experiences: Grid Interoperation. *Journal of Grid Computing: Special Issue on Grid Interoperability*, 7(3):287–296, 2009.
[166] R. Fisher and W. Ury. *Getting to Yes: Negotiating Agreement Without Giving In*. Houghton Mifflin, ISBN 0140157352, 1981.
[167] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Science. In *Proceedings of Sixth IFIP International Conference on Network and Parallel Computing, Beijing, China*, pages 213–223, 2005.
[168] I. Foster, D. Gannon, H. Kishimoto, and J. Reich. *Open Grid Services Architecture Use Cases*. Open Grid Forum, Grid Final Document Nr. 29, 2004.
[169] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer. *OGSA Basic Execution Service Version 1.0*. Open Grid Forum, Grid Final Document Nr. 108, 2007.
[170] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A Security Architecture for Computational Grids. In *5th ACM Conference on Computer and Communications Security (CCS5). San Francisco, California*, pages 83–91, 1998.
[171] I. Foster, C. Kesselmann, J. M. Nick, and S. Tuecke. *Grid Computing - Making the Global Infrastructure a Reality*, chapter The Physiology of the Grid, pages 217–249. John Wiley & Sons Ltd, ISBN 0470853190, 2003.
[172] I. Foster, C. Kesselmann, and S. Tuecke. *Grid Computing - Making the Global Infrastructure a Reality*, chapter The Anatomy of the Grid - Enable Scalable Virtual Organizations, pages 171–198. John Wiley & Sons Ltd, ISBN 0470853190, 2003.
[173] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Reich. *Open Grid Services Architecture, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 30, 2005.
[174] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Reich. *Open Grid Services Architecture, Version 1.5*. Open Grid Forum, Grid Final Document Nr. 80, 2006.
[175] I. Foster, T. Maguire, and D. Snelling. *OGSA WS-RF Basic Profile 1.0*. Open Grid Forum, Grid Final Document Nr. 72, 2006.
[176] I. Foster, Z. Yong, I. Raicu, and S. Lu. Cloud Computing and Grid Computing 360-Degree Compared. In *Proceedings of the Grid Computing Environments Workshop, Austin, USA*, pages 1–10, 2008.
[177] W. Frings, M. Riedel, A. Streit, D. Mallmann, S. van den Berghe, D. Snelling, and V. Li. LLview: User-Level Monitoring in Computational Grids and e-Science Infrastructures. In *Proceedings of German e-Science Conference, Baden-Baden, Germany, Online-publication*, 2007.
[178] P. Fuhrmann and V. Guelzow. dCache - Storage System for the Future. In *Proceedings of the Europar 2006, Dresden, Germany*, pages 1106–1113, 2006.
[179] F. Gagliardi, B. Jones, F. Grey, M.-E. Begin, and M. Heikkurinen. Building an Infrastructure for Scientific Grid Computing: Status and Goals of the EGEE project. *Philosophical Transactions of the Royal Society*, 363(15):1729–1742, 2005.
[180] D. Gannon, K. Chiu, M. Govindaraju, and A. Slominski. A Revised Analysis of the Open Grid Services Infrastructure. *Journal of Computing and Informatics*, 21:321–332, 2002.
[181] D. Gannon, K. Chiu, M. Govindaraju, and A. Slominski. *An Analysis of the Open Grid Services Architecture*. A Report to the UK e-Sciences Core Program - `http://www.extreme.indiana.edu/~aslom/papers/ogsa_analysis3.pdf`, 2002.

[182] H. Gardner and G. Manduchi. *Design Patterns for e-Science*. Springer, ISBN 978-3-540-68088-8, 2007.

[183] W. Gentzsch. Sun Grid Engine: Towards Creating a Compute Power Grid. In *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid 2001 (CCGRID 2001), Brisbane, Australia*, pages 35–36, 2001.

[184] W. Gentzsch, A. Kennedy, H. Lederer, G. Pringle, J. Reetz, M. Riedel, B. Schuller, A. Streit, and J. Wolfrat. DEISA: e-Science in a Collaborative, Secure, Interoperable and User-Friendly Environment. In *Proceedings of e-Challenges 2010 Conference, Warsaw, Poland*, pages 1–10, 2010.

[185] M. Gjiermundrod, M. Dikaiakos, M. Stuempert, P. Wolniewicz, and H. Kornmayer. g-Eclipse - An Integrated Framework to Access and Maintain Grid Resources. In *Proceedings 9th IEEE/ACM International Conference on Grid Computing (Grid 2008), Tsukuba, Japan*, pages 57–64, 2008.

[186] T. Goodale, S. Jha, H. Kaiser, T. Kielmann, P. Kleijer, A. Merzky, J. Shalf, and C. Smith. *A Simple API for Grid Applications (SAGA)*. Open Grid Forum, Grid Final Document Nr. 90, 2008.

[187] D. Goodsell, G. Morris, and A. Olson. Automated Docking of Flexible Ligands: Applications of AutoDock. *Journal of Molecular Recognition*, 9(1-5), 1996.

[188] S. Graham and J. Treadwell. *Web Services Resource Properties 1.2*. Organization for the Advancement of Structured Information Standards, 2006.

[189] M. Gronager, D. Johansson, J. Kleist, C. Sottrup, A. Waananen, L. Field, D. Qing, K. Happonen, and T. Linden. Interoperability between ARC and gLite - Understanding the Grid-job Life Cycle. In *Proceedings of the IGIIW Workshop, Fourth IEEE International Conference on eScience, Indianapolis, USA*, pages 493–500, 2008.

[190] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. Nielsen. *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation, 2003.

[191] F. Hedman, M. Riedel, P. Mucci, G. Netzer, A. Gholami, M. Memon, A. Memon, and A. Shah. Benchmarking of Integrated OGSA-BES with the Grid Middleware. In *Proceedings of the Europar 2008, Gran Canaria, Spain*, pages 113–122, 2008.

