DELIVERABLE REPORT

Deliverable 7.2
Prototype Software for Testing

Grant Agreement number:	289434
Project acronym:	BioPreDyn
Project title:	From Data to Models: New Bioinformatics Methods and Tool for Data-Driven Predictive Dynamic Modeling in Biotechnological Applications
Funding Scheme:	Collaborative Project
Due date of deliverable:	month 30
Actual submission date:	31.03.2014
Start date of project:	01.12.2011
Duration:	until 31.03.2015
Dissemination Level:	PP (restricted to program participants)
Organisation name of lead contractor for this deliverable:	CSM P9

Introduction

One of BioPreDyn's main objectives is the development of a software platform integrating several tools for running dry experiments on numerical models. More specifically, this platform should be an implementation of the systems biology model building cycle, as defined in deliverable 7.1 ("Specifications for Software Functionality & GUI"). Deliverable 7.2 is the prototype, or proof of concept, for this integrative software platform.

As such, it should include as many of the following elements as possible (from D7.1):

- A **work flow** editor that allows user to edit a file describing the [systems biology modeling] cycle steps that should be applied to the model as part of the model life cycle;
- A **collection of libraries and tools**, developed by the consortium members; this should be capable of performing one or more of the cycle steps;
- A **parser** capable of reading a work flow, describing the tasks, and then passing them to the simulation engine;
- A **simulation engine** capable of calling the tools (from the collection of tools) necessary for each task of the work flow.

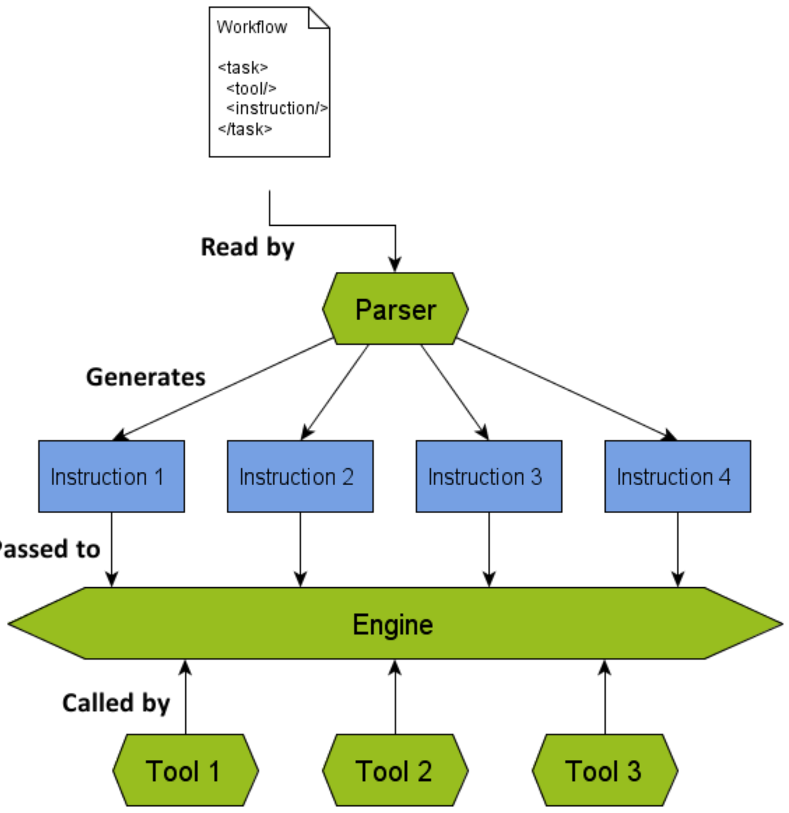


Figure 1: Proposal of architecture after the requirements expressed in D7.1.

In addition to those high level specifications, a pre-study[1] based on cases provided by several consortium members lead to the identification of the following technical requirements:

- The developed tool should be as user-friendly as possible, and therefore requires a **graphical user interface**;
- As an implementation of the model building cycle, the developed software should offer **modularity** when building a numerical experiment;
- **Standards** for data, model or simulation formats should be used in order to make the interfacing with other tools easier.

