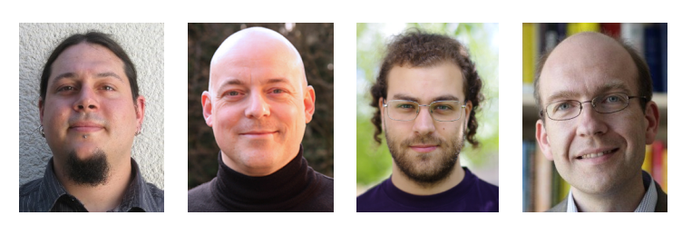
The NEST software development infrastructure

Jochen Martin Eppler[1], Bernd Wiebelt[1], Yury V. Zaytsev[2], Markus Diesmann[1,3,4]

[1] Institute of Neuroscience and Medicine (INM-6), Computational and Systems Neuroscience, Forschungszentrum Jülich
[2] Bernstein Center Freiburg, Albert-Ludwigs-Universität Freiburg
[3] RIKEN Brain Science Institute, Wako-Shi, Japan
[4] Medical Faculty, RWTH Aachen University



Contact: `j.eppler@fz-juelich.de`

Introduction

Software development in the Computational Sciences has reached a critical level of complexity in the recent years. This "complexity bottleneck" shows not only for the programming languages and technologies that are used during development, but also for the infrastructure, which must provide a backbone for sustainable development of large-scale software projects and manageability of the code base [1].

As the development shifts from specialized and solution-tailored in-house code (often developed by a single person or only a few developers) towards more general software packages written by larger teams of programmers, it becomes inevitable to use professional software engineering tools also in the realm of scientific software development. In addition the move to collaboration-based large-scale projects (e.g. BrainScaleS) also means a larger user base, which depends and relies on the quality and correctness of the code.

In this contribution, we present the techniques that helped to support the sustainability of NEST development.

NEST

NEST is a simulator for large heterogeneous networks of point neuron models or neurons with few electrical compartments [2].

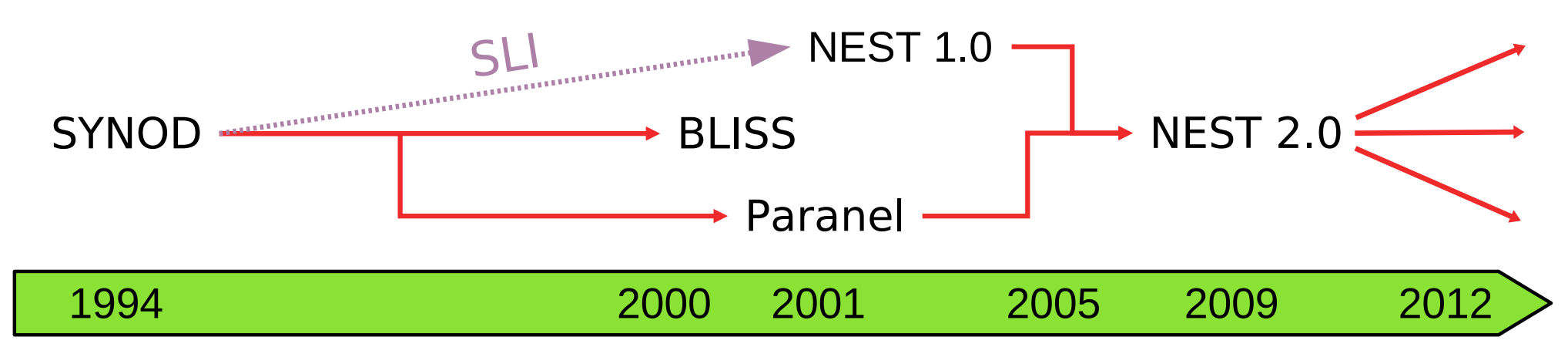
It is suited for a broad range of neuronal network modeling approaches and computer architectures from standard desktop computers to computer clusters or large HPC facilities such as IBM's Blue Gene.