[192] E. Hinrichs, M. Hinrichs, and T. Zastrow. Weblicht: web-based LRT services for German. In *Proceedings of the 10th ACL System Demonstration, Association for Computational Linguistics, Stroudsburg, USA*, 2010.

[193] S. Holl, M. Riedel, B. Demuth, M. Romberg, A. Streit, and V. Kasam. Life Science Application Support in an Interoperable E-Science Environment. In *Proceedings of the 22nd IEEE International Symposium on Computer-Based Medical Systems (CBMS), Perth, Australia*, pages 1–8, 2009.

[194] G. Host and L. Laaksonen. *e-IRG White Paper 2011*. e-Infrastructure Reflection Group, 2011.

[195] R. Housley, W. Ford, W. Polk, and D. Solo. *Internet X.509 Public Key Infrastructure*. Internet Engineering Task Force, RFC 2459, 1999.

[196] T. Hows, M. Smith, and G. Good. *Understanding and Deploying LDAP Directory Services*. Pearson Education Inc., ISBN 0-672-32316-8, 2003.

[197] M. Humphrey, C. Smith, M. Theimer, and G. Wasson. *JSDL HPC Profile Application Extension, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 111, 2007.

[198] G. Huysmans. HELENA. In *Proceedings of Conference on Computational Physics (CP90)*, 1991.

[199] G. Huysmans. ILSA. *Phys. Plasmas*, 8(10), 2001.

[200] IEEE. *IEEE - The Authoritative Dictionary of IEEE Standard Terms, 7th Edition*. IEEE Standards Information Network, ISBN 0-7381-2601-2, 2000.

[201] T. Jensen, P. Millar, R. Mueller-Pfefferkorn, J. Nilsen, M. Zsolt, and R. Zappi. *Definition of a Storage Accounting Record*. EMI Project, `http://cdsweb.cern.ch/record/1352472`, 2011.

[202] S. Jha, H. Kaiser, A. Merzky, and O. Weidner. Grid Interoperability at the Application Level Using SAGA. In *Proceedings of the IGIIW Workshop, Third IEEE International Conference on eScience, Bangalore, India*, pages 584–591, 2007.

[203] I. JTC1. *ISO Reference Model - Open Distributed Processing - Overview*. ISO/IEC 10746-1, Final Draft International Standard, 1998.

[204] P. Kacsuk, Z. Farkas, and G. Fedak. Towards making BOINC and EGEE Interoperable. In *Proceedings of the IGIIW Workshop, Fourth IEEE International Conference on eScience, Indianapolis, USA*, pages 478–484, 2008.

[205] S. Kannan, M. Roberts, P. Mayes, D. Brelsford, and J. Skovira. *Workload Management with LoadLeveler*. IBM Redbook, ISBN 0738422096, 2001.

[206] F. Karagiannis. *e-IRG White Paper 2009*. e-Infrastructure Reflection Group, 2009.

[207] N. Karonis, B. Toonen, and I. Foster. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface. *Journal of Parallel and Distributed Computing*, 63(5):551–563, 2003.

[208] V. Kindratenko, J. Enos, S. Guochun, M. Showerman, G. Arnold, J. Stone, J. Philips, and H. Wen-mei. GPU clusters for high-performance computing. In *Proceedings of the IEEE Cluster Computing and Workshops (Cluster 2009), New Orleans, USA*, pages 1–8, 2009.

[209] B. Kryza, L. Skital, J. Kitowski, M. Li, and T. Itagaki. Analysis of Interoperability Issues between EGEE and VEGA Grid Infrastructures. In *Proceedings of High Performance Computing and Communications: Second International Conference (HPCC 2006), Munich, Germany*, pages 793–802, 2006.

[210] L. Laaksonen. *e-IRG Blue Paper 2010*. e-Infrastructure Reflection Group, 2010.

[211] J. Larson, B. Norris, E. Ong, D. Bernholdt, J. Drake, W. Elwasif, M. Ham, C. Rasmussen, G. Kufert, D. Katz, S. Zhou, C. DeLuca, and N. Collins. Components, the Common Component Architecture, and the Climate/Weather/Ocean Community. In *Proceedings of the 84th American Meteorological Society Annual Meeting, Seattle, USA*, 2004.

[212] E. Laure, S. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, F. Hemmer, A. D. Meglio, and A. Edlund. Programming the Grid with gLite. *Computational Methods in Science and Technology*, 12:33–46, 2006.

[213] W. Lee, A. McGough, and J. Darlington. Performance Evaluation of the GridSAM Job Submission and Monitoring System. pages 915–922, 2005. Proceedings of the UK 2005 All Hands Meeting.

[214] E. Lindahl, B. Hess, and D. van der Spoel. GROMACS 3.0: A Package for Molecular Simulation and Trajectory Analysis. *Journal of Molecular Modeling*, 7(8):306–317, 2001.

[215] B. Ludaescher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[216] R. Mach, R. Lepro-Metz, S. Jackson, and L. McGinnis. *Usage Record - Format Recommendation*. Open Grid Forum, Grid Final Document Nr. 98, 2007.

[217] C. MacKenzie, K. Laskey, F. McCabe, P. Brown, R. Metz, and B. Hamilton. *Reference Model for Service Oriented Architecture 1.0*. Organization for the Advancement of Structured Information Standards, 2006.

[218] T. Maguire and D. Snelling. *OGSA Profile Definition Version 1.0*. Open Grid Forum, Grid Final Document Nr. 59, 2006.

[219] S. Manos, S. Zasada, M. Mazzeo, R. Haines, G. Doctors, S. Brew, R. Pinning, J. Brooke, and P. Coveney. Patient Specific Whole Celebral Blood Flow Simulation: A Future Role in Surgical Treatment for Neurovascular Pathologies. In *Proceedings of the 3rd TeraGrid Conference, Virginia, USA*, 2008.

