FACETS

FP6-2004-IST-FETPI 15879
Fast Analog Computing with Emergent Transient States

Deliverable D8-6

Towards generic tools for computational neuroscience: a perspective

Report Version: 1.0

Report Preparation: Théodore Papadopoulo (INRIA), Thierry Viéville (INRIA), Andrew Davison (CNRS-UNIC)

Classification: PU

Contract Start Date: 01/09/2005 Duration: 4 Years

Project Co-ordinator: Karlheinz Meier (Heidelberg)

Partners: U Bordeaux, CNRS (Gif-sur-Yvette, Marseille), U Debrecen, TU Dresden, U Freiburg, TU Graz, U Heidelberg, EPFL Lausanne, Funetics S.a.r.l., U London, U Plymouth, INRIA, KTH Stockholm

Information Society Technologies

Project funded by the European Community

under the "Information Society Technologies' Programme

DELIVERABLES TABLE

Project Number: FP6-2004-IST-FETPI 15879

Project Acronym: FACETS

Title: Fast Analog Computing with Emergent Transient States

Del. No.	Revision	Title	Type[1]	Classification[2]	Due Date	Issue Date
D8-6	1.0	Towards generic tools for computational neuroscience: a perspective.	R	PU	2008/08/31	2008/08/31

1 *R: Report; D: Demonstrator; P: Prototype; O: Other – Specify in footnote*

2 *PU: Public*
PP: Circulation within programme participants, including the Commission Services
RE: Restricted circulation list (specify in footnote), including the Commission Services
CO: Confidential, only for members of the consortium, including the Commission Services

DELIVERABLE SUMMARY SHEET

Project Number: FP6-2004-IST-FETPI 15879

Project Acronym: FACETS

Title: Fast Analog Computing with Emergent Transient States

Deliverable N^o: D8-6

Due date: 2008/08/31

Delivery Date: 2008/08/31

Short Description:

Generic tools are those that are adaptable enough to be used by many different people, in many different circumstances, rather than specific tools developed by a single person for a specific problem.
This report summarizes three years of experience in the FACETS research project in evaluating, using and developing tools in a context of large-scale, interdisciplinary and inter-laboratory collaboration. The report addresses three areas: (i) neuronal network simulation, (ii) data analysis, and (iii) data sharing. In each of these areas, we present our view of the state-of-the-art in software tools for computational neuroscience, summarize developments within FACETS (more details are given in other FACETS reports) and in collaboration with external partners, and sketch out our hopes/expectations for developments in the near future.

Partner owning: INRIA, CNRSa

Partners contributed: INRIA, CNRSa

Made available to: Public

Contents

FACETS
Fast Analog Computing with Emergent Transient States

1. Introduction

What do we mean by “generic” tools for computational neuroscience? Tools that are adaptable enough to be used by many different people, in many different circumstances, rather than specific tools developed by a single person for a specific problem. The advantages of generic tools are that they increase productivity – the extra effort required to make a tool generic rather than specific being outweighed by the time saved by using tools developed by others – and reduce errors, since a generic tool will be used and tested by many more people than will use a specific one.

The advantages of generic tools seem obvious, and many such tools do exist in the domains of computational neuroscience and neuroinformatics, and yet many simulations of neural systems, and still more analyses of both simulated and experimental data, are still done using custom-built tools, sometimes shared with the community, but more often not. This results in a waste of researchers' time, increased likelihood of undetected errors, and increased difficulty or impossibility of reproducing the results.

The reasons for this state of affairs are various. First, given the high degree of specialisation of scientific research, it may be the case that no generic tool exists than can do the job, although it might be better in such cases to extend an existing generic tool rather than build an entirely new tool from scratch. Second, many scientists do not have confidence in software that they have not written themselves. This may in some cases be valid, although even the best programmer may introduce bugs or hidden, incorrect assumptions into code that are more likely to be caught the more people use and develop the code. This lack-of-confidence provides a challenge to developers of generic software tools, to ensure that their code is both well tested, and visibly well-tested, so that any sceptical potential user can easily follow a chain of validation to convince themselves that the software gives correct results. Third, people may not be aware that a suitable generic tool exists. Several tool databases or indexes now exist, however, that should make it much easier to find a suitable tool (SimToolDB, the INCF Software Center, the Neuroscience Database Gateway). Fourth, for an individual scientist, it undeniably takes more effort to create a generic tool that may be used by others than a tool to be used by one person. On a long-term view, this effort is likely to be repaid many times over, but under the pressure to finish a thesis or submit an article for publication, such an investment of time may not be possible. In general, however, it is, or should be, much easier to contribute to an existing generic tool than to develop a new one. Developers of such tools should therefore make it as easy as possible, without compromising quality, for new people to contribute.

In this report, we survey existing tools, summarize the developments that have taken place within FACETS, and present, based on our experience of developing generic tools in the three years so-far of FACETS, some perspectives on future developments. We divide the body of the report into three sections, covering (i) simulation software, (ii) tools for data analysis, and (iii) tools for data sharing.

2. Neuronal network simulation tools

2.1 Tools for simulating biologically-realistic[1] neural circuits

This is perhaps the most well developed area of computational neuroscience in terms of generic tools. A large number of simulation tools are available (see review in [10] (FACETS Report D5-1)), of which a smaller number are widely used in research. Problems still exist, however. In general, each simulator has its own domain-specific programming language. Since most researchers are familiar with only one simulator there is a tendency for the community to fragment according to simulator, with reduced commnication between users of different simulators and a risk of solving the same problem twice or more in different communities. Due to the difficulty in translating a model from one simulator-language to another, there is also a barrier to the re-use of previously-published models and hence to building on previous work.

2.2 Developments within FACETS

The problems outlined above were particularly important for FACETS, since the nine or so groups doing neural simulations use at least five simulators between them, and in addition the FACETS project is developing neuromorphic hardware which also requires an interface.

There are several parallel but inter-related lines of development within FACETS aimed at overcoming the above problems and at improving neural simulation tools in general, in the following areas:

- Development of individual simulators
- Development of generic interfaces for simulators

2.2.1 Development of individual simulators

Two of the groups within FACETS are actively developing simulators: the ALUF group is a major participant in the development of NEST[5], which is widely used within FACETS and elsewhere, while the INRIA group develops both the Mvaspike and Brian simulators.