A neural network model simulation in NEST basically imitates an electrophysiological experiment: the user builds a neural system from a set of pre-defined neuron and synapse types that can be connected in an arbitrary fashion and instrumented by devices for stimulation and measurement.

A LiveCD to try NEST without installation and the current release of the source code are available on the website of the NEST Initiative (`http://www.nest-initative.org`), as is extensive documentation and a list of neuroscientific publications that use NEST.

Collaborative development

NEST is developed by the NEST Initiative, a non-profit organization distributed over different labs in Europe.



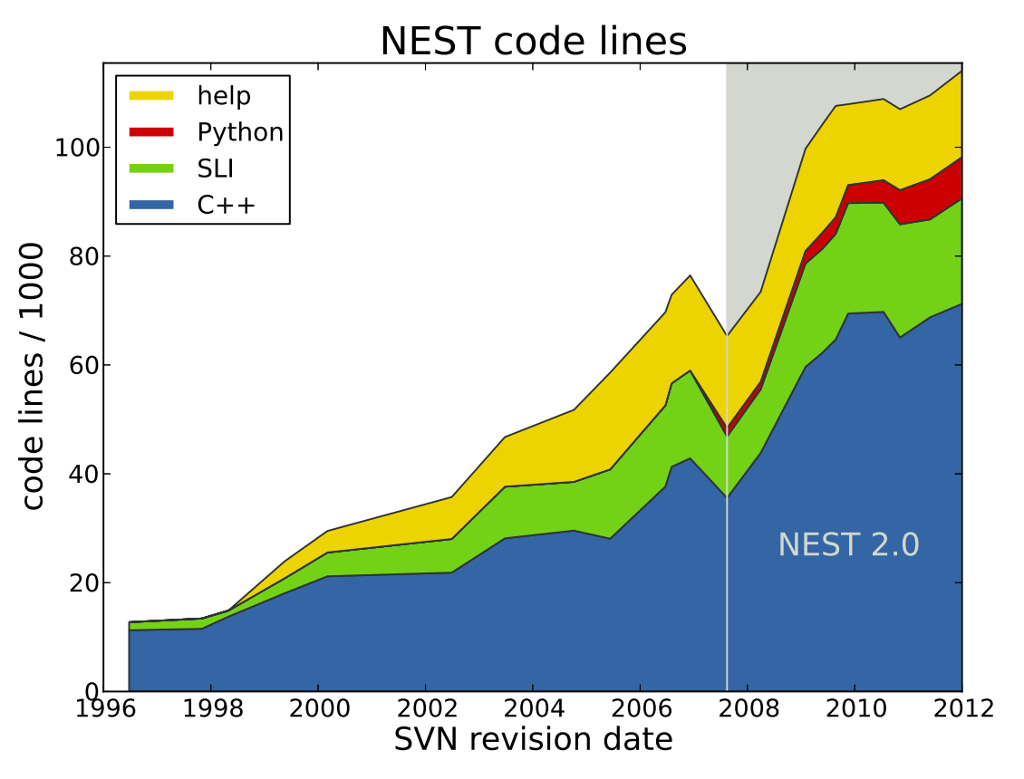
The time line shows that often different versions of NEST were developed concurrently. Branching was necessary in order to address different neuroscientific questions on the one side, but also to explore new algorithms and data structures or new computing platforms on the other side.

Regular video conferences between the different members of the NEST Initiative allow to coordinate the development process, while each lab is still able to pursue its own direction of research.

Currently, we are again in the phase of merging different branches in order to obtain one canonical release of NEST.

