Forschungssoftware: Chancen und Herausforderungen‡

Jens Saak [1], **Robert Speck** [2], **Jan Philipp Thiele** [3]

[1]Max Planck Institut für Dynamik komplexer technischer Systeme Madgeburg; [2]Forschungszentrum Jülich; [3]TU Braunschweig

‡ Zweitveröffentlichung: Dieser Artikel ist zuerst im Rundbrief 01/2026 der Gesellschaft für Angewandte Mathematik und Mechanik, GAMM e.V. erschienen.

Lizenz: Dieses Werk ist lizenziert unter einer „CC BY 4.0“ Lizenz.

Einleitung

Schnell durchrechnen, ob wir die gewünschte Konvergenz sehen? Kurz den Verlauf der Funktion anschauen? Passt das, was wir gerade aufs Papier gebracht haben? Mit modernen, mathematisch orientierten Programmiersprachen ist so etwas schnell erledigt. Der zugehörige Code sieht dann oft so aus, wie in Listing 1 gezeigt. Was aber, wenn man damit weiterarbeiten will? Oder in ein paar Wochen nochmals auf den Code schauen muss? Vielleicht wird die Erkenntnis aus diesem Schnipsel gar die Basis für eine Veröffentlichung? Ganz schnell entsteht selbst aus wenigen Zeilen Code die Basis für tiefergehende wissenschaftliche Untersuchungen.

```
clear all; close all; clc

r=2.7;
K=10;
y0=0.5;

f=@(t,y)r*y*(1-y/K);

[t,y]=ode45(f,[0,20],y0);

plot(y) %plot
title('ODE45 result')
```

Listing 1. Klar soweit?

Spätestens dann müssen diese Zeilen Code belastbar sein: wir brauchen eine geeignete Dokumentation, um anderen aber auch unserem „Future Self“ deutlich zu machen, was genau hier passiert. Wir brauchen Tests, um sicherzustellen, dass der Code wirklich das tut, was wir erwarten. Eine Lizenz, um den Code zur Verfügung zu stellen, und verbreitete Standards bei der Behandlung von Daten und beim Zusammenspiel mit anderer Software. All dies wird dann wichtig, wenn aus der kurzen Analyse einer mathematischen Fragestellung mehr wird, geplant oder ungeplant.

Für einen großen Teil der Forschung, insbesondere natürlich im Bereich der computergestützten Mathematik und der Mechanik, spielt Forschungssoftware mittlerweile eine kaum zu überschätzende Rolle. Dies ist jedoch nicht nur in diesen Bereichen und nicht nur in Deutschland der Fall.

Zwei Umfragen aus den Jahren 2014 und 2017 belegen die Bedeutung von Forschungssoftware für die Wissenschaft: In Großbritannien [1] bestätigten 92 Prozent der teilnehmenden Forschenden, dass sie Forschungssoftware verwenden, und 69 Prozent gaben an, dass diese für ihre Forschung von grundlegender Bedeutung ist. Die Umfrage aus den USA [2] ergab, dass sogar 95 Prozent der Teilnehmenden Forschungssoftware nutzen, und für 66 Prozent der Befragten war sie

für ihre Forschung von grundlegender Bedeutung.

Software ist für die Forschung also von entscheidender Bedeutung. Ihre Entwicklung sollte daher als Pfeiler der Wissenschaft und insbesondere als eigenständige Forschungsleistung betrachtet werden. So ist die Anerkennung von Forschungssoftware und der Arbeit an dieser von zentraler Bedeutung – eine Einsicht, die sich im Motto „Better Software, Better Research" widerspiegelt.

Definitionen

Was aber ist eigentlich Forschungssoftware? Aus der UK Research Software Survey 2014 [3] kommt die folgende Definition: „Research software can be anything from a few lines of code written by yourself, to a professionally developed software package.". In [4] wurde dies präzisiert durch eine inklusive und eine exklusive Definition: während erstere im Wesentlichen alle Software als Forschungssoftware interpretiert, die im Rahmen des Forschungsprozesses auf die eine oder andere Weise relevant ist, stellt die exklusive, strikte Definition die Forschungsintention in den Vordergrund. Dieser Definition folgend ist nur solche Software als Forschungssoftware zu sehen, die mit der Intention entwickelt wurde, Teil des Forschungsprozesses zu sein, möglicherweise spezialisiertes Domänenwissen erfordert und selbst einen Beitrag zu Wissenschaft und Forschung leistet. Zudem fällt unter diese Definition auch Software, die selbst Gegenstand von Forschung ist. Microsoft Excel, auch wenn es immer noch gerne und oft für das Sammeln und Visualisieren von Daten genutzt wird, ist somit keine Forschungssoftware. Python selbst ist ebenfalls keine Forschungssoftware, ebenso wenig wie ein Betriebssystem (sofern diese nicht Ziel von Forschung selbst sind) obwohl beide oft im Forschungsprozess eine wesentliche Rolle spielen. Diese Diskussion zeigt aber auch, dass die Einordnung und Kategorisierung von Forschungssoftware [5] stark vom Betrachtenden abhängt. Aufgrund dieser notwendigen Abgrenzung folgen wir hier der exklusiven, strikten Definition von Forschungssoftware.