NEST (NEural Simulation Tool) is a simulator for “large heterogeneous networks of point neurons or neurons with a small number of compartments. NEST is best suited for models that focus on the dynamics, size, and structure of neural systems rather than on the detailed morphological and biophysical properties of individual neurons.”

Mvaspike is “a general purpose tool aimed at modeling and simulating large, complex networks of biological neural networks. It is based on an event-based modeling and simulation strategy, targetting mainly pulse-coupled, spiking neural networks (e.g. integrate-and-fire neurons, other spiking point neurons).” Related to Mvaspike is the ENAS (Event Neural Assembly Simulation) package, a set of classes designed as a plug-in for existing simulators or to enable neural simulations on general computation platforms such as Scilab or Python.

The motivation behind the Brian simulator is that a simulator should not only save the time of processors, but also the time of scientists. Brian is thus easy to learn and use, highly flexible and easily extensible. The Brian package itself and simulations using it are all written in Python, thus allowing interoperability with other existing tools. Brian has a system for physical quantities with units built in, catching errors based on using incorrect units, and ensures that simulated models are physically meaningful. Furthermore, Brian allows to define the neuron model “explicitly” i.e. directly from its equation. This is a major feature, allowing to introduce a symbolic declarative mechanism and ease the developments of small scale models.

In addition to the above simulators, for which FACETS members are core developers, there has been strong collaboration between members of FACETS and the principal developers of the widely-used NEURON[2] simulator for detailed biophysical neuron and network models, in adding Python support to the simulator.

[1]Note that we use the term “biologically-realistic” here more with relation to intent than to implementation, since all models are simplified to some extent, and one person’s realistic is another’s over-simplified.

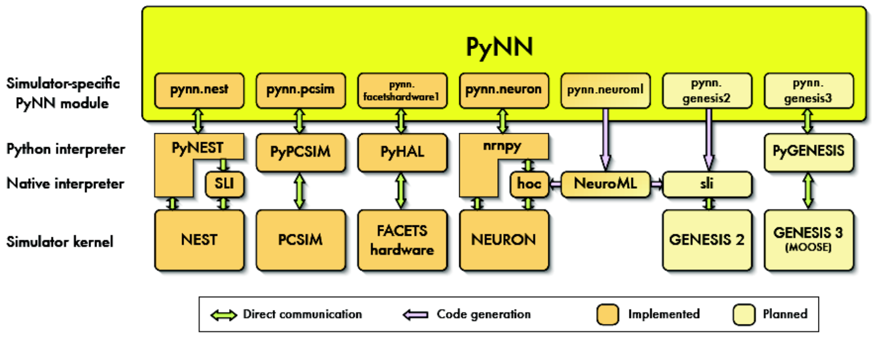


Figure 2.1. PyNN architecture

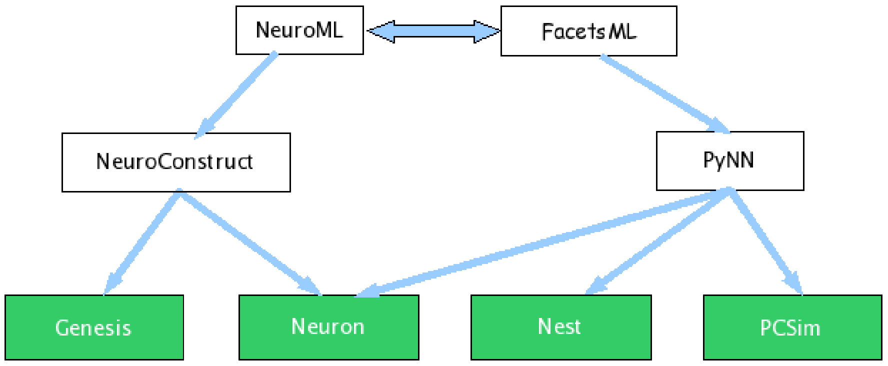


Figure 2.2. Simulator interoperability via PyNN and FacetsML

2.2.2 Generic simulator interfaces

To address the problems in supporting multiple simulators, each with its own programming language, within a single research consortium, FACETS has developed PyNN, a Python package for simulator-independent specification of neuronal network models. PyNN allows a researcher to write the code for a model once, using the PyNN API, and then run it without modification on any simulator that PyNN supports (currently NEURON, NEST, PCSIM and Brian, with support for MOOSE under development). PyNN also facilitates porting of code written for one Python-supporting simulator to another, since native simulator code can be mixed with and gradually replaced by PyNN code, allowing testing at each stage in the translation rather than only at the end. The overall architecture of PyNN is schematized in Figure 2.1.

As a step towards interoperability of PyNN with NeuroML (another, complementary approach to simulator-independent model specification), and for integration with the FACETS Knowledge Base (see below) FACETS has also developed the FacetsML XML dialect, a declarative equivalent to PyNN. FacetsML can be easily translated to PyNN with an XSLT transformation, as schematized in Figure 2.2.

The goal of this alternative specification is two-fold. At the theoretical level, it allows to specify as much as possible the problem knowledge in a static way, which is known to be the best way to perennialise the data. It also adds constraints to make things “as simple as possible” which is crucial in an multi-disciplinary context. Finally, it allows the use of a lot of existing tools to manage the logical-structure, e.g., validation, indexing, translation between dialects, etc... At a more pragmatical level, this allows PyNN to be interfaced with existing tools and in particular to use graphical editors to edit the logical-structures. The NeuroConstruct tool[6], developed outside FACETS, is now used in synergy with FACETS tools thanks to this interoperability.

Recent developments in PyNN and FacetsML are summarized in FACETS Report D8-3.

2.2.3 In-silico simulation at the unit and network level

One of the major goals of the FACETS project is to lay a theoretical and experimental foundation for the practical realisation of novel computing hardware that which exploits the concepts experimentally observed in the brain.

Two series of neuromorphic hardware based on custom, analog VLSI circuits are being developed, one series aimed at real-time, small-scale, adaptive neuronal networks, using conductance-based models (Hodgkin-Huxley formalism), the other at faster-than-real-time, large-scale adaptive neuronal networks using integrate-and-fire models.