Handling the growing complexity

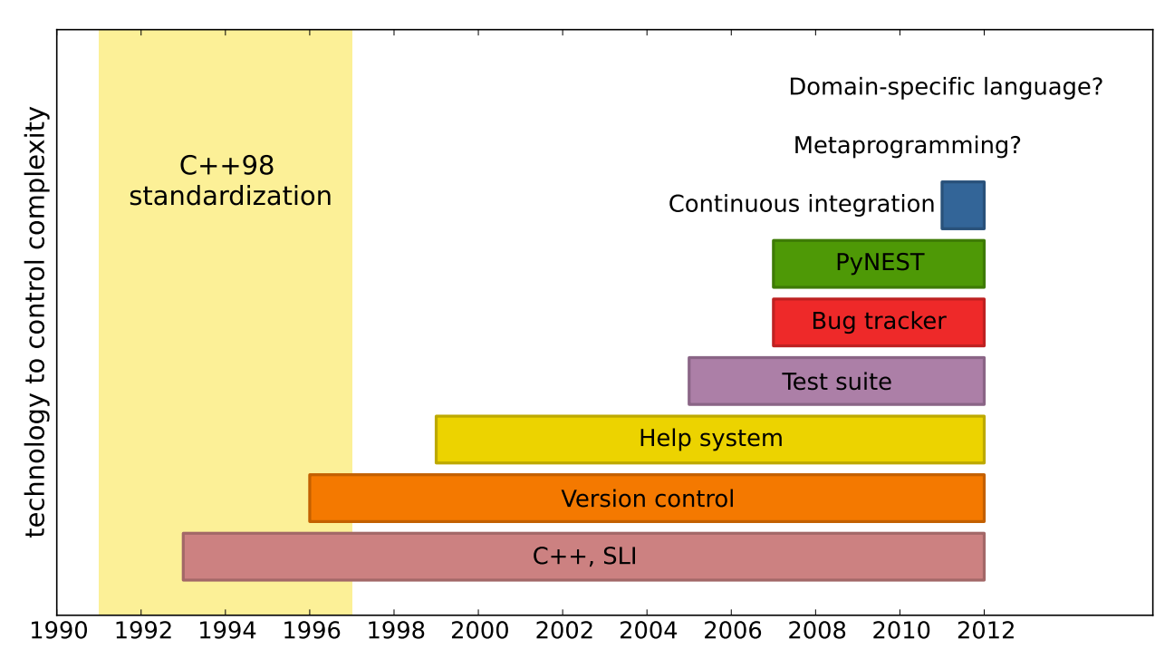
To handle the growing complexity of the code base of NEST (exemplified by the number of lines of code in the figure to the right), we constantly need to explore new software engineering tools and techniques and integrate them into our work-flow.



Important steps in this respect were

- the use of version control to ease collaborative development
- the use of high-level languages for the model specification
- the use of unit tests to ensure correctness of the code
- the use of a system that constantly monitors code quality

The figure below shows the different techniques that were introduced to the NEST development process since its early days.



Only recently, we introduced modern Web 2.0 techniques such as blogs, wikis and WebDAV based storage systems to the development process.