Auch abseits solcher Definitionen gibt es wesentliche Unterschiede zwischen allgemeiner Software und Forschungssoftware. Ganz prägnant: Wir wissen meist nicht, was das Resultat unserer Forschungssoftware sein wird, denn um dies herauszufinden, schreiben wir ja gerade diese Software [6]. Wir kennen nicht die Kräfte, die auf einen Knochen unter Last wirken. Wir kennen nicht die Rate, mit der geladene Teilchen die Wand eines Reaktors malträtieren. Wir wissen nicht, wie die Lösung unseres Newton-Verfahrens aussieht. Dies ist bei klassischer, industrieller Software im Allgemeinen anders: hier ist das zu erwartende Endergebnis klarer definiert und meistens auch testbar.

Weniger prägnant, aber mindestens genauso wichtig: industriell eingesetzte Software wird primär von professionellen Software Engineers geschrieben, erweitert und gewartet. Forschungssoftware, auf der anderen Seite, wird in den meisten Fällen von Domänenwissenschaftler:innen geschrieben, die üblicherweise keine Ausbildung auf dem Gebiet der Softwareentwicklung haben. Hierbei ist das Spektrum groß: von der Wissenschaftlerin mit starker Affinität zur Softwareentwicklung bis hin zum Wissenschaftler ohne tiefere Vorkenntnisse. Sie alle schreiben Forschungssoftware, sie alle verlassen sich auf die Ergebnisse. Die erste Gruppe wird hierbei oft Research Software Engineers (RSEs) genannt, die zweite eher „Scientists who code", um die Präferenzen deutlicher zu machen. Diese Unterscheidung und insbesondere die Abgrenzung zum Software Engineer aus der Informatik ist ganz wesentlich in der Kommunikation, in der Art und Weise, wie Unterstützung angeboten und angenommen wird und schließlich in der Schaffung einer Community rund um das Thema Forschungssoftware und Forschungssoftwareentwicklung.

Gemeinsames

Der Begriff *Research Software Engineering* (ebenfalls RSE oder auch neuerdings RSEng abgekürzt, der Kontext lässt jedoch meist keine Verwirrung entstehen) entstand Anfang der 2010er Jahre im Kontext der zunehmenden Bedeutung von Software für die wissenschaftliche Forschung. Während Software in vielen Disziplinen schon lange eine zentrale Rolle spielte, wurde erst allmählich deutlich,

dass es für deren Entwicklung und nachhaltige Pflege spezifische Kompetenzen benötigt, die weder vollständig in der klassischen Informatik noch in der reinen Forschung verortet sind.

[1] software.ac.uk

Im März 2012 traf sich eine kleine Gruppe zu einem vom Software Sustainability Institute (SSI)[1] ausgerichteten „Collaborations Workshop“ am Queen’s College Oxford. Dort wurde der Begriff „Research Software Engineer“ als Berufsbezeichnung und „Research Software Engineering“ für die Praxis der in diesem Bereich Tätigen geprägt. Ziel war es, über den Mangel an Karrieremöglichkeiten für Softwareentwickler:innen in der Wissenschaft zu diskutieren und dem Berufsbild mehr Anerkennung zu verschaffen.

Am ersten RSE-Workshop an der Universität Oxford 2013 nahmen gerade einmal 56 Personen teil. Nach diesem Workshop wurde die UK RSE Association gegründet, wobei das SSI die noch junge Gemeinschaft vertrat. Im Jahr 2016 veranstaltete das SSI die erste RSE-Konferenz und 2017 das erste internationale Treffen der RSE-Führungskräfte in Großbritannien. Im Jahr 2019 entstand zudem die gemeinnützige Gesellschaft „The Society of Research Software Engineering“ (SocRSE)[2], um dem Berufsstand und der Gemeinschaft eine Organisation als Impulsgeber für die notwendigen Veränderungen zur Seite zu stellen.

[2] society-rse.org

All diese Aktivitäten inspirierten die Gründung weiterer nationaler RSE-Verbände in den USA und in Deutschland, denen bald weitere Gruppen in Skandinavien, Australien und Neuseeland folgten, siehe Abbildung 1. Auch im eScience Center[3] in den Niederlanden ist die Unterstützung von Forschungssoftware ein wesentliches Ziel, hier sogar auf nationaler Ebene. In Deutschland formierte sich 2018 die nationale Gruppe "de-RSE e.V."[4]. Ihre Ziele sind vielfältig, die Mitgliedschaft steht allen Interessierten offen. de-RSE veranstaltet jährlich eine Konferenz und bietet im Online-Community-Bereich Möglichkeiten zur Vernetzung und zum Austausch.

[3] esciencecenter.nl

[4] de-rse.org



Abbildung 1. Weltkarte der 2025 bestehenden (inter-)nationalen RSE Vereinigungen

Neben diesen nationalen oder zumindest überregionalen Initiativen gibt es zudem eine Vielzahl an domänenspezifischen Gruppen, die sich für Forschungssoftware in ihrem jeweiligen Gebiet einsetzen. Vorreiterin in Deutschland ist hier die Fachgruppe RSE der Gesellschaft für Informatik, die sich mit RSE aus der Sicht der Informatik beschäftigt. Hier geht es insbesondere um die Frage der Informatik-Forschung auf dem Gebiet von RSE [7], also der Forschung über Forschungssoftware, über Methoden der Entwicklung von Forschungssoftware, aber auch über geeignete Lehr- und Weiterbildungsangebote. Hier arbeitet die Fachgruppe eng mit der de-RSE zusammen, die jeweiligen Arbeitsgruppen sind oft gemischt besetzt. Es sei jedoch nicht verschwiegen, dass die Sicht auf Forschungssoftware zwischen der Informatik und anderen Disziplinen derzeit eher auseinander geht, wie oben bereits angedeutet.