For the second, large-scale hardware series, the low-level software layer allows for a direct interface with PyNN, allowing the same simulation script to be run directly in both software (e.g. NEURON, NEST) and hardware (within hardware limitations on neuron type and network size). This makes the ongoing effort towards precise convergence between hardware behavior and numerical simulation much easier.

More details on the large-scale hardware system and on the PyNN integration may be found in FACETS Report M7-3.

2.3 Perspectives

Although we have no statistics to confirm this impression, it does seem to be the case that the use of general-purpose neuroscience simulators continues to grow at the expense of custom, from-the-ground-up code.

Further, the topic of interoperability of simulators attracts increasing attention[3, 1], and is moving from being a desire to a reality, due to the NeuroML standards, the general trend for simulators to adopt the Python programming language as an optional or principal interface, the increased use of modular approaches to the design of new simulators and the redesign of existing ones (e.g. the GENESIS 3 project), and specific software to support interoperability such as NeuroConstruct, PyNN, and MUSIC.

The benefits to be expected from such developments include greater productivity and credibility for the field of computational neuroscience, as it becomes easier to validate, re-use and build upon computational models of neuronal systems. By reducing the amount of effort that must be expended at the level of individual simulations, this may also lead to increased attention being paid to the next level: managing the entire lifecycle of a simulation project including validation, analysis, storage, communication and reproducibility of simulation results.

3. Data analysis tools

3.1 Generic tools for data analysis

There are of course many very general tools for data analysis (Matlab, R, etc.), used throughout science and engineering. In neuroscience specifically, a number of Matlab toolboxes are available, e.g. the FIND (partly developed within FACETS, see below) and PANDORA toolboxes.

For Python, there are several packages that provide numeric and scientific functionality, notably numpy/scipy, PyGSL and ScientificPython, but, as far as we are aware, only one neuroscience-specific tool, OpenElectrophy. OpenElectrophy is a project that aims at simplifying data and analysis sharing for intra- and extra-cellular recordings. In OpenElectrophy, absolutely everything – from raw data to analysis results – is stored in a database rather than in files and folders. It has an intuitive graphic user interface for browsing data, managing recordings, plotting, organizing, and performing first order analyses (time frequency, spike detection, spike rate, ...). It has an API to write scripts for more specific analyses. The OpenElectrophy project is not developed within FACETS, but its website and code repository is hosted on the FACETS-run NeuralEnsemble site.

3.2 Developments within FACETS

The FIND (Finding Information in Neural Data) toolbox for multiple-neuron recordings and network simulations is a project of the BCCN, Freiburg, one of the participating labs in FACETS. The motivation for FIND is given as follows:

> In parallel to the tremendous technical progress in data acquisition (e.g. large number of simultaneous electrode recordings), there is a growing need for new computational tools to analyze and interpret the resulting large data flow from experiments and simulations. Indeed, progressing from single-neuron to network physiology implies a non-linear increase in complexity of data analysis. While there is undeniable progress in novel analysis methods, implementations are difficult to reproduce based on literature or are hidden in (ill-documented, in-house) software collections.
>
> We are developing FIND to address the urgent need of an unified, well-documented interface, following suggested best development practices, to various analysis tools. FIND [...] will be shared to the community as an open-source analysis toolbox for electrophysiological recordings and network simulation environments.This platform-independent, MATLAB based - toolbox can be used to analyze neurophysiological data from single- and multiple-electrode recordings by providing a set of standard and more advanced analysis and visualization methods.
>
> FIND website

A report on FIND appeared as FACETS Report D3-2 and is in press in Neural Networks [9].

For Python, several of the groups in FACETS have collaborated to develop the NeuroTools package, a set of tools for the management, storage and analysis of data from neuroscience simulations and experiments. It is intended to work closely with the PyNN package (see above), but is not dependent on it: either package can be used independently.

3.3 Perspectives

Data analysis is an area in which Matlab (The Mathworks, Inc) has a huge presence, and has an enormous number of add-on toolboxes, both commercial and free. The development of the FIND and PANDORA toolboxes for neural data analysis is a huge improvement over each investigator writing their own code.

Matlab has a number of disadvantages for use in a scientific environment with distributed computation, in particular being a commercial, non-open-source product and a rather inelegant approach to object-oriented programming. Again, we lack statistics, but there is no lack of anecdotal accounts on the internet of people transitioning from Matlab to Python.

Where Python lags behind Matlab is in discipline-specific toolboxes. In neuroscience, FACETS is attempting to remedy this with the NeuroTools package, as is the OpenElectrophy project, but we are not aware of other efforts in this area. One possible future avenue would be the development of a Python version of the FIND toolbox.

4. Data sharing tools

4.1 Data sharing in neuroscience

Data sharing in neuroscience is a very active topic, with journal articles [11, 7, 4, 12, 8], an editorial in Nature Neuroscience (`http://www.nature.com/neuro/journal/v10/n8/full/nn0807-931.html`), conference workshops (`http://www.cosyne.org/wiki/Cosyne08_Data_sharing_and_data_analysis_challenges_in_neuroscience`), and several online databases, for example:

- the CRCNS (Collaborative Research in Computational Neuroscoemce) data sharing website. `http://www.crcns.org`
- the Neurodatabase.org Data Server. `http://neurodatabase.org`
- the Cell Centered Database. `http://ccdb.ucsd.edu`
- the INCF Japan Node. `http://www.neuroinf.jp`

See the Neuroscience Database Gateway for a more extensive list. Of considerable potential interest is the ongoing CARMEN research project (`http://www.carmen.org.uk`).

Most of the publicly-accessible websites are most suitable for making datasets available to the public, and are not suitable for sharing unpublished, raw data between members of a collaboration. Some of them do, however, make available the source code used to run the data repository/portal, for example Neurodatabase.org and the XooNIps (`http://xoonips.sourceforge.jp`) system used as the platform for the INCF Japan Node.

Apart from website/portal tools, we are aware of only one publicly-available, neuroscience-oriented system for building databases: Neurosys. This is a Java-based tool for creating user-configurable, XML-based databases, and supports distributed collaboration.