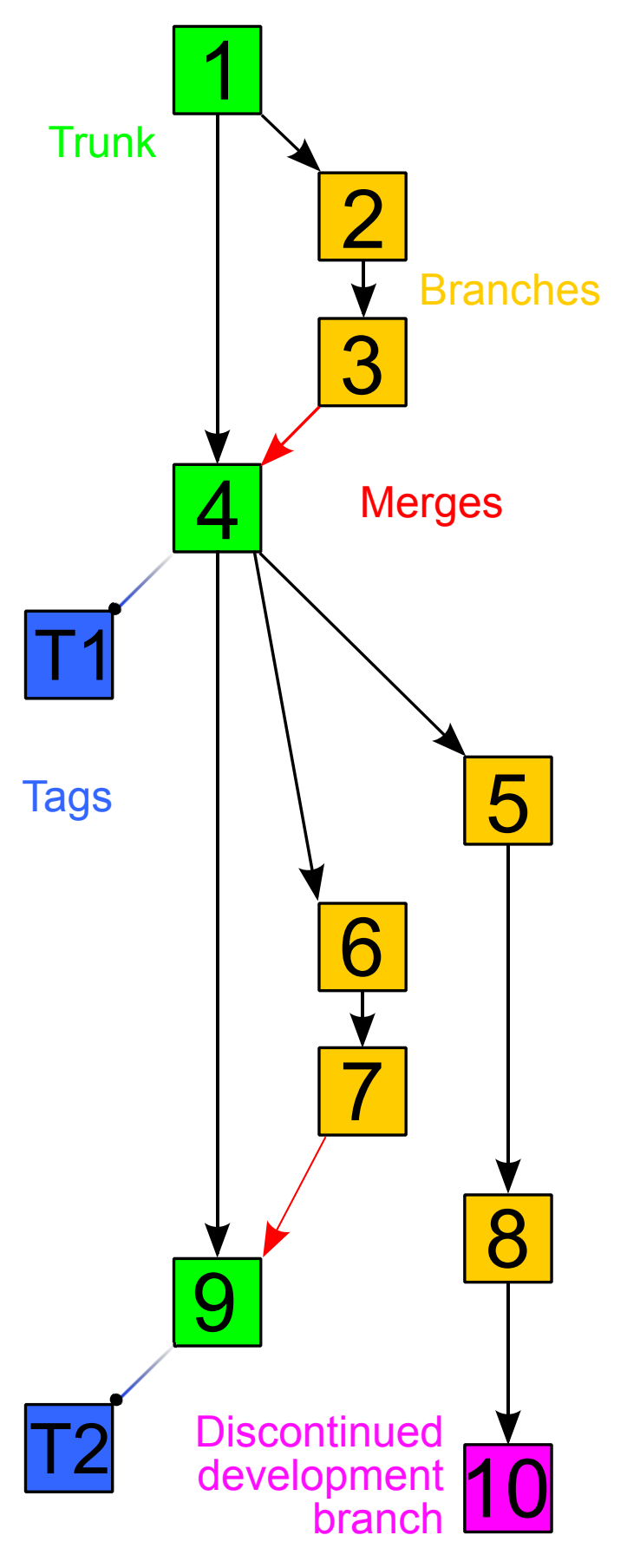
In the future, we plan to increase our use of meta programming and domain-specific languages above our use of C++ templates to handle model generation in a more generic and efficient manner.

Revision Control

Revision control is a powerful concept that can be applied to any collection of files and directories. It is not limited to program development, but can also be used for preparing scientific manuscripts.

Compared to the traditional "shared folder" paradigm, the advantages of revision controlled repositories are numerous.