Auch in der GAMM (Gesellschaft für Angewandte Mathematik und Mechanik) gibt es seit kurzem eine Gruppe, die sich um das Thema RSE, hier zusammen mit dem Thema Forschungsdatenmanagement, kümmert. Die Entwicklung von Forschungssoftware und der Umgang mit Forschungsdaten sind seit Jahrzehnten für viele Bereiche innerhalb der GAMM von entscheidender Bedeutung und reichen von Simulationsskripten und Daten zur Begleitung von Publikationen bis hin zu umfangreichen Softwarebibliotheken und Petabytes an Daten für spezifische Anwendungsbereiche. Der GAMM Fachausschuss für Forschungssoftwareentwicklung und Forschungsdatenmanagement in Mathematik und Mechanik (oder kurz GAMM FA RSE&RDM)[5] will hier Impulse aus der Perspektive der computergestützten Wissenschaften geben, Synergien im wissenschaftlichen Austausch zwischen den Fachausschüssen fördern und GAMM-spezifische Infrastrukturanforderungen und Handlungsfelder identifizieren.

Zudem gibt es weitere Interessenvertretungen in verschiedenen Einrichtungen oder als Teil größerer Initiativen, wie zum Beispiel nfdi.software innerhalb der Base4NFDI oder das Joint Lab HiRSE[6] in der Helmholtz-Gemeinschaft.

Besonders hervorzuheben sind außerdem die bereits existierenden institutionellen RSE-Gruppen, beispielsweise an der SUB Göttingen[7], der TU Braunschweig [8][8], der Universität Jena[9], dem Forschungszentrum Jülich[10] oder der Universität Heidelberg [9][11], um nur diejenigen mit direktem GAMM Bezug zu nennen. Sie zeigen jeweils, wie Forschungssoftware und Forschungssoftwareentwicklung an einzelnen Einrichtungen erfolgreich verankert werden konnten. Sowohl die Entstehungsgeschichte als auch die Ausrichtung dieser Gruppen sind sehr unterschiedlich, was unter anderem die Anfertigung eines Positionspapiers des de-RSE e.V. zu institutionellen RSE-Gruppen [10] motiviert hat. Gerade mit Blick auf die Handreichung der DFG zum „Umgang mit Forschungssoftware im Förderhandeln der DFG“ [11] und der GI- und DE-RSE Muster-Leitlinie zur effizienten Entwicklung von Forschungssoftware [12] gewinnt die Etablierung solcher Gruppen an Universitäten und außeruniversitären Forschungseinrichtungen immer mehr an Bedeutung, sowohl lokal als auch national.

[5] gamm-ag-rse-rdm.github.io

[6] helmholtz-hirse.de

[7] sub.uni-goettingen.de/digitale-bibliothek/service-research-software-engineering

[8] tu-braunschweig.de/struktur/praesidium/vp-dn/digital-science-support-lab

[9] zedif.uni-jena.de

[10] fz-juelich.de/rse

[11] ssc.uni-heidelberg.de

Praxis

Die Entwicklung und die Pflege von Forschungssoftware wird häufig unter dem Begriff des Research Software Engineerings zusammengefasst. Hierbei werden klassische Prinzipien aus der Softwaretechnik (Software Engineering) auf wissenschaftliche Software angewandt. Was genau Research Software Engineering umfasst, ist jedoch nicht klar definiert und kann zwischen Domänen, Gruppierungen und Kontext variieren.

Für diesen Artikel nutzen wir die FAIR-Prinzipien in ihrer Ausprägung für Forschungssoftware (FAIR4RS). FAIR steht für Findable, Accessible, Interoperable, Reusable und kommt eigentlich aus dem Bereich des Forschungsdatenmanagements [13]. Dort beschreiben diese Prinzipien den Idealzustand von offenen Forschungsdaten: die Daten inklusive der Metadaten sollen für Mensch und Maschine auffindbar, zugänglich, interoperabel mit etablierten Standards und nachnutzbar sein.

Hierbei wird deutlich, dass Forschungssoftware und Forschungsdaten unterschiedliche Elemente des Forschungsprozesses sind: Forschungssoftware kann Forschungsdaten erzeugen, etwa in Form von Simulationsausgaben, oder sie dient dazu, bestehende Daten zu verarbeiten, zu analysieren und interpretierbar zu machen. Forschungsdaten können prinzipiell auch unabhängig von Software existieren, etwa in Form handschriftlicher Messprotokolle, doch in der heutigen Forschungspraxis ist der Einsatz spezialisierter Software nahezu unverzichtbar, um Daten in strukturierter und nachnutzbarer Form zu erzeugen und zu nutzen. In der wissenschaftlichen Diskussion werden Forschungsdaten und Forschungssoftware daher als eigenständige, gleichwertige Forschungsergebnisse wahrgenommen.

In [14] wurde daher der FAIR-Gedanke auf Forschungssoftware angewandt und konkretisiert. Wir können diesen Ansatz nutzen und daraus Techniken ableiten, die im Bereich des RSE wichtig sind, um diesen Idealzustand für Forschungssoftware und die zugehörigen Metadaten zu errei-

chen.