4.2 The FACETS Knowledge Base

The Facets Knowledge Base (FKB) was conceived as a federated, distributed database system (Fig. 4.1) for sharing experimental and simulated data, annotated with the necessary metadata to make the raw data meaningful, between the groups within FACETS.

Rather than use an off-the-shelf, neuroscience-specific, all-in-one solution such as Neurosys (see above), it was felt that a system composed of more generic tools for distributed data storage together with a customised web interface would be simpler and more flexible in meeting the needs of the FACETS research groups.

The data storage layer of the Facets Knowledge Base (FKB) is based on federated Storage Resource Broker (SRB; http://www.sdsc.edu/srb) servers. SRB is a data server which allows efficient file storage, with metadata. Once a file has been uploaded on one SRB node of the FKB, it becomes available to all the SRB nodes of the FKB.

In order to access the FKB, several options are available:

- a graphical client (written in Java)
- a Python API, allowing data to be loaded/saved directly from within any Python software
- an online web interface with both a general query interface and several specific interfaces for particular uses (see below).

In order to standardize data formats and hence share data in an efficient way, a subset of the HDF5 (http://hdf.ncsa.uiuc.edu specification is used as the format for raw-data storage. More precisely, HDF5 can store datasets, a multidimensional array of data elements. A group is a structure for organizing objects in an HDF5 file. Using this basic object, one can create and store almost any kind of scientific data structure, such as images, arrays of vectors, and structured and unstructured grids. It was created to address the data management needs of scientists and engineers working in high performance, data intensive computing environments: storage and I/O efficiency is optimized, including on parallel computing systems.

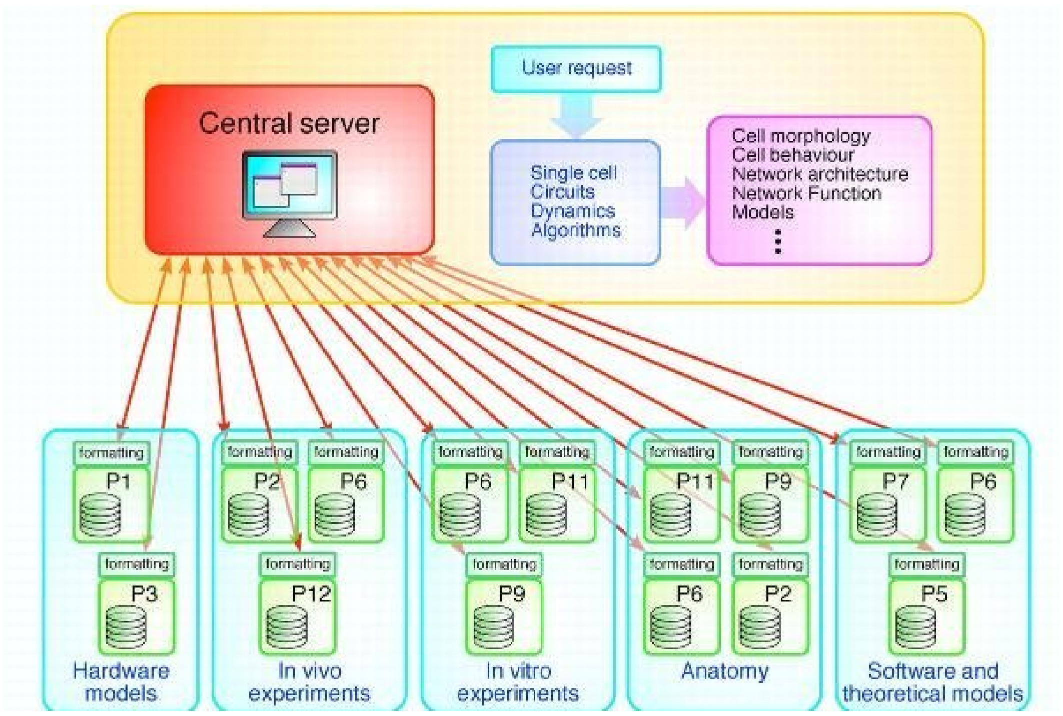
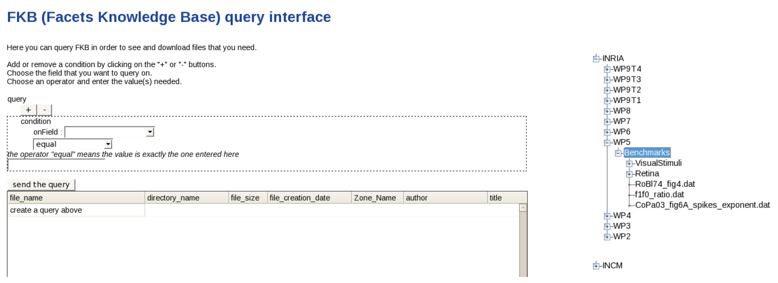


Figure 4.1. Conceptual diagram for the FACETS Knowledge Base


FKB (Facets Knowledge Base) query interface

Here you can query FKB in order to see and download files that you need.

Add or remove a condition by clicking on the "+" or "-" buttons.
Choose the field that you want to query on.
Choose an operator and enter the value(s) needed.

query
+ -
condition
onField :
equal
the operator "equal" means the value is exactly the one entered here

send the query

file_name	directory_name	file_size	file_creation_date	Zone_Name	author	title
create a query above						

```
-INRIA
   -WP9T4
   -WP9T3
   -WP9T2
   -WP9T1
   -WP8
   -WP7
   -WP6
   -WP5
      -Benchmarks
         -VisualStimuli
         -Retina
         -RoBl74_fig4.dat
         -f1f0_ratio.dat
         -CoPa03_fig6A_spikes_exponent.dat
   -WP4
   -WP3
   -WP2

-INCM
```

Figure 4.2. General query interface to the FKB (left) and repository tree (right)

In order to share meta-data in an efficient way, XML-based logical-structures are used, partially inspired by BrainML (http://brainml.org), a set of standards and practices for using XML to facilitate information exchange between user application software and neuroscience data repositories, and NeuroML (http://www.neuroml.org), an ongoing set of efforts to specify data, structures and models for software relating to neuroscience modeling.