[220] M. Marzolla, M. Riedel, P. Andreetto, V. Venturi, A. Ferraro, A. Memon, M. Memon, B. Tweddell, D. Mallmann, A. Streit, S. van de Berghe, V. Li, D. Snelling, K. Stamou, Z. Shah, and F. Hedman. Open Standards-based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE. In *Proceedings of the IGIIW Workshop, Third IEEE International Conference on eScience, Bangalore, India*, pages 592–601, 2007.

[221] S. Matsuoka, S. Shinjo, M. Aoyagi, S. Sekiguchi, H. Usami, and K. Miura. Japanese Computational Grid Research Project: NAREGI. *Proceedings of the IEEE*, 93(3):522–533, 2005.

[222] A. McGough and A. Savva. *Implementation and Interoperability Experiences with the Job Submission Description Language (JSDL) 1.0*. Open Grid Forum, Grid Final Document Nr. 140, 2008.

[223] A. Memon, M. Memon, P. Wieder, and B. Schuller. CIS: An Information Service based on the Common Information Model. In *Proceedings of the Third IEEE International Conference on e-Science, Bangalore, India*, pages 465–472, 2007.

[224] M. Memon, A. Memon, M. Riedel, A. Streit, and F. Wolf. Enabling Grid Interoperability by Extending HPC-driven Job Management with an Open Standard Information Model. In *Proceedings of International Conference on Computer and Information Science (ICIS 2009), Shanghai, China*, pages 506–511, 2009.

[225] M. S. Memon, M. Riedel, A. Memon, F. Wolf, A. Streit, T. Lippert, M. Plociennik, M. Owsiak, D. Tskhakaya, and C. Konz. Lessons Learned from Jointly Using HTC- and HPC-driven e-Science Infrastructures in Fusion Science. In *Proceedings of 2nd IEEE International Conference on Information and Emerging Technologies (ICIET 2010), Karachi, Pakistan*, 2010.

[226] D. Merrill. *Secure Addressing Profile 1.0*. Open Grid Forum, Grid Final Document Nr. 131, 2008.

[227] D. Merrill. *Secure Communication Profile 1.0*. Open Grid Forum, Grid Final Document Nr. 132, 2008.

[228] H. Moore, K. Dvorakova, N. Jenkins, and W. Breed. Exceptional Sperm Cooperation in the Wood Mouse. *Letters to Nature*, 418(174):174–177, 2002.

[229] R. W. Moore, A. Rajasekar, and R. Marciano. *Proceedings iRODS User Group Meeting 2010 - Policy-based Data Management, Sharing, and Preservation*. CreateSpace, ISBN 1452813426, 2010.

[230] M. Morgan. *ByteIO Specification 1.0*. Open Grid Forum, Grid Final Document Nr. 87, 2007.

[231] M. Morgan and S. Grimshaw. Genesis II - Standards Based Grid Computing. In *Proceedings of the Seventh IEEE/ACM International Symposium on Cluster Computing and the Grid 2007 (CCGRID 2007), Rio de Janeiro, Brazil*, pages 611–618, 2007.

[232] R. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein. Federated Security: The Shibboleth Approach. *EDUCAUSE Quarterly*, 27(4):12–17, 2004.

[233] E. Morris, L. levine, C. Meyers, P. Place, and D. Plakosh. *Systems of Systems Interoperability (SOSI) - Final Report*. Carnegie Mellon University, Technical Report, CMU/SEI-2004-TR-004, 2004.

[234] T. Moses. *eXtensible Access Control Markup Language - Version 2.0 - Core Specification*. Organization for the Advancement of Structured Information Standards, 2005.

[235] M.Riedel and B. Demuth. UNICORE in XSEDE: Towards a Large-Scale Scientific Environment based on Open Standards. 9(2):18–19, 2011.

[236] M. Nelson, W. Humphrey, A. Gursoy, A. Dalke, L. Kale, R. Skeel, and K. Schulten. NAMD: A Parallel, Object-Oriented Molecular Dynamics Program. *International Journal of High Performance Computing Applications*, 10(4):251–268, 1996.

[237] B. Neuman and T. Tso. Kerberos: An authentication service for computer networks. In *IEEE Communications Magazine 32(9)*, pages 33–88, 1994.

[238] H. Neuroth, M. Kerzel, and W. Gentzsch. *D-Grid: German Grid Initiative*. Universitaetsverlag Goettingen, ISBN 978-3-938616-99-4, 2007.

[239] R. Nieuwpoort, T. Kielmann, and H. Bal. User-friendly and Reliable Grid Computing Based on Imperfect Middleware. In *Proceedings of the International Supercomputing Conference 2007, Reno, USA*, pages 1–11, 2007.