Hierzu betrachten wir die folgenden fünf Bereiche:

Dokumentation

Eine zentrale Voraussetzung für die Nachnutzbarkeit und Verständlichkeit von Forschungssoftware ist eine konsistente und zugängliche Dokumentation. Sie sollte beschreiben, welche Funktionen die Software bereitstellt, wie sie installiert und genutzt werden kann, und unter welcher Lizenz sie verbreitet wird. Eine gute Dokumentation erleichtert nicht nur die Nutzung durch Dritte, sondern unterstützt auch die langfristige Wartung durch die ursprünglichen Entwickler:innen selbst. Dabei empfiehlt es sich, etablierte Werkzeuge und Konventionen einzusetzen, um eine einheitliche Struktur und Lesbarkeit zu gewährleisten. Damit unterstützt der Bereich Dokumentation die Punkte Interoperabilität und Nachnutzbarkeit im Sinne der FAIR4RS-Prinzipien.

Community

Die Einbindung einer aktiven Nutzungs- und Entwicklungsgemeinschaft trägt wesentlich zur Nachhaltigkeit von Forschungssoftware bei. Transparente Kommunikationskanäle – etwa Issue-Tracker, Beitragsrichtlinien oder Mailinglisten – schaffen Möglichkeiten für Rückmeldungen, Beiträge und gemeinsames Weiterentwickeln. Offene Beteiligungsstrukturen erhöhen die Chancen, dass externe Nutzer:innen zu aktiven Mitwirkenden werden. Entscheidend ist hierbei die Orientierung an etablierten Standards sowie ein respektvoller, inklusiver Umgang in der Interaktion. Diese Standards können in einem Code of Conduct (siehe z.B. NumFOCUS[12] oder The Carpentries[13]) festgehalten werden. Im Sinne der FAIR4RS-Prinzipien werden so die Zugänglichkeit, die Nachnutzbarkeit und auch die Auffindbarkeit erhöht.

[12] numfocus.org/code-of-conduct

[13] docs.carpentries.org/policies/coc/

Versionierung

Die Nachvollziehbarkeit wissenschaftlicher Ergebnisse setzt voraus, dass Forschungssoftware versioniert wird. Jede veröffentlichte Version sollte einen eindeutigen Identifikator besitzen, um Referenzierbarkeit und Zitation zu ermöglichen. Die Verwendung von Versionskontrollsystemen wie Git erlaubt es, Änderungen transparent nachzuvollziehen und mit klaren Commit-Nachrichten oder Release Notes zu dokumentieren. Eine konsistente Versionierung unterstützt nicht nur die Auffindbarkeit und Nachnutzung durch Dritte im Sinne von FAIR4RS, sondern erleichtert auch die langfristige Weiterentwicklung sowohl durch die Autor:innen selbst als auch durch neue Entwickler:innen. Zusätzlich können Nutzende damit im Sinne der Reproduzierbarkeit kennzeichnen, mit welcher Version einer Software sie die jeweiligen Ergebnisse erzeugt haben.

Automatisierung

Automatisierte und transparente Test-Verfahren stärken das Vertrauen in Forschungssoftware, indem sie deren Funktionsfähigkeit kontinuierlich überprüfen. Tests, die die Korrektheit und Nutzbarkeit des Codes belegen, sollten möglichst automatisiert ausgeführt werden, etwa über Continuous-Integration-Pipelines. Solche Prozesse reduzieren Fehlerquellen, fördern die Wiederholbarkeit wissenschaftlicher Ergebnisse und erleichtern die Weiterentwicklung in kollaborativen Umgebungen, alles im Sinne von FAIR4RS. Auch hier gilt es, etablierte Standards zu verwenden, anstatt individuelle, schwer übertragbare Lösungen zu entwickeln.

Nachhaltigkeit

Um ein Softwarewerkzeug im Sinne von FAIR4RS nachhaltig zu machen, bedarf es verschiedener Aspekte. Die Langlebigkeit braucht neben der oben angesprochenen Communityarbeit auch klare Vorgaben zum Umgang mit der Software durch das Team. Viele dieser Details werden durch Software Management Pläne, aber auch Entwicklerdokumentationen abgedeckt. Im Wissenschaftskontext besonders wichtig sind hier Dokumentationen von Entscheidungsprozessen und Entscheidungen etwa zum Softwaredesign oder der Wahl von Algorithmen, aber auch der Übergabeprozesse

bei einem Wechsel der Entwickler:innen, etwa bei Finanzierungslücken. Zunehmend wichtiger werden auch “grüne” Nachhaltigkeitsaspekte, sowohl im Bereich der Automatisierung, aber auch im Bereich der Algorithmen. Codewartung und -refactoring sind essenzielle Voraussetzungen für all diese Aspekte, können aber nur mit passender Ausbildung umfassend bedient werden.

In diesen Bereichen finden sich viele der Kerntechniken des modernen RSEs. In Angeboten wie denen von GitLab oder GitHub sind viele der dazu nötigen Ressourcen und Instrumente bereits automatisch verfügbar: Issue Tracker, Automatisierungsumgebungen, Versionierung, Vorschläge für den Umgang mit der eigenen Community z. B. via Codes of Conduct, Zitierbarkeit durch eine Anbindung an Dienste wie Zenodo und vieles mehr.