4.3 Web interfaces to the FKB

A key aspect of the FACETS developments is that each tool integrated into the FKB has been always proposed with a web interface in addition to direct file access via SRB. This integration effort has several advantages:

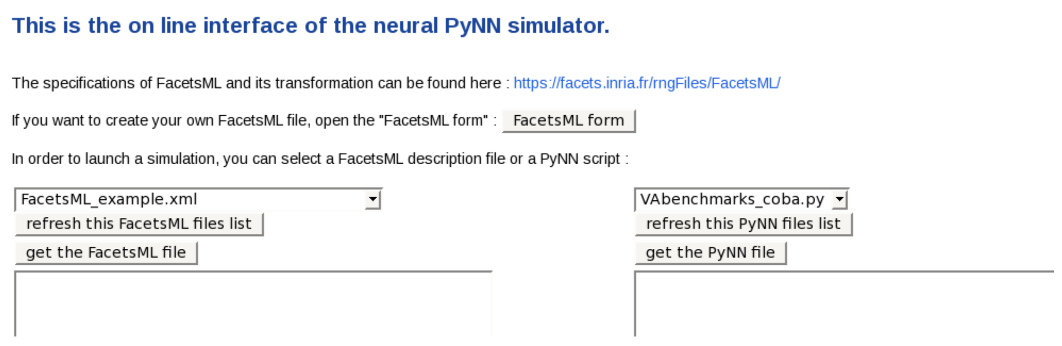
- the functionality can be used everywhere without any installation except a standard web browser and allows scientists to analyze, study, compare and benchmark the different tools, with minimal time cost.
- the user interface is standard and very simple, a simple web form, including for very complicated modules with a large, structured parameter hierarchy.

The web interfaces can also be used to generate configuration files that can be used to run software locally.

Available web interfaces include the following:

Ubiquitous data and resources repository

In Figure 4.2, the query interface is shown in the left view and a fragment of the peer-to-peer repository tree in the right view. This voluntarily minimal interface allows scientists of a community to share their data and resources.

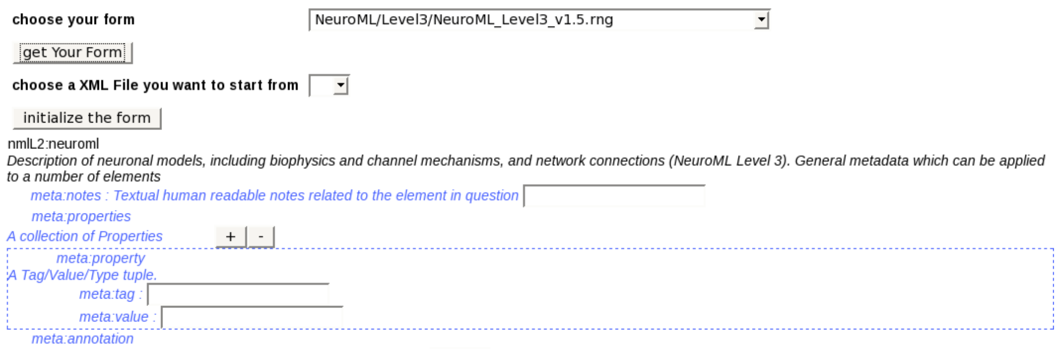

This is the on line interface of the neural PyNN simulator.

The specifications of FacetsML and its transformation can be found here : https://facets.inria.fr/rngFiles/FacetsML/

If you want to create your own FacetsML file, open the "FacetsML form" : FacetsML form

In order to launch a simulation, you can select a FacetsML description file or a PyNN script :

Figure 4.3. Online FacetsML–PyNN translator


choose your form
NeuroML/Level3/NeuroML_Level3_v1.5.rng
get Your Form
choose a XML File you want to start from
initialize the form

nmlL2:neuroml
Description of neuronal models, including biophysics and channel mechanisms, and network connections (NeuroML Level 3). General metadata which can be applied to a number of elements
meta:notes : Textual human readable notes related to the element in question
meta:properties
A collection of Properties
meta:property
A Tag/Value/Type tuple.
meta:tag :
meta:value :
meta:annotation

Figure 4.4. NeuroML logical-structure validating editor

It is based on SRB and HDF5 and has been deployed, parameterized and validated for FACETS members and beyond.

FacetsML–PyNN translator

The FacetsML–PyNN translator (Fig. 4.3) allows network simulation specifications in FacetsML format to be loaded from/saved to the FKB, to be edited, validated and translated to Python code. The interface is designed to support both remote server execution and local use of the code output. It is also a didactic tool, allowing to check and visualize the defined logical-structures in both target languages.

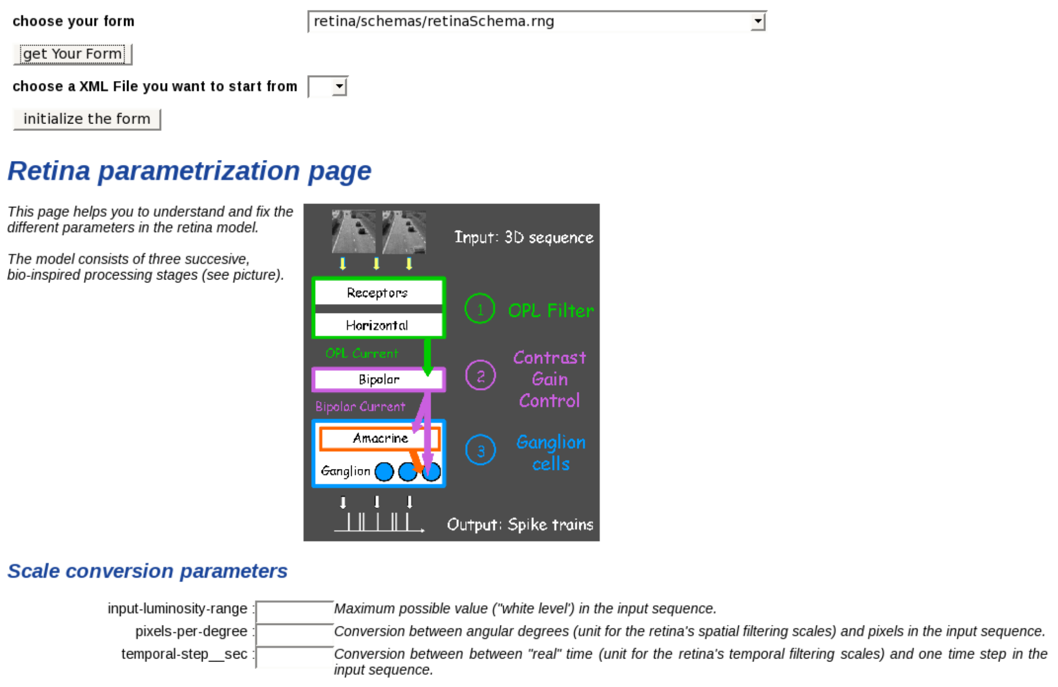
NeuroML logical-structure validating editor