Results

With those requirements in mind, CSM developed a prototype written in Python[2], and made available (through a Subversion[3] server) to all consortium members.

Languages

The following languages are currently used in deliverable 7.2:

- As a work flow description language: **Simulation Experiment Description Markup Language[4] (SEDML)**
- For model representation: **Systems Biology Markup Language[5] (SBML)**
- As a numerical result description language: **Numerical Markup Language[6] (NuML)**

As a consequence, tools to be integrated in the software suite should be compatible with those technologies, when required.

Architecture

Parser

As SEDML is the chosen language for describing numerical experiments in BioPreDyn, the prototype uses **libSEDML**[7] as a parser for SEDML files. This library is currently integrated as a built-in dependency in the BioPreDyn prototype.

As part of the development effort on BioPreDyn, CSM provides feedback and reports bugs to the developers of libSEDML.

Engine

Developed in Python, the engine of the prototype is divided into five components, as shown in Figure 2.

1 See attached document IntegratedSoftwareSuiteRequirementsGuide.pdf
2 https://www.python.org/
3 http://subversion.apache.org/
4 http://sed-ml.org/
5 http://sbml.org/Main_Page
6 http://code.google.com/p/numl/
7 https://github.com/fbergmann/libSEDML

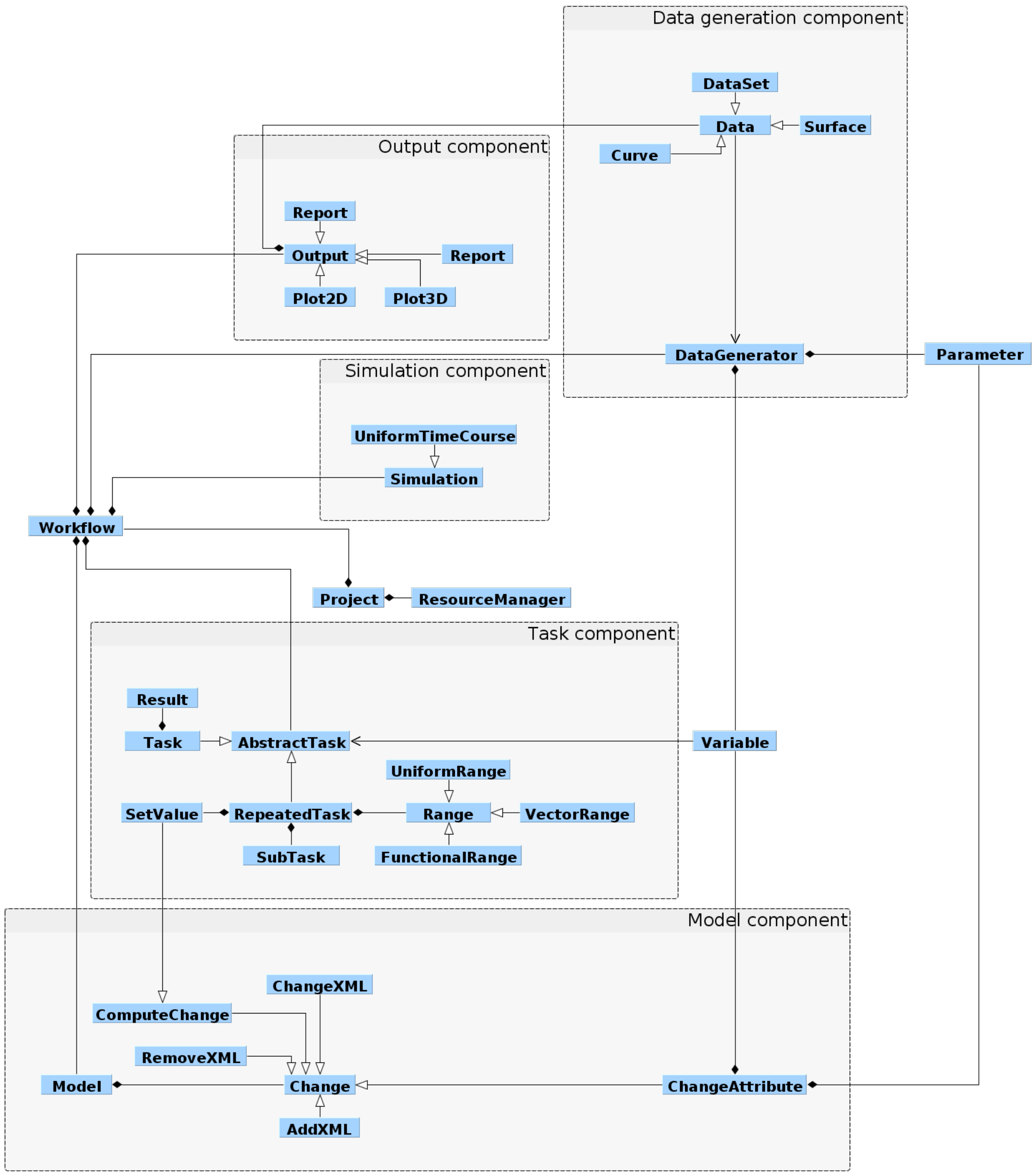


Figure 2: Simplified UML class diagram of the BioPreDyn engine

Each component handles a specific part of the numerical experiment (as defined in SEDML):

- **Model component**: for SBML model manipulation and modifications. In particular, changes in the model (such as changing the initial value of a parameter, or removing or adding elements) are implemented in this component.
- **Simulation component**: handles the different types of simulations to be executed in the current numerical experiment. In SEDML, a simulation is defined as an algorithm along with the set of parameters it requires.

- **Task component**: running a task means executing a simulation with a specific model. This component handles all the aspects of the execution of a task; this is where external simulation engines are called. Results of simulations are also part of this component.
- **Data generation component**: for post-processing the result of simulations, and preparing them for output (graphical or not).
- **Output component**: processes the data generated in the data generation component. Data can be displayed as a 2D or 3D plot, or exported as a NuML or CSV file.

Tools

The BioPreDyn prototype relies on several tools for writing or reading specific file formats, running simulations, displaying results, etc, as pictured in Figure 3:

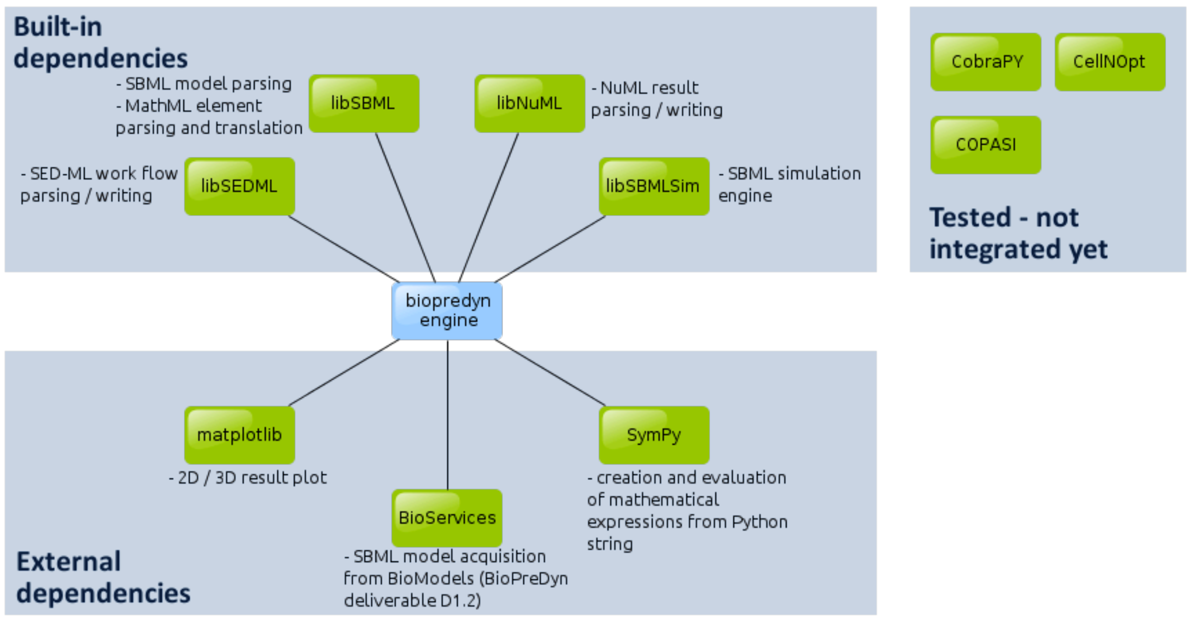

Built-in dependencies
- SBML model parsing
- MathML element parsing and translation
libSBML
libNuML
- NuML result parsing / writing
- SED-ML work flow parsing / writing
libSEDML
libSBMLSim
- SBML simulation engine
CobraPY
CellNOpt
COPASI
Tested - not integrated yet
biopredyn engine
matplotlib
- 2D / 3D result plot
SymPy
- creation and evaluation of mathematical expressions from Python string
BioServices
- SBML model acquisition from BioModels (BioPreDyn deliverable D1.2)
External dependencies

Figure 3: BioPreDyn engine dependencies

Among those tools, the built-in dependencies (namely libSEDML, libSBML, libNuML and libSBMLSim) are compiled and built with the project. External dependencies have to be installed separately. Among them, BioServices is the deliverable 1.2 of the BioPreDyn project.

Usage

Installation

The prototype has been tested on the following platforms:

- Ubuntu 12.04 (32 and 64-bit)
- Windows XP, 7 (32-bit)

The installation process (along with the required dependencies) is detailed in the installation guide[8].

Execution

Once the BioPreDyn prototype is installed, it can be used to run numerical experiments encoded as SEDML files. Simply open a shell and navigate to the folder where the prototype is installed, and type the following commands:

```
cd build/install/bin
```

This folder contains a main.py file which can be used to run the prototype. To do so, one can use the following syntax:

```
python main.py <options>
```

Depending on the option(s) chosen by the user, various operations can be done. Valid options are listed below:

- `--sedml <path/to/file.xml>`: opens the input SEDML file, executes the tasks it contains and process its outputs. Graphical outputs are displayed, if any; path/to/file.xml must point to a valid SEDML file.
- `--output <path/to/output.csv>`: write the result of a numerical experiment (if any) to the input location as a CSV file. This option should be used only when the opened SEDML file contains one or more “report” elements.
- `--output <path/to/output.xml>`: identical to the previous one, except that the result is exported as a NuML file instead.
- `--csv <path/to/file.csv>`: opens the input CSV file and plot its content; path/to/file.csv must point to a valid CSV file.
- `--numl <path/to/file.xml>`: opens the input NuML file and plot its content; path/to/file.xml must point to a valid NuML file.

For instance, for running an experiment encoded in the file test_graphical_output.xml on the CSM repository, one can type:

```
python main.py --sedml
https://thecosmocompany.com/svn/repos/SVN/BioPreDyn/trunk/Prototyp
e/testing/test_graphical_output.xml
```

In case the experiment went smoothly, the windows pictured on Figure 4 should be displayed. They show the evolution of the concentrations of various components interacting in the model.

For writing the results of another experiment as a NuML file:

```
python main.py --sedml
https://thecosmocompany.com/svn/repos/SVN/BioPreDyn/trunk/Prototyp
e/testing/test_report.xml --output report.xml
```

8 See annex : PrototypeUserGuide.pdf

This command will create a file called report.xml in the folder where the command was called (in this case, in <path/to/build/dir>/install/bin).