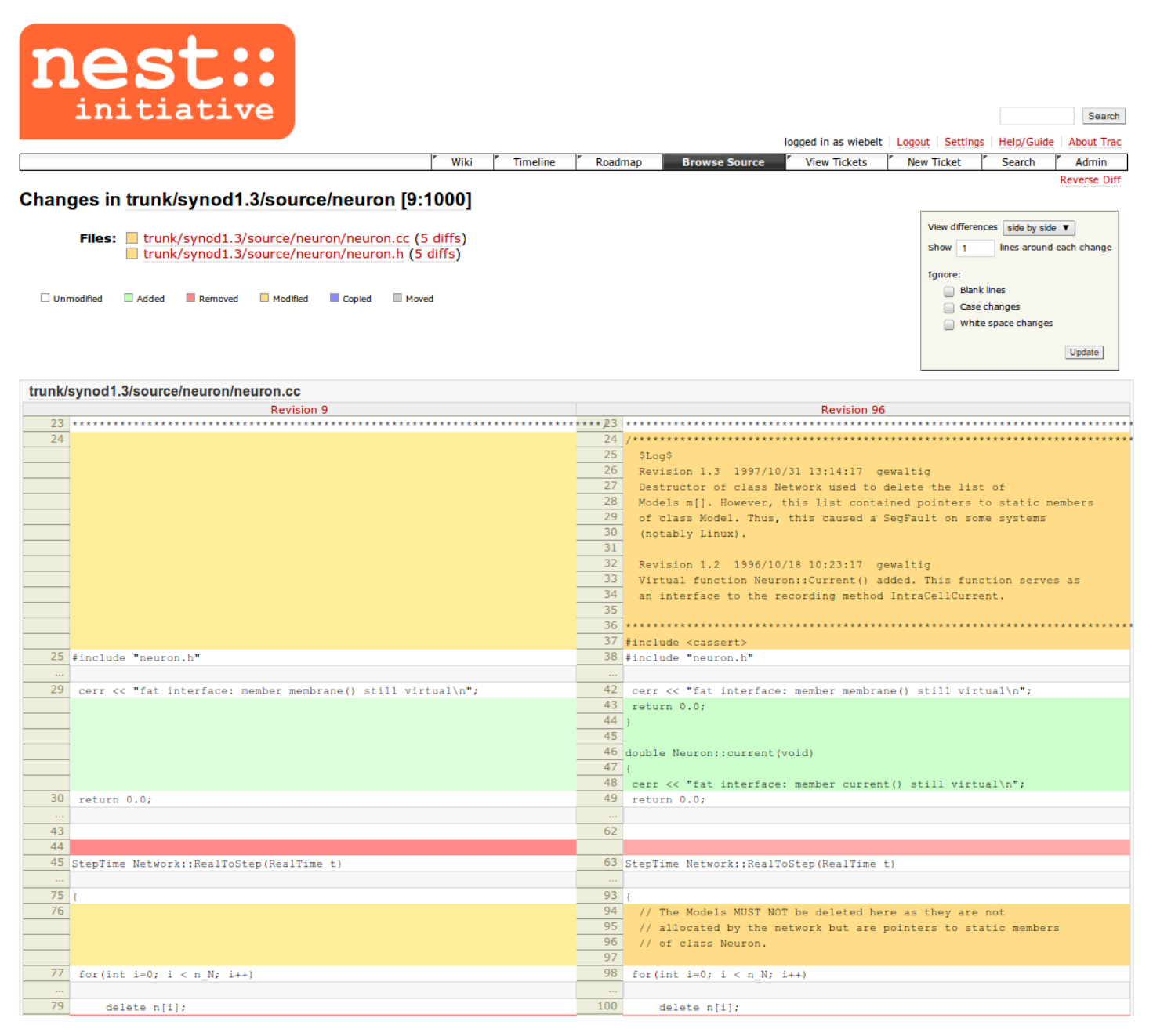
- Built-in unlimited backup/restore: recover any previous version of a file, directory or the whole repository.
- Independence: work on your private ("checked out") local copy of the repository.
- Synchronization: incorporate changes from other collaborators ("update") or share changes made by you ("commit").
- History: find out what was changed in the repository, at what time and by whom.
- Annotation: give a description of your changes on each commit.
- Conflict resolution: merge changes somebody else made to a file you are working on.
- Nonlinear development: branch off another line of development (e.g. "version 2.0", "experimental", "legacy").
- Cherry picking: merge back interesting changes from other branches.



Internet based collaboration services

Internet based services have become ubiquitous in daily life: "Amazon knows what we have, Google knows what we want and Facebook knows who we are."

It is the ease of use that makes these services so popular. Consequently, the NEST infrastructure offers Internet web based services to improve scientific collaboration.



Changes in trunk/synod1.3/source/neuron [9:1000]