Es gibt viele weitere Ansätze, RSE als Themenfeld zu definieren. Hier sei insbesondere auf die klassischen Werke von Greg Wilson zu „Best Practices for Scientific Computing“ [15] und „Good Enough Practices in Scientific Computing“ [16] verwiesen. Einen etwas anderen Blickwinkel erhält man zudem durch die Betrachtung der für modernes RSE nötigen Kompetenzen. Hierzu gibt es ein Positionspapier der deutschen RSE-Community, in dem diese Kompetenzen sehr detailliert und in enger Verbindung zu Techniken aus dem Bereich des klassischen Software-Engineerings analysiert wurden [17].

Aufmerksamkeit

Auch wenn die Entwicklung, Pflege und nachhaltige Nutzung von Forschungssoftware zunehmend an Bedeutung für den wissenschaftlichen Fortschritt gewinnt, ist die Finanzierung dieses Bereichs bislang fragmentiert und lückenhaft. Während Forschungssoftware für viele Disziplinen eine essenzielle Grundlage bildet, wird ihre Finanzierung häufig, wenn überhaupt, nur indirekt berücksichtigt.

In Deutschland bestehen einzelne gezielte Fördermöglichkeiten über die Deutsche Forschungsgemeinschaft (DFG). Dazu gehören sowohl spezielle Ausschreibungen, die explizit auf RSE abzielen, als auch Programme zur Förderung von Forschungssoftware-Infrastrukturen[14]. Allerdings fehlt es bislang an einer systematischen, dauerhaften institutionellen Förderung. Bundesministerien wie das Bundesministerium für Forschung, Technologie und Raumfahrt oder das Bundesministerium für Digitales und Staatsmodernisierung haben bisher keine spezifischen Förderlinien für Forschungssoftware etabliert. Ersteres hat unter dem etwas breiteren Begriff der “Open Research Tools” eine erste Studie durchgeführt, bei der es um ähnliche Anliegen wie denen des RSEs geht [18]. Auf Landesebene existieren bereits einzelne Initiativen, etwa in Baden-Württemberg [19] oder Niedersachsen[15], bei denen nachhaltige Forschungssoftwareentwicklung explizit im Vordergrund steht.

[14] dfg.de/fsi

[15] hochschuledigital-niedersachsen.de/project/lower-saxony-digital-science-support

Auf europäischer Ebene stehen verschiedene Programme offen, die teilweise auch Forschungssoftware berücksichtigen. Besonders hervorzuheben sind EuroHPC-Initiativen, die beispielsweise sogenannte „Lighthouse Codes“ unterstützen. Auch im Bereich der EuroHPC Center of Excellences (CoEs) spielt die Entwicklung von Forschungssoftware eine wesentliche Rolle. Zudem bietet die European Open Science Cloud (EOSC) Fördermöglichkeiten, die sich auch auf die Entwicklung und Zugänglichkeit von Forschungssoftware auswirken. Allerdings sind diese Programme nicht spezifisch auf RSE zugeschnitten, sondern binden Forschungssoftware in übergeordnete digitale Strategien ein.

Neben den öffentlichen Förderinstitutionen spielen private Stiftungen und Organisationen eine Rolle. Beispiele sind die Klaus-Tschira Stiftung oder die Eclipse Foundation, die gezielt Projekte im Bereich Forschungssoftware unterstützen. So wird im Rahmen der aktuellen Forschungssoftware-Förderlinie der Klaus-Tschira Stiftung etwa ein Curriculum für RSE Masterstudiengänge entwickelt, ein nationaler Software Award ins Leben gerufen und die Machbarkeit und Zukunft eines deutschlandweiten Forschungssoftware-Instituts ergründet. Solche Initiativen sind jedoch bisher eher punktuell und meist thematisch begrenzt.

Trotz bestehender Ansätze bleibt somit ein strukturelles Defizit erkennbar: Es mangelt an nachhaltigen, langfristig gesicherten Finanzierungsmodellen für Forschungssoftware. Vor allem eine stärkere institutionelle Verankerung – etwa in Form einer nationalen RSE-Institution oder einer infrastrukturellen Förderung von Forschungssoftware – könnte dies ändern und Deutschland in diesem Bereich aufholen lassen. Vergleichbare nationale Initiativen existieren bereits in anderen Ländern, wie das Software Sustainability Institute im Vereinigten Königreich oder das eScience Center in den Niederlanden, beide mit verhältnismäßig stabiler, dedizierter Finanzierung. Auch in den USA haben insbesondere die National Labs eine lange Tradition der Unterstützung von Forschungssoftware und auch dort hat sich bereits eine lebendige RSE Community mit struktureller Finanzierung gebildet. Die Beispiele aus diesen Ländern zeigen, wie wichtig eine nachhaltige Förderung für nachhaltige Forschungssoftware ist und wie groß die Auswirkung einer solchen Förderung auf den internationalen Erfolg dieser Software-Produkte sein kann.

Ein Kernproblem sowohl bei der Anerkennung als auch bei der Förderung von Forschungssoftware als wissenschaftliche Leistung ist jedoch die konkrete Bewertung und Zählung.