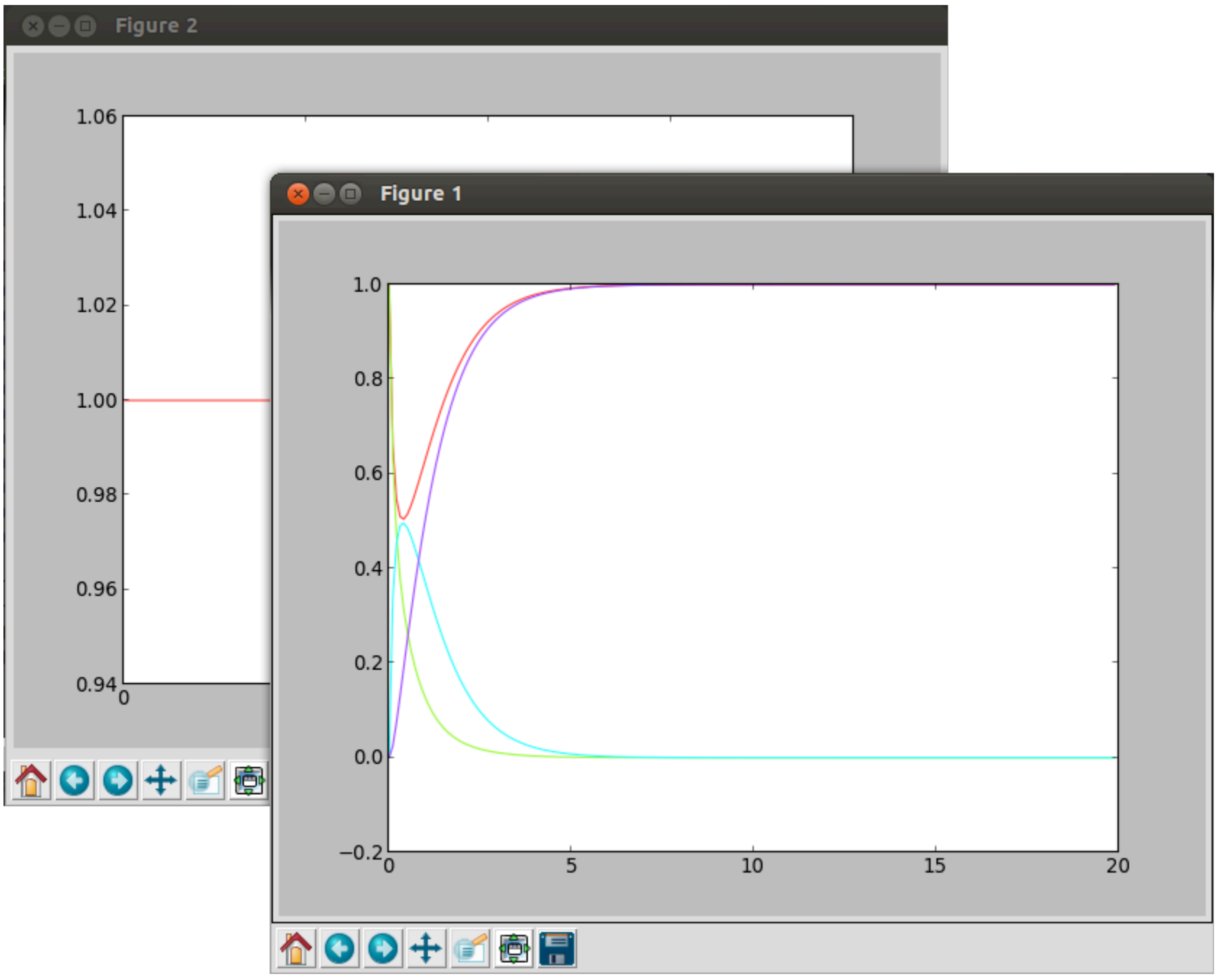


Figure 4: Displayed 2D plots after running the experiment encoded in test_graphical_output.xml SEDML file.

More information on how to use the prototype can be found in the prototype user guide[9].

Testing

Testing and bug reporting procedure

EVOLVA conducted a series of tests on the above illustrated prototype that CSM has developed for the BioPreDyn software suite. The prototype consists of a number of subroutines (tools) which need to be installed before installing the actual software engine.

