



# NEST Code Generation

## Motivation and prior work

INCF Workshop on Code Generation from Model Description Languages

December 8, 2014 | Jochen Martin Eppler <j.eppler@fz.juelich.de>
Institute of Neuroscience and Medicine (INM-6)
Institute for Advanced Simulation (IAS-6)
Jülich Aachen Research Alliance (JARA)

# Outline

- Neuron and synapse models in NEST
- Reasons why we want to generate code
- What is important to us

- What has been done already?
- Where to go from here?

Disclaimer: This talk mainly contains the NEST perspective

# The neural simulation tool NEST

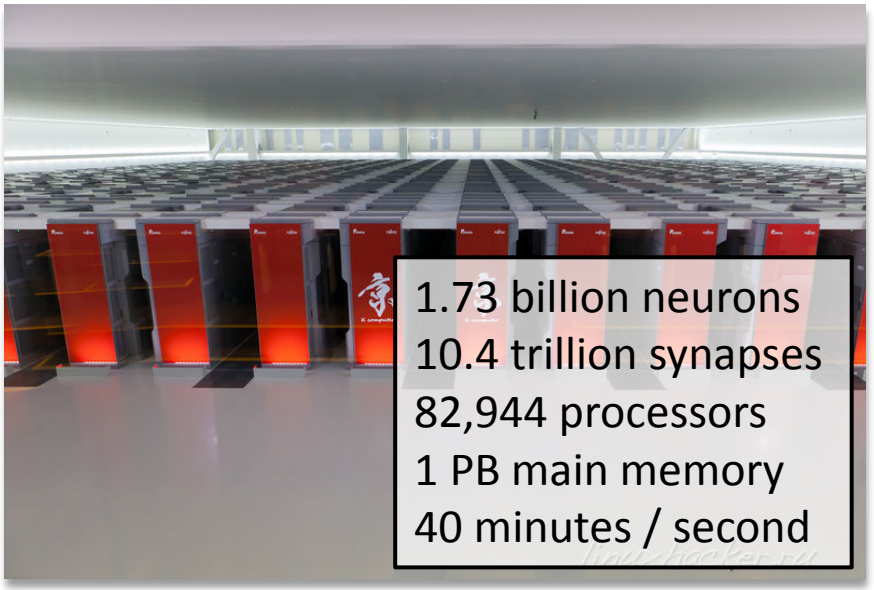
NEST is a hybrid parallel (OpenMP+MPI) simulator for spiking neural networks, written in C++, but with a Python frontend

Neuron models are mainly point neurons, synapses are based off phenomenologic models (STDP, STP, neuromodulation)

The focus of NEST is on large-scale simulations

Read more and get it on

**`nest-simulator.org`**



# Creating neuron models in NEST


1. Copy & paste
2. Modify parts of the code
3. Ideally adapt the comments ;-)
4. Add to Makefiles
5. Re-compile and test
6. Goto 2…

# Creating neuron models in NEST

NEST is C++, while our PhD students are trained in Python, with little or no experience in software engineering

Often, variable names, comments, solvers, and such are not adapted if the code finally works

Writing neurons requires learning about a lot of boring interface functions

# Creating neuron models in NEST

NEST is C++, while our PhD students are trained in Python, with little or no experience in software engineering