This interface (Fig. 4.4) allows editing and validation of NeuroML data and resources. This corresponds to specifications which have not been developed in FACETS but by the NeuroML group. Integration with NeuroML is one of the goals of FACETS developments. The starting point for the FACETS common data model for neuroscience simulations (Report D23), now expressed in PyNN and in FacetsML, was the NeuroML specifications, and feedback from FACETS to the NeuroML developers has influenced more recent versions of the specifications.

VirtualRetina on-line configuration interface

This interface (Fig. 4.5) is intended to help with the rather large complexity of VirtualRetina simulation software parametrization. In order to run a simulation, heterogeneous parameters such as the input image-sequence, the retina topography, spatio-temporal analog filters, critical non-linear gain control parameters, miscellaneous processing and visualization parameters are to be given. This is mandatory to make good use of the underlying software. Using all software functionality would have been intractable without the help of this interface.

Here, providing default values and pre-defined sets of parameters allows to reduce the number of inputs. Using an adaptive interface (hiding unused values in a given processing mode, proposing closed-list of items, etc...) allows to quickly browse among the parameter hierarchy. Furthermore, the logical-structure is “validated” in the sense that parameter ranges, types and relations are either de-facto bounded by the web-interface or checked after input.



choose your form: retina/schemas/retinaSchema.rng

get Your Form

choose a XML File you want to start from

initialize the form

Retina parametrization page

This page helps you to understand and fix the different parameters in the retina model.

The model consists of three succesive, bio-inspired processing stages (see picture).

Scale conversion parameters

input-luminosity-range :	Maximum possible value ("white level") in the input sequence.
pixels-per-degree :	Conversion between angular degrees (unit for the retina's spatial filtering scales) and pixels in the input sequence.
temporal-step__sec :	Conversion between between "real" time (unit for the retina's temporal filtering scales) and one time step in the input sequence.

Figure 4.5. VirtualRetina on-line configuration interface

Real-time hardware on-line configuration interface

As for the previous software, the PAX real-time hardware specification requires a rather large set of parameters, and is subject to a similar complexity. Using the same methodology, it has been possible to reduce the complexity of interacting with this hardware. This web-interface (Fig. 4.6) is interfaced directly with the hardware itself, via a semi-automatic queuing mechanism, to give full operational behavior.

Building other web interfaces

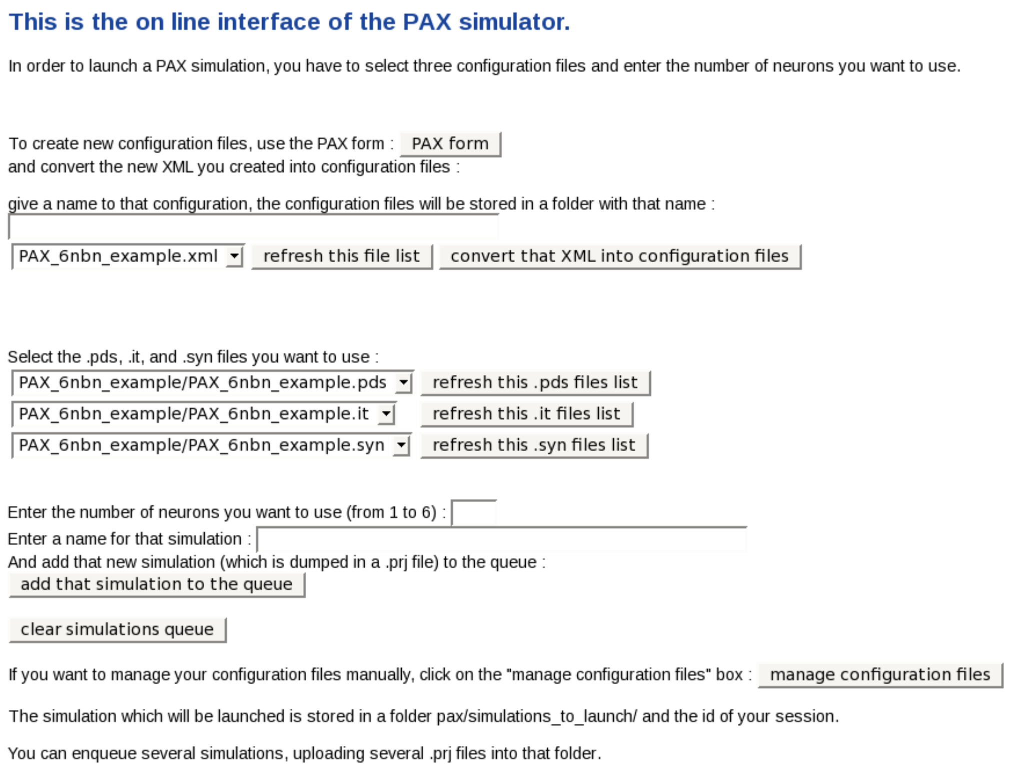
The crucial technological choice here was to entirely base the implementation on open-source, fully supported software tools:

- SRB (http://www.sdsc.edu/srb) to support shared data and resources distributed across multiple organizations and heterogeneous storage systems.
- HDF (http://www.hdfgroup.org) to organize, store, discover, access, analyze, share, and preserve data locally.
- OASIS (http://www.oasis-open.org) RelaxNG validator and Saxon's XSLT processor (http://www.saxonica.com) for logical-structure.
- Dojo (http://dojotoolkit.org), Ajax Javascript web user interface, with Tomcat (http://tomcat.apache.org) for implementation of server side mechanisms.