[240] J. Novotny, M. Russell, and O. Wehrens. GridSphere - An Advanced Portal Framework. pages 412–419, 2004. Proceedings of the 30th Euromicro Conference (EUROMICRO'04), Rennes, France.
[241] P. Pacheco. *Parallel Programming with MPI*. Morgan Kaufmann, ISBN 1558603395, 1996.
[242] D. Pearlman, D. Case, J. Caldwell, W. Ross, T. Cheatham, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. AMBER, a Package of Computer Programs for Applying Molecular Mechanics, Normal Mode Analysis, Molecular Dynamics and Free Energy Calculations to Simulate the Structural and Energetic Properties of Molecules. *Computer Physics Communications*, 91(1–3):1–41, 1995.
[243] M. Pereira, O. Tatebe, L. Luan, and T. Anderson. *Resource Namespace Service Specification*. Open Grid Forum, Grid Final Document Nr. 101, 2007.
[244] S. Pfalzner and P. Gibbon. *Many-Body Tree Methods in Physics*. Cambridge University Press, ISBN 0521019168, 1996.
[245] R. Piro, A. Guarise, and A. Werbrouck. An Economy-based Accounting Infrastructure for the DataGrid. In *Proceedings for the 4th International Workshop on Grid Computing, Phoenix, USA*, pages 202–204, 2003.
[246] D. Piscitello and A. Chapin. *Open Systems Networking: TCP/IP and OSI*. Addison Wesley, ISBN 0201563347, 1993.
[247] I. C. Plasencia, E. Fernandez, L. Cabellos, M. Plociennik, M. Owsiak, B. Guillerminet, and A. Soba. Modelling Mixed Workflows between Grid and HPC in EUFORIA. In *Proceedings of the 3rd Iberian Grid Infrastructure Conference, Valencia, Spain*, pages 256–265, 2009. ISBN 978-84-9745-406-3.
[248] R. Pordes, B. Kramer, M. Livny, P. Avery, K. Blackburn, T. Wenaus, K. Wuerthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, and R. Quick. The Open Science Grid. *Journal of Physics: Conference Series*, 78, 2007.
[249] G. L. Presti, O. Barring, A. Earl, R. Rioja, S. Ponce, G. Taurelli, D. Waldron, and M. Coelho. CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN. In *Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007), San Diego, USA*, pages 275–280, 2007.
[250] H. Rajic, R. Borbst, W. Chan, F. Fersti, J. Gardiner, A. Haas, B. Nitzberg, D. Templeton, J. Tollefsrud, and P. Troeger. *Distributes Resource Management Application API Specification 1.0*. Open Grid Forum, Grid Final Document Nr. 133, 2008.
[251] R. Ratering, M. Riedel, A. Lukichev, D. Mallmann, A. Vanni, C. Cacciari, S. Lanzarini, P. Bala, K. Benedyczak, M. Borcz, R. Kluszcynski, and G. Ohme. GridBeans: Supporting e-Science and Grid Applications. In *Proceedings of the Second IEEE International Conference on e-Science 2006, Amsterdam, Netherlands*, pages 45–51, 2006.
[252] J. Reich. *Open Grid Services Architecture Use Cases Tier 2*. Open Grid Forum, OGSA-WG Draft, 2004.
[253] W. Reisig. System Design using Petri Nets. In *Requirements Engineering, Arbeitstagung der GI, Friedrichshafen, Germany*, pages 29–41, 1983.
[254] M. Riedel. *Guide to e-Science: Next Generation Scientific Research and Discovery*, chapter E-Science Infrastructure Interoperability Guide - The Seven Steps towards Interoperability for e-Science. Computer Communications and Networks. Springer, ISBN 978-0-85729-438-8, 2011.
[255] M. Riedel, T. Eickermann, W. Frings, S. Dominiczak, T. Duessel, A. Streit, P. Gibbon, F. Wolf, W. Schiffmann, and T. Lippert. Design and Evaluation of a Collaborative Online Visualization and Steering Framework Implementation for Computational Grids. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007), Austin, USA*, pages 169–177, 2007.
[256] M. Riedel, E. Laure, T. Soddemann, L. Field, J. P. Navarro, J. Casey, M. Litmaath, J. P. Baud, B. Koblitz, C. Catlett, D. Skow, C. Zheng, P. M. Papadopoulos, M. Katz, N. Sharma, O. Smirnova, B. Konya, P. Arzberger, F. Wuerthwein, A. S. Rana, T. Martin, M. Wan, V. Welch, T. Rimovsky, S. Newhouse, A. Vanni, Y. Tanaka, Y. Tanimura, T. Ikegami, D. Abramson, C. Enticott, G. Jenkins, R. Pordes, S. Timm, G. Moont, M. Aggarwal, D. Colling, O. van der Aa, A. Sim, V. Natarajan, A. Shoshani, J. Gu, S. Chen, G. Galang, R. Zappi, L. Magnoni, V. Ciaschini, M. Pace, V. Venturi, M. Marzolla, P. Andreetto, B. Cowles, S. Wang, Y. Saeki, H. Sato, S. M. P. Uthayopas, S. Sriprayoonsakul, O. Koeroo, M. Viljoen, L. Pearlman, S. Pickles, D. Wallom, G. Moloney, J. Lauret, J. Marsteller, P. Sheldon, S. Pathak, S. D. Witt, J. Mencak, J. Jensen, M. Hodges, D. Ross, G. N. S. Phatanapherom, A. R. Gregersen, M. Jones, S. Chen, P. Kacsuk, A. Streit, D. M. F. Wolf, T. Lippert, T. Delaitre, E. Huedo, and N. Geddes. Interoperation of World-wide Production e-Science Infrastructures. In *Concurrency and Computation: Practice and Experience*, volume 21, pages 961–990, 2009.
[257] M. Riedel and D. Mallmann. Standardization Processes of the UNICORE Grid System. In *Proceedings of 1st Austrian Grid Symposium 2005, Schloss Hagenberg, Austria*, pages 191–203. Austrian Computer Society, 2005.
[258] M. Riedel, D. Mallmann, and A. Streit. Enhancing Scientific Workflows with Secure Shell Functionality in UNICORE Grids. In *Proceedings of the First IEEE International Conference on e-Science 2005, Melbourne, Australia*, pages 132–139, 2005.
[259] M. Riedel, A. Memon, M. Memon, D. Mallmann, A. Streit, F. Wolf, T. Lippert, V. Venturi, P. Andreetto, M. Marzolla, A. Ferraro, A. Ghiselli, F. Hedman, Z. A., J. Salzemann, A. DaCosta, V. Breton, V. Kasam, M. Hofmann-Apitius, D. Snelling, S. van de Berghe, V. Li, S. Brewer, A. Dunlop, and N. DeSilva. Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA. In *Proceedings of the IEEE 31rd International Convention MIPRO 2008, Opatija, Croatia*, pages 225–231, 2008.