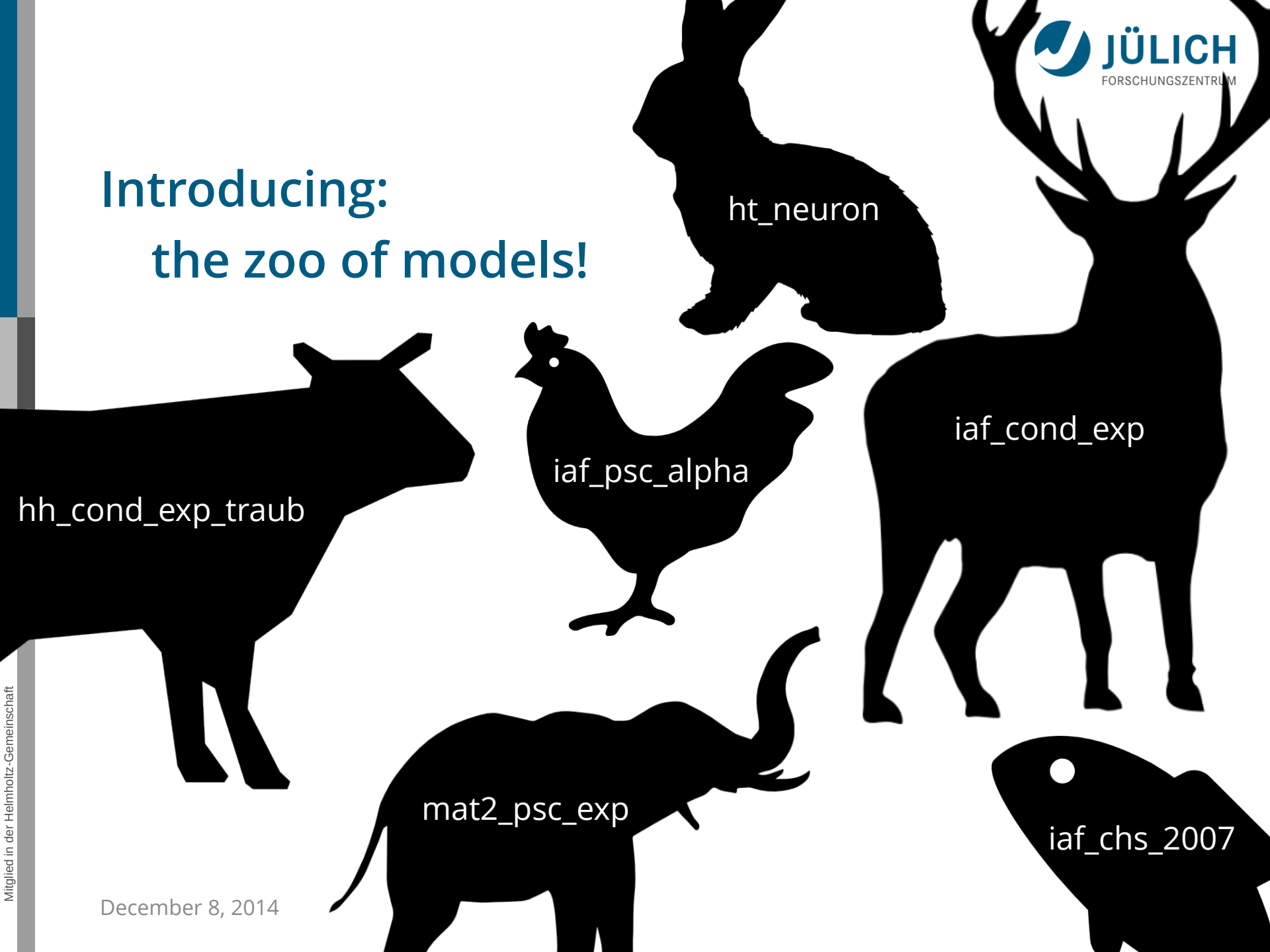
Often, variable names, comments, solvers, and such are not adapted if the code finally works

Writing neurons requires learning about a lot of boring interface functions

→ Decreased code quality, maintainability and correctness

# But despite the intricacies,
# our community was quite productive…

# Introducing:
# the zoo of models!



# NEST 2.6 will have 36 neuron models built in

19 are simple integrate-and-fire models
2 are based on the Hodgkin&Huxley formalism
11 have alpha-shaped post-synaptic responses
10 use exponentially decaying post-synaptic responses
15 with current-based dynamics solved exactly
9 conductance-based neurons using different solvers
plus some more exotic specimen

… and there‘s about 12 synapse models in addition

# The diversity leads to new problems

If we change the simulator API, we have to adapt all models manually, which is tedious and can lead to errors again

Models are not semantically checked for errors, e.g. if units are used correctly in calculations

# The diversity leads to new problems

If we change the simulator API, we have to adapt all models manually, which is tedious and can lead to errors again

Models are not semantically checked for errors, e.g. if units are used correctly in calculations

→ A domain specific language for neuron and synapse models plus code generation could make our lifes much easier!