Während wissenschaftliche Artikel durch Peer Review geprüft werden und dadurch im Idealfall eine unabhängige, anerkannte Bewertung erfahren, fehlen für Forschungssoftware bislang vergleichbare standardisierte Verfahren. Damit Forschungssoftware als wissenschaftlicher Output zählt, muss ihre Bewertung sowohl technische als auch wissenschaftliche Dimensionen berücksichtigen. Wie genau dieses „research software peer review“ durchgeführt werden kann, ist jedoch ein ungelöstes Problem. Als Ausweg bieten sich derzeit Begleitpublikationen in einschlägigen Journals wie JOSS (Journal of Open Source Software), JORS (Journal of Open Research Software) oder ACM TOMS (Transactions on Mathematical Software) an. Hier werden sowohl die Software als auch ein beschreibender Artikel einem Review-Verfahren unterzogen und beides gemeinsam veröffentlicht. Auch im klassischen Publikationsprozess bewegen sich Journals in Richtung der Aufwertung der unterliegenden Software-Beiträge. So etabliert z. B. SISC (SIAM Journal on Scientific Computing) eine Sektion, die auch Softwareveröffentlichungen abdeckt, und das vergleichsweise junge Springer-Nature CSAE (Computational Science and Engineering) Journal betreibt neben dem Editorial Board auch ein Code Review Board, das dediziert die Software zu jeder Publikation einer Qualitätsprüfung unterzieht. Wieder andere Journals beschränken sich auf Reproduzierbarkeits-Badges, die die Verfügbarkeit von Codes zu den Experimenten und die Reproduzierbarkeit der Ergebnisse in verschiedenen Abstufungen anzeigen. Auch das NFDI Konsortium für die Mathematik (MaRDI) hat am Beispiel der Computeralgebra in den vergangenen Jahren einen Fragebogen zur Softwarebewertung entwickelt [20] und im Rahmen mehrerer Fachkonferenzen getestet.

Eine weitere Hürde neben der Bewertung ist zudem die Zählung. Was gilt als Forschungssoftware und was wird als Output gezählt? Sind Version 1.1 und Version 1.2 einer Software zwei Einträge? Die Antwort hängt ganz stark von der Art und Weise ab, wie die jeweilige Software entwickelt wird: von semantischer Versionierung über datumsbezogenen Releases und sogenannte Nightly Builds, bis zu keinerlei Versionierung, haben alle Varianten Vor- und Nachteile und sollten durch eine einheitliche Zählweise abdeckbar sein.

In der Helmholtz-Gemeinschaft wird aus diesem Grund derzeit ein Konzept für einen Qualitätsindikator entwickelt, mit dem Forschungsdaten- und Forschungssoftwarepublikationen systematisch bewertet werden können [21]. Ziel ist es, neben klassischen quantitativen Metriken (z. B. Publikationszahlen, Zitationen) auch qualitative Aspekte wie Dokumentation, Nachvollziehbarkeit oder wissenschaftliche Einbettung erfassbar zu machen.

Kern des Ansatzes ist die Kombination der FAIR-Prinzipien mit einem Reifegradmodell. Ein solches Modell beschreibt stufenweise, wie weit bestimmte Qualitätsmerkmale ausgeprägt sind: von „nicht vorhanden“ über „grundlegend“ bis hin zu „vollständig etabliert und optimiert“. Für Forschungssoftware können diese Attribute wie Vergabe eines persistenten Identifikators, strukturierte Versionierung, ausführliche Metadaten, klare Lizenzierung, Interoperabilität mit Standards, Test- und Reproduzierbarkeitsmechanismen oder nachhaltige Wartung sein. Jeder dieser Aspekte wird auf einer Skala (z. B. 0–4) eingeordnet.

Die so entstehenden Reifegrade lassen sich pro Dimension aggregieren und grafisch darstellen (z. B. in Radarplots, siehe Abbildung 2). Dadurch entsteht ein differenziertes Bild der Stärken und Schwächen einer Daten- oder Software-Publikation, das institutionelle Steuerung, Förderentscheidungen und Qualitätssicherung unterstützt. Darauf aufbauend kann dann eine Art Minimalpolygon definiert werden, mit dem entschieden werden kann, ob die Qualität einer Software ausreicht, um formal als Publikation gezählt zu werden. Herausforderungen bestehen derzeit noch in der disziplinübergreifenden Vergleichbarkeit, in der Festlegung geeigneter Gewichtungen des Minimalpolygons sowie in der praktischen Umsetzung, da manche Kriterien automatisierbar sind (z. B. Vorhandensein einer Lizenz), andere jedoch qualitative Begutachtung erfordern (z. B. Verständlichkeit der Dokumentation). Zudem löst dieser Ansatz noch nicht die Frage nach dem Umgang mit verschiedenen Versionen bzw. Releases, ist aber ein erster Schritt hin zu einem flexiblen, partizipativen Instrument, das die Sichtbarkeit und Nachnutzbarkeit von Forschungssoftware verbessern und zugleich zur Anerkennung dieser Publikationsform beiträgt.