[260] M. Riedel, M. Memon, A. Memon, D. Mallmann, T. Lippert, D. Kranzlmueller, and A. Streit. e-Science Infrastructure Integration Invariants to Enable HTC and HPC Interoperability Applications. In *Proceedings of the Eights High-Performance Grid Computing Workshop, International Parallel and Distributed Processing Symposium (IPDPS 2011), Anchorage, USA*, pages 922–931, 2011.

[261] M. Riedel, M. Memon, A. Memon, A. Streit, F. Wolf, T. Lippert, B. Konya, O. Smirnova, A. Konstantinov, L. Zangrando, M. Marzolla, J. Watzl, and D. Kranzlmueller. Improvements of Common Open Grid Standards to Increase High Throughput and High Performance Computing Effectiveness on Large-scale Grid and e-Science Infrastructures. In *Proceedings of the Seventh High-Performance Grid Computing Workshop, International Parallel and Distributed Processing Symposium (IPDPS 2010), Atlanta, USA*, pages 1–7, 2010.

[262] M. Riedel, R. Menday, A. Streit, and P. Bala. A DRMAA-based Target System Interface Framework for UNICORE. In *Proceedings of the 2nd International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems, 12th International Conference on Parallel and Distributed Systems (ICPADS 2006), Minneapolis, USA*, pages 133–138, 2006.

[263] M. Riedel, B. Schuller, M. Rambadt, M. Memon, A. Memon, A. Streit, F. Wolf, T. Lippert, S. Zasada, S. Manos, P. Coveney, F. Wolf, and D. Kranzlmueller. Exploring the Potential of Using Multiple e-Science Infrastructures with Emerging Open Standards-based e-Health Research Tools. In *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2010), Melbourne, Australia*, pages 341–348, 2010.

[264] M. Riedel, A. Streit, D. Kranzlmueller, D. Mallmann, and T. Lippert. Requirements of an e-Science Infrastructure Interoperability Reference Model. In *Proceedings of the IEEE 34th International Convention MIPRO 2011, Opatija, Croatia*, pages 221–226, 2011.

[265] M. Riedel, A. Streit, T. Lippert, F. Wolf, and D. Kranzlmueller. Concepts and Design of an Interoperability Reference Model for Scientific- and Grid Computing Infrastructures. In *Proceedings of Applied Computing Conference in Mathematical Methods and Applied Computing (ACC 2009), Athens, Greece*, pages 691–698, 2009.

[266] M. Riedel, A. Streit, T. Lippert, F. Wolf, and D. Kranzlmueller. Towards Individually Formed Computing Infrastructures with High Throughput and High Performance Computing Resources of Large-scale Grid and e-Science Infrastructures. In *Proceedings of the IEEE 33rd International Convention MIPRO 2010, Opatija, Croatia*, pages 192–197, 2010.

[267] M. Riedel, A. Streit, D. Mallmann, F. Wolf, and T. Lippert. Experiences and Requirements for Interoperability between HTC- and HPC-driven e-Science Infrastructures. In *Proceedings of the Korea e-Science All Hands Meeting, Workshop HPC for e-Science: Future Applications and Middleware Technology on e-Science, Daejeon, Korea, 2008*, pages 113–123, 2010.

[268] M. Riedel, A. Streit, F. Wolf, T. Lippert, and D. Kranzlmueler. Classification of Different Approaches for e-Science Applications in Next Generation Computing Infrastructures. In *Proceedings of the Fourth IEEE International Conference on eScience 2008, Indianapolis, USA*, pages 198–205, 2008.

[269] M. Riedel and G. Terstyanszky. Grid Interoperability for e-Research. *Journal of Grid Computing: Special Issue on Grid Interoperability*, 7(3):285–286, 2009.

[270] M. Riedel and J. Watzl. *OGF Production Grid Infrastructure: Use Case Collection, Version 1.0.* Open Grid Forum, Grid Final Document Nr. 180, 2011.

[271] M. Riedel, P. Wittenburg, J. Reetz, M. van de Sanden, J. Rybicki, B. von St. Vieth, G. Fiameni, G. Mariani, A. Michelini, C. Cacciari, W. Elbers, D. Broeder, R. Verkerk, E. Erastova, M. Lautenschlaeger, R. Budig, H. Thielmann, P. Coveney, S. Zasada, O. Buechner, C. Manzano, M. Memon, A. Memon, D. Lecarpentier, H. Helin, K. Koski, and T. Lippert. A Data Infrastructure reference Model with Applications: Towards Realization of a ScienceTube Vision with a Data Replication Service. *Journal of Internet Services and Applications*, 4(1), 2013.

[272] M. Riedel, F. Wolf, D. Kranzlmueller, A. Streit, and T. Lippert. Research Advances by using Interoperable e-Science Infrastructures - The Infrastructure Interoperability Reference Model applied in e-Science. *Journal of Cluster Computing, Special Issue on Recent Advances in e-Science*, 12(4):357–372, 2009.

[273] T. Rings, G. Caryer, J. Gallop, J. Grabowski, T. Kovacikova, S. Schulz, and I. Stokes-Rees. Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network. *Journal of Grid Computing: Special Issue on Grid Interoperability*, 7(3):375–393, 2009.

[274] C. Rizzuto. *European Roadmap for Research Infrastructures*. European Commission, ISBN 978-92-79-10117-5, 2008.

[275] A. Robbins. *UNIX in a Nutshell*. O'Reilly Media, ISBN 0596100299, 2005.

[276] M. Rosen. *Applied SOA: Service-oriented Architecture and Design Strategies*. John Wiley and Sons, ISBN 0470223650, 2008.

[277] H. Rosmanith and J. Volkert. Traffic Forwarding with GSH/GLOGIN. In *Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP), Lugano, Switzerland*, pages 213–219, 2005.

[278] M. Russell, P. Dziubecki, P. Grabowski, M. Krysinski, T. Kuczynski, D. Szjenfeld, D. Tarnawczyk, G. Wolniewicz, and J. Nabrzyski. The Vine Toolkit: A Java Framework for Developing Grid Applications. In *Proceedings of the Seventh International Conference on Parallel Processing and Applied Mathematics (PPAM 2007), Gdansk, Poland*.