Files: trunk/synod1.3/source/neuron/neuron.cc (5 diffs)
trunk/synod1.3/source/neuron/neuron.h (5 diffs)

trunk/synod1.3/source/neuron/neuron.cc

Revision 9 | Revision 96

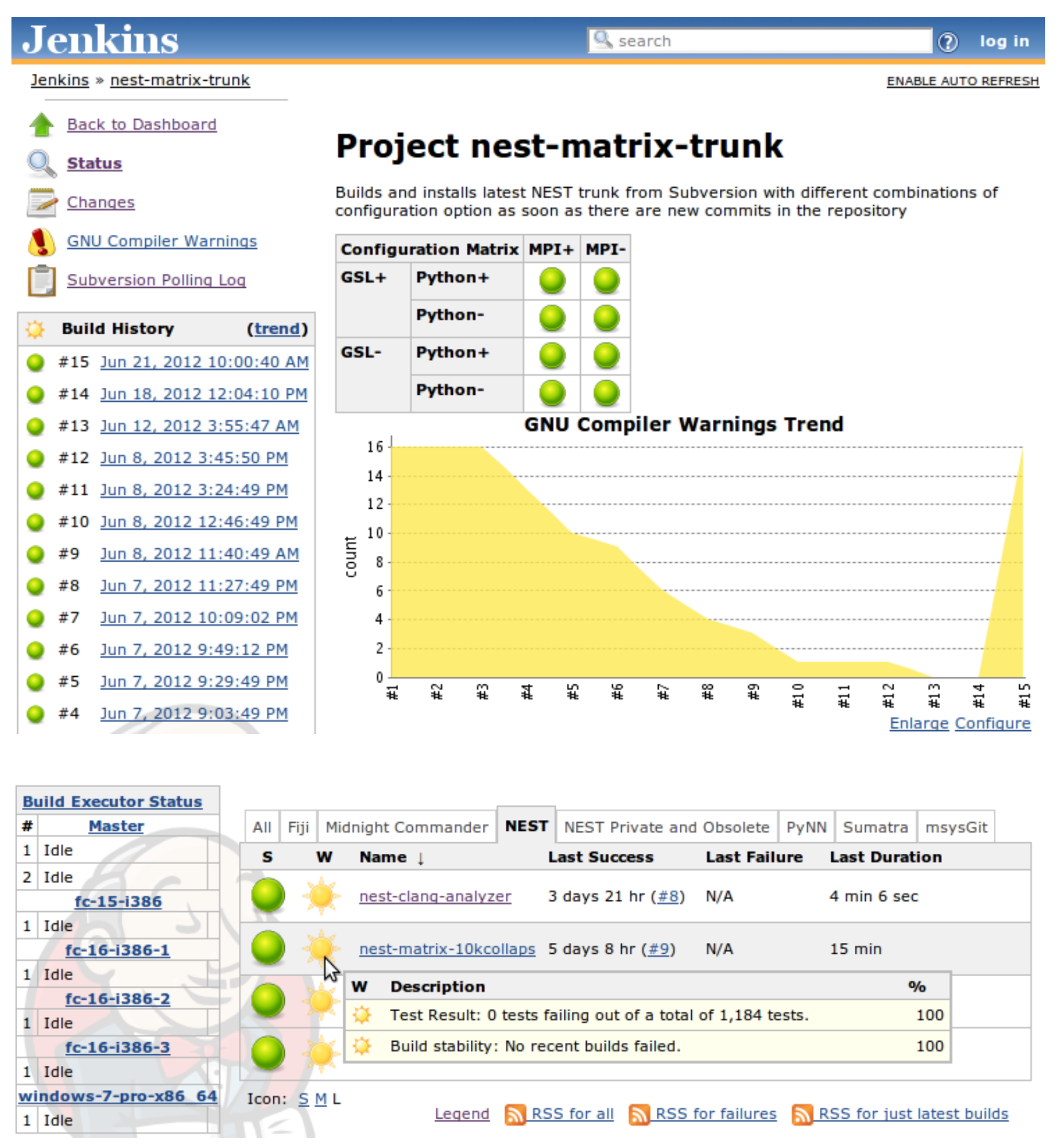
```
/****************************************************************
 $Log$
 Revision 1.3  1997/10/31 13:14:17  gewaltig
 Destructor of class Network used to delete the list of
 Models m[]. However, this list contained pointers to static members
 of class Model. Thus, this caused a SegFault on some systems
 (notably Linux).

 Revision 1.2  1996/10/18 10:23:17  gewaltig
 Virtual function Neuron::Current() added. This function serves as
 an interface to the recording method IntraCellCurrent.

****************************************************************
#include <cassert>
#include "neuron.h"
...
 cerr << "fat interface: member membrane() still virtual\n";
 return 0.0;
}

double Neuron::current(void)
{
 cerr << "fat interface: member current() still virtual\n";
 return 0.0;
...
StepTime Network::RealToStep(RealTime t)
...
{
 // The Models MUST NOT be deleted here as they are not
 // allocated by the network but are pointers to static members
 // of class Neuron.

 for(int i=0; i < n_N; i++)
...
    delete n[i];
```