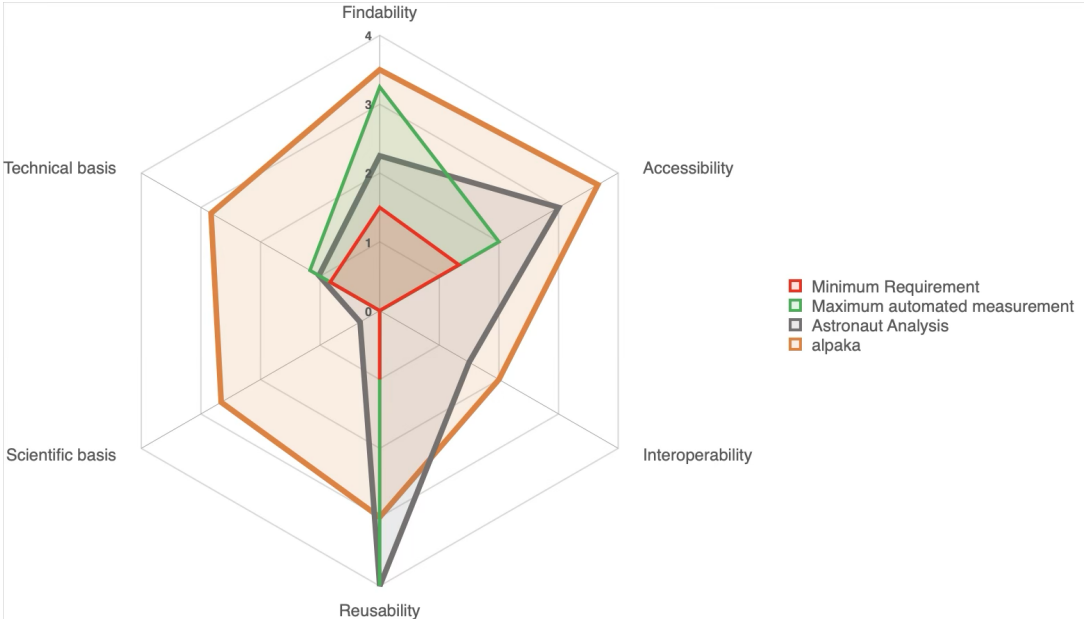

Abbildung 2. Netzdiagramm (radar plot) zur Darstellung der Bewertung von Forschungssoftware, aus [22]

Die Anerkennung von Forschungssoftware als zählbarer wissenschaftlicher Output erfordert einen kulturellen Wandel im Wissenschaftssystem. Dazu gehören die Entwicklung geeigneter Peer-Review-Verfahren, die Etablierung klarer Standards für Softwarezitation und die Verankerung von Software in Forschungsbewertungssystemen. Ziel ist es, den Aufwand und die Expertise, die in die Entwicklung und Pflege von Forschungssoftware fließen, sichtbar zu machen und gleichwertig mit anderen wissenschaftlichen Outputs anzuerkennen.

Zu diesem Wandel gehört es auch, dass Personen, die sich auf die Entwicklung von Forschungssoftware fokussiert haben, langfristig zur Wissenschaft beitragen können. Ein Weg zu diesem Ziel sind die oben erwähnten Infrastruktureinheiten. Diese unterstützen „Scientists who code“, indem sie z. B. bei Standardaufgaben Hilfestellungen geben und so den Forschenden mehr Zeit für ihre Kernforschungsfragen ermöglichen. Andererseits generieren sie Karrierewege für Forschungssoftwareentwickler:innen und tragen so zur Nachhaltigkeit der an den Standorten entwickelten Softwarewerkzeuge bei, weil dediziertes Personal für Wartungs- und Refactoringarbeiten existiert. Die Kombination aus Fachwissen und Softwarekompetenz macht diese Fachkräfte zu wertvollen Projektpartner:innen, statt reinen Dienstleister:innen, wie die Leuchtturmprojekte in den existierenden institutionellen RSE-Gruppen schon heute zeigen.

Fazit

Forschungssoftware ist längst zu einem unverzichtbaren Bestandteil wissenschaftlicher Praxis geworden. Von vielen wird sie nicht nur als Hilfsmittel verstanden, sondern als eigenständiger wissenschaftlicher Output, der zum Beispiel die Generierung, Analyse und Interpretation von Forschungsdaten erst ermöglicht. Ihre Besonderheit liegt darin, dass sie in der Regel nicht von professionellen Softwareentwickler:innen, sondern von Domänenwissenschaftler:innen entwickelt wird, was spezifische Herausforderungen für Qualität, Nachhaltigkeit und Anerkennung mit sich bringt. Aus diesem Grund hat sich das Feld des Research Software Engineerings etabliert, das Prinzipien der Softwaretechnik auf den wissenschaftlichen Kontext überträgt und zugleich eigene methodische, organisatorische und kulturelle Anforderungen berücksichtigt.

Die nationale RSE-Community, unterstützt durch domänenspezifische und lokale Initiativen, hat in den vergangenen Jahren wichtige Impulse gesetzt, um Forschungssoftware sichtbarer zu machen, Standards für ihre Entwicklung zu formulieren und Strukturen für ihre nachhaltige Nutzung zu schaffen. Möglichkeiten zur Mitgestaltung der Bewegung gibt es viele, etwa über den offenen de-RSE Matrix Chat[16], auf den jährlich stattfindenden de-RSE Events[17], oder im Rahmen des GAMM Fachausschusses zu RSE & RDM[18]. Zentrale Leitlinien wie die FAIR4RS-Prinzipien bieten Orientierung, wie die eigene Forschungssoftware auffindbar, zugänglich, interoperabel und reproduzierbar gestaltet werden kann.

[16] de-rse.org/en/matrix

[17] de-rse.org/en/events

[18] gamm-ag-rse-rdm.github.io

Dennoch bleibt die Frage der langfristigen Finanzierung und institutionellen Verankerung offen. Ohne stabile Fördermechanismen und ohne die Anerkennung von Forschungssoftware als zählbaren wissenschaftlichen Output droht ein zentrales Fundament moderner Forschung unterbewertet zu bleiben. Die deutsche Wissenschaftslandschaft hat hier noch viel nachzuholen, ist aber mit einigen spannenden Projekten und Kooperationen auf einem guten Weg.