[279] T. Sandholm, P. Gardfjaell, E. Elmroth, O. Mulmo, and L. Johnsson. A Service-Oriented Approach to Enforce Grid Resource Allocation. *International Journal of Cooperative Information Systems, Vol.15*, 15:439–459, 2006.
[280] N. Santos and B. Koblitz. Security in Distributed Metadata Catalogue. *Concurrency and Computation: Practice and Experience*, 20:1995–2007, 2008.
[281] A. Savva. *JSDL SPMD Application Extension, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 115, 2007.
[282] J. Schuhmacher. *NIC Symposium 2008*, volume 39 of *NIC series*, chapter The Fine-Scale Structure of Turbulence. John von Neumann Institute for Computing, Juelich, ISBN 978-3-9810843-5-1, 2008.
[283] B. Schuller, M. Riedel, S. Memon, A. Memon, B. Konya, O. Smirnova, A. Konstantinov, S. Andersen, L. Zangrando, M. Sgaravatto, and E. Frizziero. *EMI Execution Service Specifation*. EMI Project, `https://twiki.cern.ch/twiki/pub/EMI/EmiExecutionService/EMI-ES-Specification_v1.0.odt`, 2010.
[284] B. Schuller, M. Riedel, and A. Streit. Recent Advances in the UNICORE 6 Middleware. *inSiDE*, 8:46–49, 2010.
[285] M. Siddiqui, A. Villazon, R. Prodan, and T. Fahringer. Advanced Reservation and Co-Allocation of Grid Resources: A Step towards an invisible Grid. In *Proceedings of the 9th IEEE Int. Multitopic Conference INMIC, Karachi, Pakistan*, pages 1–6, 2005.
[286] A. Sim and A. Shoshani. *The Storage Resource Manager Interface Specification - Version 2.2*. Open Grid Forum, Grid Final Document Nr. 129, 2008.
[287] C. Smith, T. Kielmann, S. Newhouse, and M. Humphrey. The HPC Basic Profile and SAGA: Standardizing Compute Grid Access in the Open Grid Forum. *Concurrency and Computation: Practice and Experience*, 21(8):1053–1068, 2009.
[288] D. Snelling, D. Merrill, and A. Savva. *OGSA Basic Security Profile 2.0*. Open Grid Forum, Grid Final Document Nr. 138, 2008.
[289] B. Snyder, D. Bosanac, and R. Davies. *ActiveMQ in Action*. Manning, ISBN 1933988940, 2011.
[290] C. Sosa and B. Knudson. *IBM Blue Gene Solution: Blue Gene/P Application Development*. IBM Redbook, ISBN 0738433330, 2009.
[291] I. Stokes-Rees. *A REST Model for High Throughput Scheduling in Computational Grids*. PhD thesis, Oxford University, 2006.
[292] D. Stone, S. Marcio, N. Miranda, and F. Costa. A Model for Transparent Grid Interoperability. In *Proceedings of the Seventh IEEE/ACM International Symposium on Cluster Computing and the Grid 2007 (CCGRID 2007), Rio de Janeiro, Brazil*, 2007.
[293] A. Streit, P. Bala, A. Beck-Ratzka, K. Benedyczak, S. Bergmann, R. Breu, J. Daivandy, B. Demuth, A. Eifer, A. Giesler, B. Hagemeier, S. Holl, V. Huber, D. Mallmann, A. Memon, M. Memon, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, T. Schlauch, A. Schreiber, T. Soddemann, and W. Ziegler. UNICORE 6 - Recent and Future Advancements. *Annals of Telecommunication*, 65(11):757–762, 2010.
[294] A. Streit, D. Erwin, T. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, and P. Wieder. UNICORE - From Project Results to Production Grids. *Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing*, 14:357–376, 2005.
[295] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, ISBN 9780130661029, 2002.
[296] A. S. Tanenbaum and M. van Steen. *Distributed Systems - Principles and Paradigms*. Prentice Hall International, ISBN 0132392275, 2006.
[297] J. Treadwell. *Open Grid Services Architecture Glossary of Terms, Version 1.5*, 2006.
[298] A. Trefethen, V. Menon, C. Chang, G. Czajowski, C. Meyers, and L. Trefethen. MultiMATLAB: MATLAB on Multiple Processors. In *Technical Report CTC96TR293*. Cornell Theory Center, 1996.
[299] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. *Open Grid Services Infrastructure, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 15, 2003.
[300] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile*. Internet Engineering Task Force, RFC 3820, 2004.
[301] S. Vaerttoe. *DEISA - Advancing Science in Europe*. DEISA, ISBN 978-952-5520-32-3, 2008.
[302] V. Venturi, M. Riedel, A. Memon, M. Memon, F. Stagni, B. Schuller, D. Mallmann, B. Tweddell, A. Gianoli, V. Ciaschini, S. van de Berghe, D. Snelling, and A. Streit. Using SAML-based VOMS for Authorization within Web Services-based UNICORE Grids. In *Proceedings of the 3rd UNICORE Summit 2007, 2007 Conference on Parallel Processing (Euro-Par 2007), Rennes, France*, volume LNCS4854, pages 112–120, 2008.
[303] V. Venturi, F. Stagni, A. Gianoli, A.Ceccanti, and V. Ciaschini. Virtual Organization Management Across Middleware Boundaries. In *Proceedings of the IGIIW Workshop, Third IEEE International Conference on eScience, Bangalore, India*, pages 545–552, 2007.
[304] Y. Wang, D. Roberto, M. Boniface, D. Qian, D. Cui, and J. Jiang. Cross-Domain Middlewares Interoperability for Distributed Aircraft Design Optimization. In *Proceedings of the IGIIW Workshop, Fourth IEEE International Conference on eScience, Indianapolis, USA*, pages 485–492, 2008.
[305] G. Wasson and M. Humphrey. *HPC File Staging Profile, Version 1.0*. Open Grid Forum, Grid Final Document Nr. 135, 2008.
[306] P. Wittenburg, N. Bel, L. Borin, G. Budin, N. Calzolari, E. Hajicova, K. Koskenniemi, L. Lemnitzer, B. Maegaard, M. Piasecki, J. Pierrel, S. Piperidis, I. Skadina, D. Tufis, R. Veenendaal, T. Varadi, and M. Wynne. Resource and Service Centres as the Backbone for a Sustainable Service Infrastructure. In *Proceedings of 7th International Conference on Language Resources and Evaluation (LREC) 2010 Conference, Valetta, Malta*, 2010.