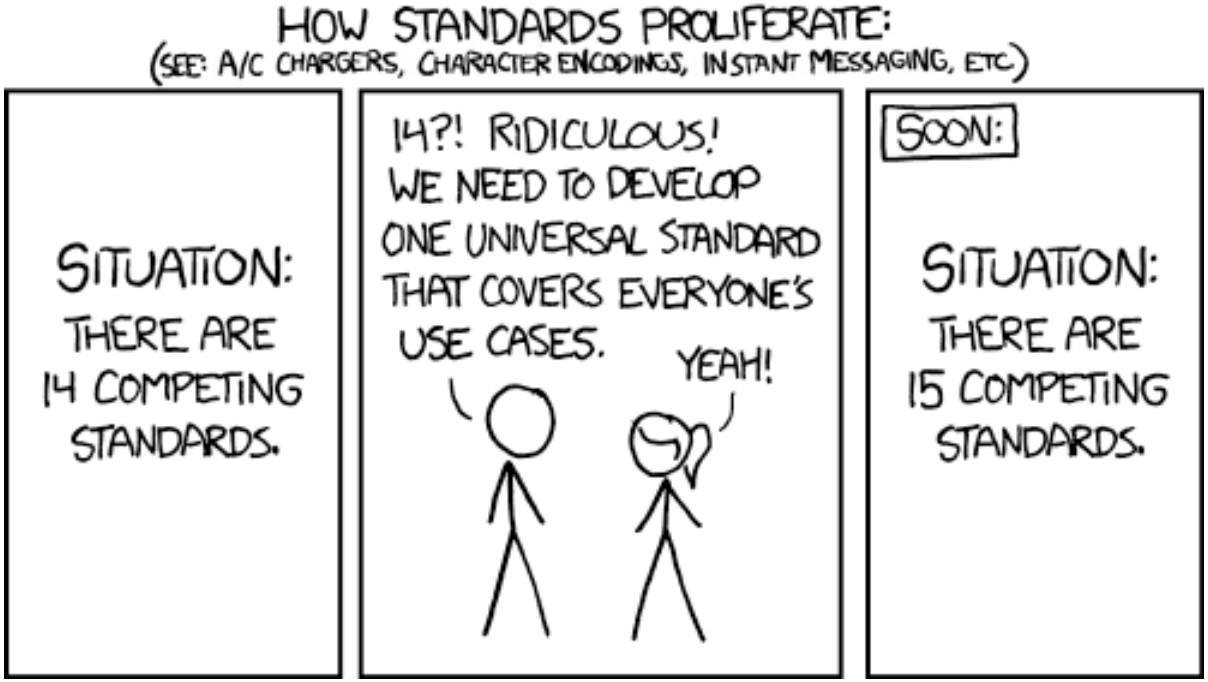
# An imperative modeling DSL for NEST

In a Master's project, we created a prototype of NESTML, which is our test bed for solving the NEST specific problems

It's a Python-like language with units, a notion of parameters and dynamic states, and context conditions