Evaluating, choosing and deploying all these technologies was initially time-consuming, but once done the time-cost to develop yet another web-service of this kind is minimal, the existing code being fully documented (Fig. 4.7).

4.4 Perspectives

There are two approaches to sharing data. One is to upload your data to a public repository. This has the advantage of simplicity: someone else takes care of setting up and managing servers, backup, etc. The disadvantage is that the interface and data-access permissions are controlled by the host website. The second approach is to host your data locally. This has the advantages of increased control over data-access and over how the data is presented, and of allowing integration with systems for managing unpublished data. The main disadvantage is that there is no standard way to build such a system, which adds significant development costs to the costs already associated with installing and running a data server.



This is the on line interface of the PAX simulator.

In order to launch a PAX simulation, you have to select three configuration files and enter the number of neurons you want to use.

To create new configuration files, use the PAX form : [PAX form]
and convert the new XML you created into configuration files :

give a name to that configuration, the configuration files will be stored in a folder with that name :

PAX_6nbn_example.xml [refresh this file list] [convert that XML into configuration files]

Select the .pds, .it, and .syn files you want to use :
PAX_6nbn_example/PAX_6nbn_example.pds [refresh this .pds files list]
PAX_6nbn_example/PAX_6nbn_example.it [refresh this .it files list]
PAX_6nbn_example/PAX_6nbn_example.syn [refresh this .syn files list]

Enter the number of neurons you want to use (from 1 to 6) :
Enter a name for that simulation :
And add that new simulation (which is dumped in a .prj file) to the queue :
[add that simulation to the queue]

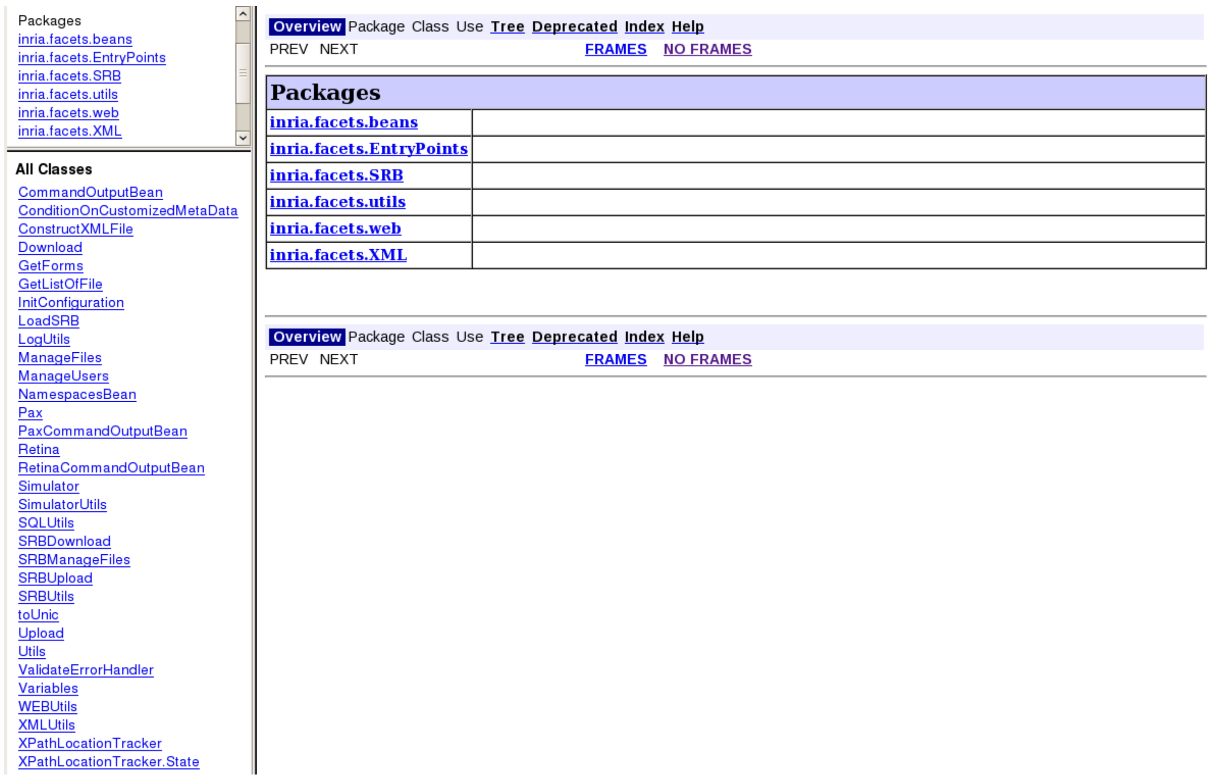
[clear simulations queue]

If you want to manage your configuration files manually, click on the "manage configuration files" box : [manage configuration files]

The simulation which will be launched is stored in a folder pax/simulations_to_launch/ and the id of your session.

You can enqueue several simulations, uploading several .prj files into that folder.

Figure 4.6. Real-time hardware on-line configuration interface



Packages
inria.facets.beans
inria.facets.EntryPoints
inria.facets.SRB
inria.facets.utils
inria.facets.web
inria.facets.XML

All Classes
CommandOutputBean
ConditionOnCustomizedMetaData
ConstructXMLFile
Download
GetForms
GetListOfFile
InitConfiguration
LoadSRB
LogUtils
ManageFiles
ManageUsers
NamespacesBean
Pax
PaxCommandOutputBean
Retina
RetinaCommandOutputBean
Simulator
SimulatorUtils
SQLUtils
SRBDownload
SRBManageFiles
SRBUpload
SRBUtils
toUnic
Upload
Utils
ValidateErrorHandler
Variables
WEBUtils
XMLUtils
XPathLocationTracker
XPathLocationTracker.State

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Packages	
inria.facets.beans	
inria.facets.EntryPoints	
inria.facets.SRB	
inria.facets.utils	
inria.facets.web	
inria.facets.XML	

Overview Package Class Use Tree Deprecated Index Help
PREV NEXT FRAMES NO FRAMES

Figure 4.7. Screenshot of the Java class reference for the FKB infrastructure

While it is to be hoped that more and more investigators will make use of the public neuroscience data repositories, there is also a need for "turn-key", easily configurable systems for local databasing, seamlessly managing both private and public data. Neurosys seems to be one such system, although we have not evaluated it in detail. It might also be of interest for FACETS to continue development of the FKB system so as to create a generic package that could be used by other labs and research consortiums.