[307] Z. XU, W. Li, L. Zha, H. Yu, and D. Liu. VEGA: A Computer Systems Approach to Grid Computing. *Journal of Grid Computing*, 2(2):109–120, 2004.

[308] S. Zasada and P. Coveney. Virtualizing access to scientific applications with the Application Hosting Environment. *Computer Physics Communications*, 180(12):2513–2525, 2009.

1. **Three-dimensional modelling of soil-plant interactions: Consistent coupling of soil and plant root systems**
   by T. Schröder (2009), VIII, 72 pages
   ISBN: 978-3-89336-576-0
   URN: urn:nbn:de:0001-00505

2. **Large-Scale Simulations of Error-Prone Quantum Computation Devices**
   by D. B. Trieu (2009), VI, 173 pages
   ISBN: 978-3-89336-601-9
   URN: urn:nbn:de:0001-00552

3. **NIC Symposium 2010**
   Proceedings, 24 – 25 February 2010 | Jülich, Germany
   edited by G. Münster, D. Wolf, M. Kremer (2010), V, 395 pages
   ISBN: 978-3-89336-606-4
   URN: urn:nbn:de:0001-2010020108

4. **Timestamp Synchronization of Concurrent Events**
   by D. Becker (2010), XVIII, 116 pages
   ISBN: 978-3-89336-625-5
   URN: urn:nbn:de:0001-2010051916

5. **UNICORE Summit 2010**
   Proceedings, 18 – 19 May 2010 | Jülich, Germany
   edited by A. Streit, M. Romberg, D. Mallmann (2010), iv, 123 pages
   ISBN: 978-3-89336-661-3
   URN: urn:nbn:de:0001-2010082304

6. **Fast Methods for Long-Range Interactions in Complex Systems**
   Lecture Notes, Summer School, 6 – 10 September 2010, Jülich, Germany
   edited by P. Gibbon, T. Lippert, G. Sutmann (2011), ii, 167 pages
   ISBN: 978-3-89336-714-6
   URN: urn:nbn:de:0001-2011051907

7. **Generalized Algebraic Kernels and Multipole Expansions for Massively Parallel Vortex Particle Methods**
   by R. Speck (2011), iv, 125 pages
   ISBN: 978-3-89336-733-7
   URN: urn:nbn:de:0001-2011083003

8. **From Computational Biophysics to Systems Biology (CBSB11)**
   Proceedings, 20 - 22 July 2011 | Jülich, Germany
   edited by P. Carloni, U. H. E. Hansmann, T. Lippert, J. H. Meinke, S. Mohanty, W. Nadler, O. Zimmermann (2011), v, 255 pages
   ISBN: 978-3-89336-748-1
   URN: urn:nbn:de:0001-2011112819

9. **UNICORE Summit 2011**
Proceedings, 7 - 8 July 2011 | Toruń, Poland
edited by M. Romberg, P. Bała, R. Müller-Pfefferkorn, D. Mallmann (2011), iv, 150 pages
ISBN: 978-3-89336-750-4
URN: urn:nbn:de:0001-2011120103

10. **Hierarchical Methods for Dynamics in Complex Molecular Systems**
Lecture Notes, IAS Winter School, 5 – 9 March 2012, Jülich, Germany
edited by J. Grotendorst, G. Sutmann, G. Gompper, D. Marx (2012), vi, 540 pages
ISBN: 978-3-89336-768-9
URN: urn:nbn:de:0001-2012020208

11. **Periodic Boundary Conditions and the Error-Controlled Fast Multipole Method**
by I. Kabadshow (2012), v, 126 pages
ISBN: 978-3-89336-770-2
URN: urn:nbn:de:0001-2012020810

12. **Capturing Parallel Performance Dynamics**
by Z. P. Szebenyi (2012), xxi, 192 pages
ISBN: 978-3-89336-798-6
URN: urn:nbn:de:0001-2012062204

13. **Validated force-based modeling of pedestrian dynamics**
by M. Chraibi (2012), xiv, 112 pages
ISBN: 978-3-89336-799-3
URN: urn:nbn:de:0001-2012062608

14. **Pedestrian fundamental diagrams: Comparative analysis of experiments in different geometries**
by J. Zhang (2012), xiii, 103 pages
ISBN: 978-3-89336-825-9
URN: urn:nbn:de:0001-2012102405

15. **UNICORE Summit 2012**
Proceedings, 30 - 31 May 2012 | Dresden, Germany
edited by V. Huber, R. Müller-Pfefferkorn, M. Romberg (2012), iv, 143 pages
ISBN: 978-3-89336-829-7
URN: urn:nbn:de:0001-2012111202

16. **Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures**
by M. Riedel (2013), x, 270 pages
ISBN: 978-3-89336-861-7
URN: urn:nbn:de:0001-2013031903