- Public website: news, general information, documentation, download.
- Mailing lists: public and private mailing lists to keep users and developers up-to-date.
- Internal wiki: information for developers.
- Blogs: real-time information on various sub projects.
- Bibliography database: manage references in scientific publications.
- Repositories: projects (source code, manuscripts) under revision control.
- Issue tracking: report bugs and document the process of fixing them.

Continuous integration

CI is a practice where QA is applied continuously as opposed to the traditional procedure of applying QA during an integration phase after completing all development. It decreases the risks associated with the integration by spreading required efforts over time, which helps to improve software quality and to reduce the time taken to deliver it.

Stringent QA is particularly important for NEST, a simulator with the emphasis on correctness, reproducibility and performance. However, given limited resources, it is wasteful to make the *developers* to re-run the test suite for all target platforms upon every single change to the code base.



Project nest-matrix-trunk

Builds and installs latest NEST trunk from Subversion with different combinations of configuration option as soon as there are new commits in the repository

Configuration Matrix		MPI+	MPI-
GSL+	Python+	●	●
	Python-	●	●
GSL-	Python+	●	●
	Python-	●	●

Build History	(trend)
#15	Jun 21, 2012 10:00:40 AM
#14	Jun 18, 2012 12:04:10 PM
#13	Jun 12, 2012 3:55:47 AM
#12	Jun 8, 2012 3:45:50 PM
#11	Jun 8, 2012 3:24:49 PM
#10	Jun 8, 2012 12:46:49 PM
#9	Jun 8, 2012 11:40:49 AM
#8	Jun 7, 2012 11:27:49 PM
#7	Jun 7, 2012 10:09:02 PM
#6	Jun 7, 2012 9:49:12 PM
#5	Jun 7, 2012 9:29:49 PM
#4	Jun 7, 2012 9:03:49 PM

Build Executor Status	
#	Master
1	Idle
2	Idle
	fc-15-i386
1	Idle
	fc-16-i386-1
1	Idle
	fc-16-i386-2
1	Idle
	fc-16-i386-3
1	Idle
	windows-7-pro-x86_64
1	Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	nest-clang-analyzer	3 days 21 hr (#8)	N/A	4 min 6 sec
●	☀	nest-matrix-10kcollaps	5 days 8 hr (#9)	N/A	15 min
●	☀				
●	☀				

W	Description	%
☀	Test Result: 0 tests failing out of a total of 1,184 tests.	100
☀	Build stability: No recent builds failed.	100

Jenkins-based CI infrastructure for NEST was created during GSoC 2011 and helps with *automated* regular testing of new patches and timely reporting of identified problems (failed tests and broken builds). This way, issues are discovered within minutes and can be fixed immediately with much lower effort.

References

[1] Gregory Wilson (2006) *American Scientist*.doi:10.1511/2006.1.5.

[2] Gewaltig & Diesmann (2007) NEST Neural Simulation Tool. *Scholarpedia* 2(4), 1430.

Acknowledgments: This work was partly supported by the Helmholtz Alliance on Systems Biology, the Next-Generation Supercomputer Project of MEXT, EU Grant 269921 (BrainScaleS), and Google's Summer Of Code. We thank the members of the NEST Initiative for valuable discussions and input.

mindzoo.de/conf/inm2012.html