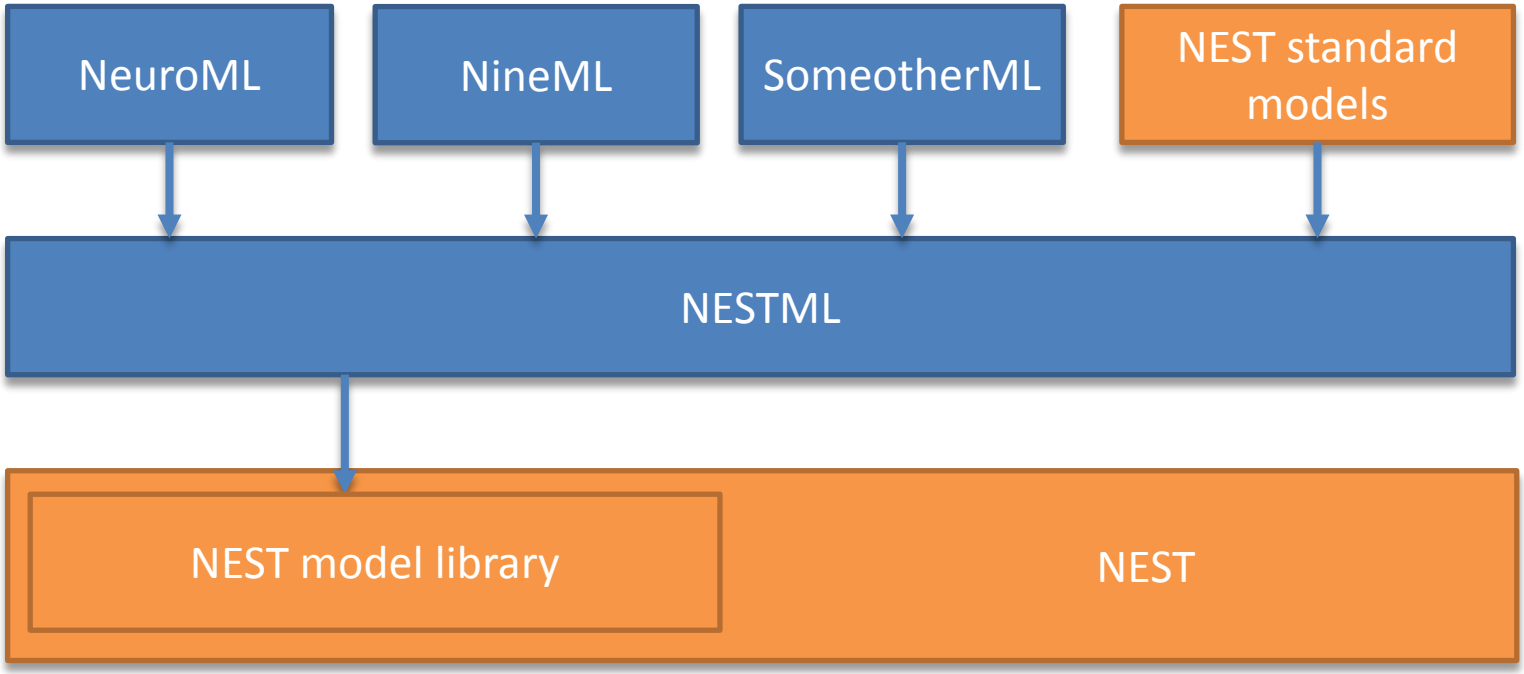
It will be extended to cover all neuron and synapse models throughout a two year project starting now

# Wait, yet another standard?



# Wait, yet another standard?

# More a layer than a standard



# Why is NESTML imperative?

The number one reason is that NEST is written in C++, which itself is iterative, so generating code is easier this way

We want to be able to express the exact way in which the differential equations are solved (cf. linear models)

Things are often expressed more easily in a piece of code than by describing the conditions and entities

# Why is NESTML not based on XML?

# Why is NESTML not based on XML?

> If syntactic sugar didn't count, we'd all be programming in assembly language.
>
> — C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond (Abrahams, Gurtovoy)