Computational simulations and thus scientific computing is the third pillar alongside theory and experiment in todays science. The term e-Science evolved as a new research field that focuses on collaboration in key areas of science using next generation data and computing infrastructures (i.e. e-Science infrastructures) to extend the potential of scientific computing. During the past decade, significant international and broader interdisciplinary research is increasingly carried out by global collaborations that often share resources within a single production e-Science infrastructure. More recently, increasing complexity of e-Science applications that embrace multiple physical models (i.e. multi-physics) and consider a larger range of scales (i.e. multi-scale) is creating a steadily growing demand for world-wide interoperable infrastructures that allow for new innovative types of e-Science by jointly using different kinds of e-Science infrastructures. But interoperable e-Science infrastructures are still not seamlessly provided today and this thesis argues that this is due to the absence of a production-oriented e-Science infrastructure reference model. The goal of this thesis is thus to present an infrastructure interoperability reference model (IIRM) design tailored to production needs and that represents a trimmed down version of the Open Grid Service Architecture (OGSA) in terms of functionality and complexity, while on the other hand being more specifically useful for production and thus easier to implement. This reference model is underpinned with lessons learned and numerous experiences gained from production e-Science application needs through accompanying academic case studies of the bio-informatics, e-Health, and fusion domain that all seek to achieve research advances by using interoperable e-Science infrastructures on a daily basis. Complementary to this model, a seven segment-based process towards sustained infrastructure interoperability addresses important related issues like harmonized operations, cooperation, standardization as well as common policies and joint development roadmaps.

This publication was edited at the Jülich Supercomputing Centre (JSC) which is an integral part of the Institute for Advanced Simulation (IAS). The IAS combines the Jülich simulation sciences and the supercomputer facility in one organizational unit. It includes those parts of the scientific institutes at Forschungszentrum Jülich which use simulation on supercomputers as their main research methodology.

IAS Series
Volume 16
ISBN 978-3-89336-861-7

JÜLICH FORSCHUNGSZENTRUM

JÜLICH

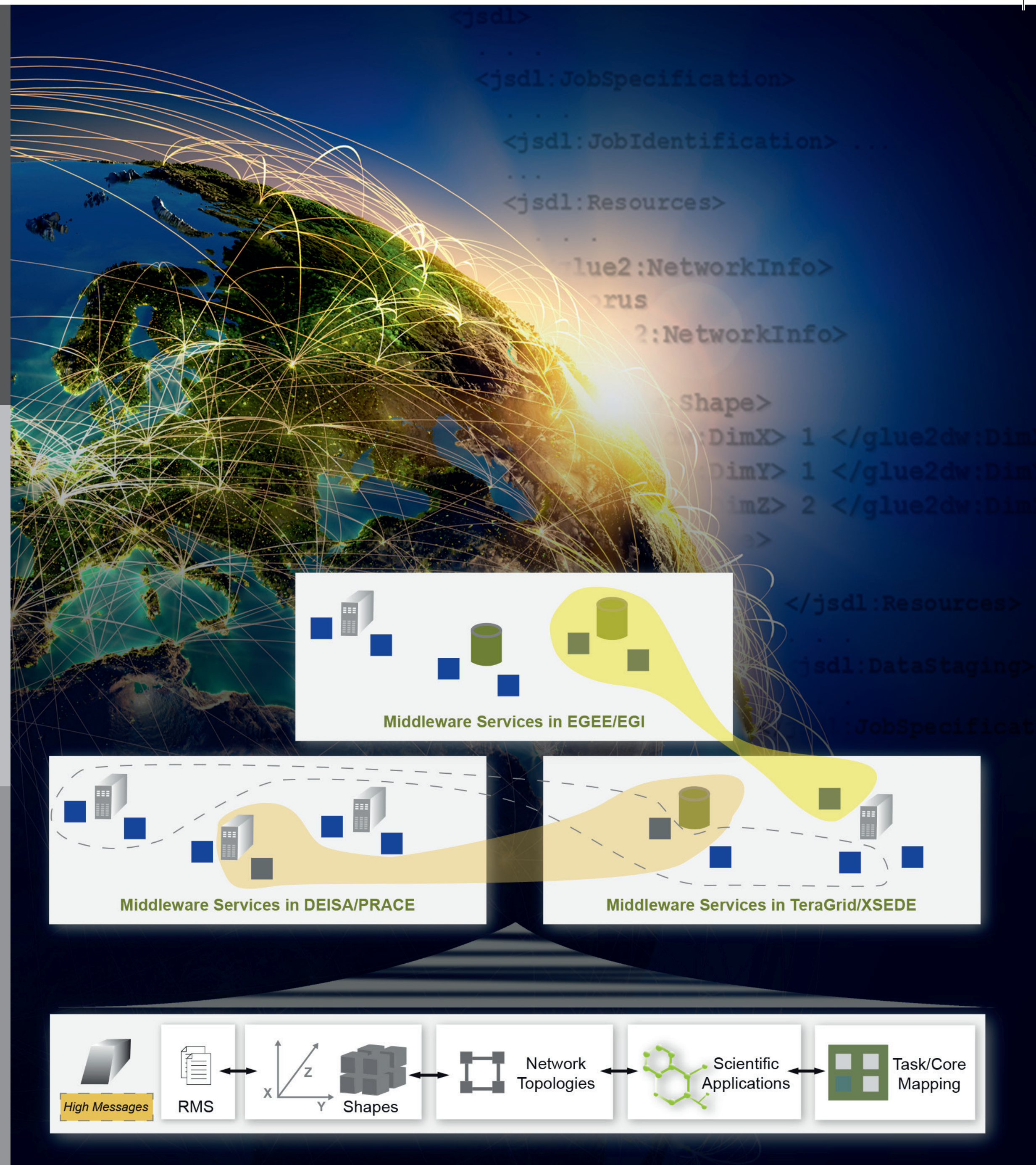
e-Science Infrastructure Design and Applications

Morris Riedel

IAS Series

16

Mitglied der Helmholtz-Gemeinschaft



Design and Applications of an Interoperability Reference Model for Production e-Science Infrastructures

Morris Riedel

JÜLICH FORSCHUNGSZENTRUM