Die weitere Etablierung von Forschungssoftware als gleichwertiger Teil wissenschaftlicher Wertschöpfung erfordert ein Umdenken in der Forschungsbewertung, klare Standards für Qualitätssicherung und Zitation sowie nachhaltige Förderstrukturen. Erst wenn diese Elemente zusammenspielen, lässt sich das Motto „Better Software, Better Research“ vollumfänglich einlösen – mit dem Ziel, wissenschaftliche Erkenntnisprozesse nicht nur effizienter, sondern auch transparenter, reproduzierbarer und zukunftsfähiger zu gestalten.

Literatur

[1] Alys Brett u. a. *Research software engineers: State of the nation report 2017*. Techn. Ber. 2017. DOI: 10.5281/zenodo.495360.

[2] Udit Nangia und Daniel S Katz. „Track 1 paper: Surveying the U.S. National Postdoctoral Association Regarding Software Use and Training in Research“. In: Zenodo, 2017. DOI: 10.5281/zenodo.814103.

[3] Simon Hettrick u. a. *Uk Research Software Survey 2014*. 2014. DOI: 10.5281/zenodo.14809.

[4] Morane Gruenpeter u. a. *Defining Research Software: a controversial discussion*. 2021. DOI: 10.5281/zenodo.5504016.

[5] Wilhelm Hasselbring u. a. „Multidimensional Research Software Categorization“. In: *Comput. Sci. Eng.* 27.2 (2025), S. 59–68. DOI: 10.1109/MCSE.2025.3555023.

[6] Arne Johanson und Wilhelm Hasselbring. „Software engineering for computational science: Past, present, future“. In: *Comput. Sci. Eng.* 20.2 (2018), S. 90–109. DOI: 10.1109/MCSE.2018.021651343.

[7] Michael Felderer u. a. „Investigating research software engineering: Toward RSE research“. In: *Commun. ACM* 68.2 (2025), S. 20–23. DOI: 10.1145/3685265.

[8] Christopher Blech u. a. „SURESOFT: Towards sustainable research software“. In: (2022). DOI: 10.24355/dbbs.084-202210121528-0.

[9] Liam Keegan, Dominic Kempf und Inga Ulusoy. *Scientific Software Center at Heidelberg University: White paper*. 2024. DOI: 10.5281/zenodo.10867903.

[10] Dominic Kempf und et.al. „Establishing central Research Software Engineering units in German research institutions“. In: *Posionspapiere des de-RSE e. V.* 004 (2025). URL: https://de-rse.org/2023_paper-RSE-groups/paper.pdf.

[11] Deutsche Forschungsgemeinschaft. *Umgang mit Forschungssoftware im Förderhandeln der DFG*. 2024. DOI: 10.5281/zenodo.13919790.

[12] Andreas Czerniak u. a. „GI- und DE-RSE Muster-Leitlinie zur effizienten Entwicklung von Forschungssoftware“. In: *Gesellschaft für Informatik e. V. (2025) & Positionspapiere des de-RSE e. V.* 003 (2025). DOI: 10.18420/2025-gi_de-rse.

[13] Mark D Wilkinson u. a. „The FAIR Guiding Principles for scientific data management and stewardship“. In: *Sci. Data* 3.1 (2016), S. 160018. DOI: 10.1038/sdata.2016.18.

[14] Michelle Barker u. a. „Introducing the FAIR Principles for research software“. In: *Sci. Data* 9.1 (2022), S. 622. DOI: 10.1038/s41597-022-01710-x.

[15] Greg Wilson u. a. „Best practices for scientific computing“. In: *PLoS Biol.* 12.1 (2014), e1001745. DOI: 10.1371/journal.pbio.1001745.

[16] Greg Wilson u. a. „Good enough practices in scientific computing“. In: *PLoS Comput. Biol.* 13.6 (2017), e1005510. DOI: journal.pcbi.1005510.

[17] Florian Goth u. a. „Foundational competencies and responsibilities of a Research Software Engineer: Current state and suggestions for future directions“. In: *F1000 Res. (2025) & Positionspapiere des de-RSE e. V.* 002 (2025). DOI: 10.12688/f1000research.157778.2.

[18] Jürgen Streicher u. a. *Open Research Tools: Stand und Herausforderungen im Spannungsfeld von Forschungssoftware, Offenheit und digitaler Infrastruktur*. 2025. DOI: 10.5281/zenodo.15771943.

[19] Dirk von Suchodoletz u. a. „Framework concept of the Universities in the State of Baden-Württemberg : high-performance computing (HPC) and data-intensive computing (DIC) in the period 2025–2032“. 2025. DOI: 10.6094/UNIFR/271667.

[20] JHanselman. *JHanselman/code-review-template: Software reviewing template*. 2025. DOI: 10.5281/zenodo.16362360.

[21] Wolfgang zu Castell u. a. „Towards a Quality Indicator for Research Data publications and Research Software publications – A vision from the Helmholtz Association“. In: (2024). DOI: 10.48550/arXiv.2401.08804. arXiv: 2401.08804 [cs.DL].

[22] Guido Juckeland, Doris Dransch und Bernadette Fritzsch. „Research Software as a new key performance indicator for evaluation in the Helmholtz Association“. In: *Electronic Communications of the EASST* 85 (2025). DOI: 10.14279/eceasst.v85.2697.