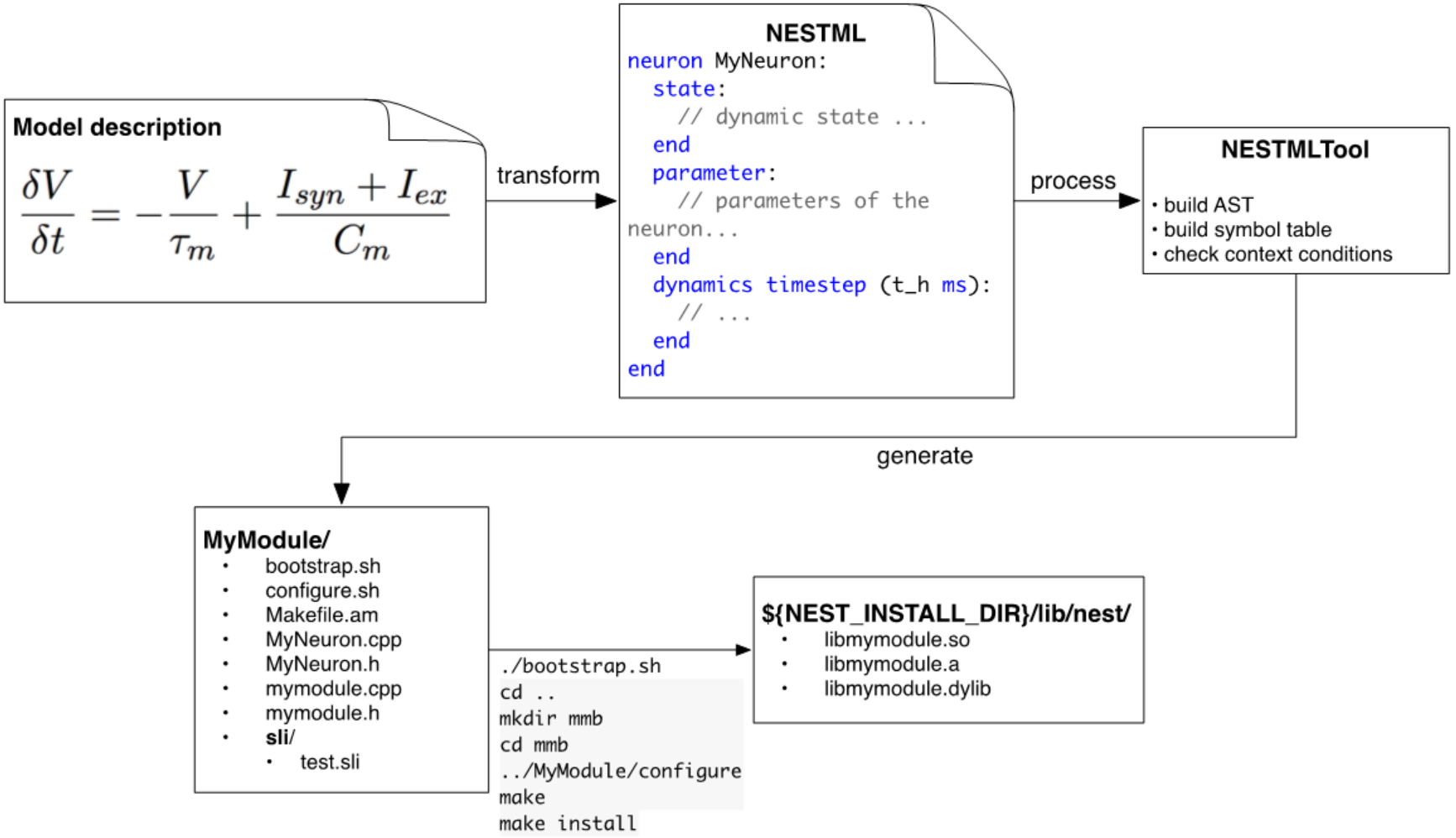
# Why is NESTML not based on XML?

XML is said to be user-readable and –writable, but the tags add a lot of clutter (especially for math)

We would like to have a clean syntax with semantics for all operators and elements, not just literal translations

A custom DSL offers more freedom in general than an embedded DSL (i.e. domain terminology)

# Writing neuron models using NESTML



# Code generation from NESTML

Errors bubble up to the level of the modeling language and are raised there (no C++ compiler error messages anymore)

Context conditions and syntax highlighting help modelers to write better code without even knowing

Generated documentation describes what actually is there

A component library will allow flexible combination of models from dynamics, post-synaptic responses and plasticity rules

# Towards a component-based zoo!

# Relation to NineML, NeuroML, …

We‘re in contact with Tom Close from NineML and Padraig Gleeson from NeuroML to get things going the right way

I applied to become a member of the NineML standardization committee

We‘re planning a community survey and workshops to assess the requirements also of others

# Acknowledgments

Thanks to Inga Blundell, Abigail Morrison, Dimitri Plotnikov and Tammo Ippen for valuable discussions



Comic “standards” by Rundall Munroe from xkcd.com

Last but not least, thanks to INCF and the organizers for making this workshop happen!