5. Conclusion

"*Increasingly, the real limit on what computational scientists can accomplish is how quickly and reliably they can translate their ideas into working code*" Gregory V. Wilson, Where's the Real Bottleneck in Scientific Computing? Neural Ensemble

For computational neuroscience to realize its potential, a large step forward in the productivity of software development is necessary. We think this step will come about in part from training researchers in modern software development methods, and in part from reduced duplication and waste of effort, which in turn relies on the development and adoption of more-or-less generic solutions to problems in simulation, data analysis, and data sharing.

In this report we have attempted to illustrate the state of the art in generic tools in each of these areas, and to highlight recent progress made both within and external to the FACETS consortium. We have also summarized our hopes/expectations for developments in the near future.

Going beyond this report, several researchers within FACETS have begun the NeuralEnsemble initiative (http://neuralensemble.org), a multilateral effort to coordinate and organize neuroscience software development efforts into a larger "meta-simulator" software system. Such a system, which should be fully-open to all researchers, seems to us a necessary condition for making progress in understanding the complexities of brain-like computing.

About the FACETS Project

The goal of the FACETS (Fast Analog Computing with Emergent Transient States) project is to create a theoretical and experimental foundation for the realisation of novel computing paradigms which exploit the concepts experimentally observed in biological nervous systems. The continuous interaction and scientific exchange between biological experiments, computer modelling and hardware emulations within the project provides a unique research infrastructure that will in turn provide an improved insight into the computing principles of the brain. This insight may potentially contribute to an improved understanding of mental disorders in the human brain and help to develop remedies.

FACETS is an international and interdisciplinary research project funded by the European Commission in the framework of the Information Society Technologies (IST) programme. Within this programme, Future Emerging Technologies (FET) is defined as the 'programme nursery' of IST, in which basic research on novel concepts of information processing beyond the classical Turing approach are studied. FACETS is an integrated project within the biologically inspired information systems branch.

http://facets-project.org

References

[1] R C Cannon, M O Gewaltig, P Gleeson, H Cornelis, M L Hines, F W Howell, J R Stiles, S Wills, and E de Schutter. Interoperability of neuroscience modeling software: current status and future directions. *Neuroinformatics*, 5(2):127–138, 2007.

[2] Nicholas T. Carnevale and Michael L. Hines. *The NEURON Book*. Cambridge University Press, 2006.

[3] Mikael Djurfeldt and Anders Lansner. Workshop report: 1st incf workshop on large-scale modeling of the nervous system. *Nature Precedings*, ¡http://dx.doi.org/10.1038/npre.2007.262.1¿, 2007.

[4] Daniel Gardner, Arthur W. Toga, Giorgio A. Ascoli, Jackson Beatty, James F. Brinkley, Anders M. Dale, Peter T. Fox, Esther P. Gardner, John S. George, Nigel Goddard, Kristen M. Harris, Edward H. Herskovits, Michael Hines, Gwen A. Jacobs, Russell E. Jacobs andEdward G. Jones, David N. Kennedy, Daniel Y. Kimberg, John C. Mazziotta, Perry Miller, Susumu Mori, David C. Mountain, Allan L. Reiss, Glenn D. Rosen, David A. Rottenberg, Gordon M. Shepherd, Neil R. Smalheiser, Kenneth P. Smith, Tom Strachan, David C. Van Essen, Robert W. Williams, and Stephen T. C.Wong. Towards effective and rewarding data sharing. *Neuroinformatics*, 1(3):289–295, 2003.

[5] Marc-Oliver Gewaltig and Markus Diesmann. NEST. *Scholarpedia*, 2(4):1430, 2007.

[6] P. Gleeson, V. Steuber, and R. A. Silver. neuroConstruct: A tool for modeling networks of neurons in 3D space. *Neuron*, 54(2):219–235, April 2007.

[7] Thomas R. Insel, Nora D. Volkow, Ting-Kai Li, Jr. James F. Battey, and Story C. Landis. Neuroscience networks: Data-sharing in an information age. *PLoS Biology*, 1(1):009–011, 2003.

[8] S.H. Koslow. Sharing primary data: A threat or asset to discovery? *Nature Rev. Neurosci.*, 3(4):311–313, 2002.

[9] Ralph Meier, Ulrich Egert, Ad Aertsen, and Martin P. Nawrot. FIND – a unified framework for neural data analysis. *Neural Networks*, In Press: `http://dx.doi.org/10.1016/j.neunet.2008.06.019`, 2008.

[10] Brette R, Rudolph M, Carnevale T, Hines M, Beeman D, Bower JM, Diesmann M, Morrison A, Goodman PH, Harris FC Jr, Zirpe M, Natschläger T, Pecevski D, Ermentrout B, Djurfeldt M, Lansner A, Rochel O, Vieville T, Muller E, Davison AP, El Boustani S, and Destexhe A. Simulation of networks of spiking neurons: a review of tools and strategies. *J. Comput. Neurosci.*, 23(3):349–398, Dec 2007.

[11] Jeffrey L. Teeters, Kenneth D. Harris, K. Jarrod Millman, Bruno A. Olshausen, and Friedrich T. Sommer. Data sharing for computational neuroscience. *Neuroinformatics*, 6(1):47–55, March 2008.

[12] The OECD Working Group on Neuroinformatics, P. Eckersley, G.F. Egan, S-I Amari, F. Beltrame, R. Bennett, J.G. Bjaalie, T. Dalkara, E. De Schutter, C. Gonzalez, S. Grillner, A. Herz, K. P. Hoffmann, I.P. Jaaskelainen, S.H. Koslow, S-Y. Lee, L. Matthiessen, P.L. Miller, F.M.D. Silva, M. Novak, V. Ravindranath, R. Ritz, U. Ruotsalainen, S. Subramaniam, A.W. Toga, S. Usui, J.V. Pelt, P. Verschure, D. Willshaw, A. Wrobel, and Y. Tang. Neuroscience data and tool sharing. a legal and policy framework for neuroinformatics. *Neuroinformatics*, 1:149–165, 2003.