Provided with a user manual prepared by CSM each of these subroutines (tools) were downloaded and installed via the command-line to a computer running Linux.

Every error observed during the installation process of these dependencies was reported through a ticket system set up by CSM on an internal web page[10]. For an

9 See annex: PrototypeUserGuide.pdf

end user, these dependencies need to be installed as smoothly as possible; therefore both corrections to software as well as the user documentation may need adjustments.

Finally, the source code for the prototype software was obtained by EVOLVA from the internal CSM web page.

Errors encountered by EVOLVA during this installation were also reported to CSM. On complete installation of all software dependencies and the prototype itself a set of tests was run. After successful termination of the test runs the prototype software without the graphical user interface is considered complete.

The next steps will be to test the prototype even further once the graphical user interface, as agreed among partners in deliverable WP 7.1, has been implemented by CSM.

Test cases

The BioPreDyn prototype is tested using a simple model of enzymatic reaction, based on the work of CWI[11]. This model was translated in SBML using COPASI and can be found on the BioPreDyn Subversion server[12].

In addition, other SBML models from BioModels[13] are used as test cases for the development of this software tool.

10 https://thecosmocompany.com/biopredyn-trac
11 http://www.ncbi.nlm.nih.gov/pubmed/19215296
12 https://thecosmocompany.com/svn/repos/SVN/BioPreDyn/trunk/Prototype/data/FEBS_copasi.xml
13 http://www.ebi.ac.uk/biomodels-main/

BioPreDyn software suite prototype - Deliverable 7.2 - User guide

Bertrand Moreau, The CoSMo company

2014

Abstract

This document is the user guide for the BioPreDyn software suite prototype (deliverable 7.2).

Table of Contents

1. Preface

1.1. Purpose of this document

This document aims at giving the user all the information he/she needs in order to set a suitable environment for the `BioPreDyn` software suite prototype (deliverable 7.2 of the `BioPreDyn` project), as long as basic instructions on how to use it.

This document does not describe the architecture of the deliverable 7.2 nor gives details about its implementation; this will be detailed respectively in the requirements guide and the design notes.

1.2. Audience

This document is primarily intended to advanced users willing to test the `BioPreDyn` software suite prototype on their own model. As a proof of concept for more advanced versions (deliverables 3.4 / 8.3), very little effort is put on the package user-friendliness; it is assumed that the reader knows how to compile and build code using GNU make, CMake, or Visual Studio.

1.3. Organization

This guide is divided into four main sections:

- **Introduction.** A short introduction to the project context, and how the current document fits in the global picture.
- **Installation.** How to install a suitable environment for working with the `BioPreDyn` software suite prototype on multiple platforms.
- **Usage.** How to use the `BioPreDyn` software suite prototype.

2. Introduction

This code is the deliverable 7.2 of the BioPreDyn project ("Prototype Software for Testing: User-friendly version of prototype software for testing in a setting for industrial applications").

For more information about the development tasks related to the BioPreDyn project (and the content of the deliverable 7.2), please visit the developer's wiki[1].

For more information about the BioPreDyn project itself, please refer to the official project website [2].

3. Installation

3.1. Ubuntu 32-bit / 64-bit (11.10 or later)

3.1.1. Dependencies

3.1.1.1. CMake

```
sudo apt-get install cmake
sudo apt-get install cmake-curses-gui
```

3.1.1.2. Subversion

```
sudo apt-get install subversion
```

3.1.1.3. Python 2.7

```
sudo apt-get install python2.7
```

3.1.1.3.1. pip

```
sudo apt-get install python-pip
```

3.1.1.3.2. Other Python dependencies

```
sudo apt-get install python-numpy
sudo apt-get install python-matplotlib
sudo apt-get install python-sympy
sudo pip install bioservices
sudo pip install lxml
sudo apt-get install python-pyside
```

3.1.1.4. SWIG 2.0

```
sudo apt-get install swig2.0
```

[1] https://thecosmocompany.com/biopredyn-trac/
[2] http://www.biopredyn.eu/

3.1.1.5. libXML2

```
sudo apt-get install libxml2 libxml2-dev
```

3.1.1.6. libzip2

```
sudo apt-get install libbz2-1.0 libbz2-dev
```

3.1.1.7. Doxygen

```
sudo apt-get install doxygen
```

3.1.1.8. Graphviz

```
sudo apt-get install graphviz graphviz-dev
```

3.1.2. BioPreDyn integrated software suite prototype

First of all the project source code must be checked-out using `Subversion`:

```
 svn co https://thecosmocompany.com/svn/repos/SVN/BioPreDyn/trunk
biopredyn
```

If required, use **dashuser-biopredyn** as a login and **Nie8eir2** as a password. When `Subversion` is done with it, open a shell, navigate to the freshly checked-out biopredyn folder and type:

```
cd biopredyn
mkdir BUILD
cd BUILD
cmake ..
make
make test
```

`CMake` options can be more finely tuned by using `ccmake` instead of `cmake`.

3.2. Windows 32 (XP, 7)

3.2.1. CMake

Download the latest stable version from the CMake download page[3] then run the installer and follow the instructions.

3.2.2. Microsoft Visual Studio 2010

Download Visual C++ 2010 Express from the Visual Studio download page[4] then run the executable and follow the instructions.

[3] http://www.cmake.org/cmake/resources/software.html
[4] http://www.visualstudio.com/en-us/downloads/download-visual-studio-vs#DownloadFamilies_4

3.2.3. Subversion

Download Apache Subversion from the VisualSVN download page[5], run the installer and follow the instructions.

3.2.4. SWIG

The latest version of swigwin can be downloaded from the SWIG download page[6]. Unzip it, then add the folder containing swig.exe to the Path environment variable.

3.2.5. Python

Download the Windows installer from the Python download page[7], run it and follow the instructions.

3.2.5.1. easy_install

Download ez_setup.py from the setuptools download page[8] then open a command prompt, navigate to the folder where ez_setup.py was downloaded and type:

```
python ez_setup.py
```

Now navigate to the Python installation folder (default C:\Python27) and type:

```
cd Tools/Sripts
python win_add2path.py
```

This will add useful Python folders to the path.

3.2.5.2. NumPy

Download the win32 installer at the NumPy download page[9] and install it.

3.2.5.3. matplotlib

Download the installer for the last version (matplotlib-X.Y.Z.win32-py2.7.exe) at the matplotlib download page[10] and install it.

3.2.5.4. Other Python dependencies

Open a command prompt and type:

```
easy_install easydev
easy_install bioservices
easy_install lxml
easy_install PySide
easy_install sympy
```

[5] http://www.visualsvn.com/downloads/
[6] http://www.swig.org/download.html
[7] http://www.python.org/download/
[8] https://bitbucket.org/pypa/setuptools/downloads
[9] http://sourceforge.net/projects/numpy/files/NumPy/1.8.1/numpy-1.8.1-win32-superpack-python2.7.exe/download
[10] https://github.com/matplotlib/matplotlib/downloads/

3.2.6. GnuWin32

The dependencies of this category must be installed in the same folder; it is done automatically for the first two ones (as win32 installers exist), it has to be done manually for the last one.

3.2.6.1. bzip2

Download the latest bzip2-X.Y.Z-setup.exe from bzip2 download page[11] and install it. Add [path/to/GnuWin32]/include and [path/to/GnuWin32]/lib to the environment Path variable.

3.2.6.2. libiconv

Download the latest libiconv-X.Y.Z-1.exe from libiconv download page[12] and install it. As it will install in the same repository than bzip2, it is not necessary to change the path.

3.2.6.3. libXML2

Download the latest version (libxml2-X.Y.Z.win32.zip) at the XMLSoft download page[13] and extract it. The resulting library folders (i.e. bin, include and bin) must be merged with the ones in GnuWin32 (see bzip2 paragraph above).

3.2.7. Doxygen

Download the latest binary package (doxygen-X.Y.Z-setup.exe) at the Doxygen download page[14] and install it.

3.2.8. Graphviz

Download the latest installer (graphviz-X.Y.msi) at the Graphviz download page[15] and execute it.

3.2.9. BioPreDyn integrated software suite prototype

First of all the project source code must be checked-out using `Subversion`:

```
 svn co https://thecosmocompany.com/svn/repos/SVN/BioPreDyn/trunk
biopredyn
```

If required, use **dashuser-biopredyn** as a login and **Nie8eir2** as a password. When `Subversion` is done with it, open a Visual Studio Command Prompt (2010), then navigate to the freshly checked-out biopredyn folder and type:

```
mkdir BUILD
cd BUILD
cmake ..
nmake
ctest
```

`CMake` options can be more finely tuned by using `ccmake` instead of `cmake`.

[11] http://sourceforge.net/projects/gnuwin32/files/bzip2/
[12] http://sourceforge.net/projects/gnuwin32/files/libiconv/1.9.2-1/
[13] http://xmlsoft.org/sources/win32/
[14] http://www.stack.nl/~dimitri/doxygen/download.html
[15] http://www.graphviz.org/Download_windows.php

4. Usage

Once the BioPreDyn prototype is installed, it can be used to run numerical experiments encoded as SEDML files. Simply open a shell or a command prompt, navigate to the folder where the prototype is installed, and type the following commands:

```
cd build/install/bin
```

This folder contains a main.py file which can be used to run the prototype. To do so, one can use the following syntax:

```
python main.py [options]
```

Depending on the option(s) chosen by the user, various operations can be done. Valid options are listed below:

- `--sedml [path/to/file.xml]`: opens the input SEDML file, executes the tasks it contains and process its outputs. Graphical outputs are displayed, if any; path/to/file.xml must point to a valid SEDML file.
- `--output [path/to/output.csv]`: write the result of a numerical experiment (if any) to the input location as a CSV file. This option should be used only when the opened SEDML file contains one or more “report” elements.
- `--output [path/to/output.xml]`: identical to the previous one, except that the result is exported as a NuML file instead.
- `--csv [path/to/file.csv]`: opens the input CSV file and plot its content; path/to/file.csv must point to a valid CSV file.
- `--numl [path/to/file.xml]`: opens the input NuML file and plot its content; path/to/file.xml must point to a valid NuML file.

Integrated software suite requirements

Bertrand Moreau, The CoSMo company

2013

Abstract

This document summarizes the requirements for the development of the BioPreDyn integrated software suite (deliverable 3.4 / 7.2 / 8.3).

Table of Contents

1. Preface

1.1. Purpose of this document

This document aims at summarizing the requirements for the development of the `BioPreDyn` integrated software suite (deliverables 3.4, 7.2 and 8.3). It lists the main specifications as listed in the official description of work, or gathered by the development team when visiting the project partners. Critical use cases are then identified using these specifications.

This document does not describe the solution chosen; this will be detailed in the design notes.

1.2. Audience

This document is primarily intended to developers willing to understand the choices made during the `BioPreDyn` software suite design stage. It is particularly useful for maintenance and modification purposes.

1.3. Organization

This guide is divided into four main sections:

- **Introduction.** A short introduction to the project context, and how the current document fits in the global picture.
- **Specifications.** Review of the high level objectives of the deliverables 3.4, 7.2 and 8.3, through the study of the project's description of work and several visits to the project's partners.
- **Use cases.** Formalization of the needs described in the *Specifications* section into use cases describing the main aspects and mechanisms to be captured by the design.
- **References.** Articles and publications related to the project.

2. Introduction

The BioPreDyn[1] project aims at standardizing the data regression process in systems biology. Nowadays, there exists a great variety of very specialized tools in this field: each one of them is capable of handling one or several steps in the systems biology model building cycle.

Deliverables 3.4, 7.2 and 8.3 of the `BioPreDyn` project consist in a software tool linking some of those tools together in a cross-platform, user-friendly framework. This document describes the requirements for such a tool.

3. Specifications

3.1. Description of work

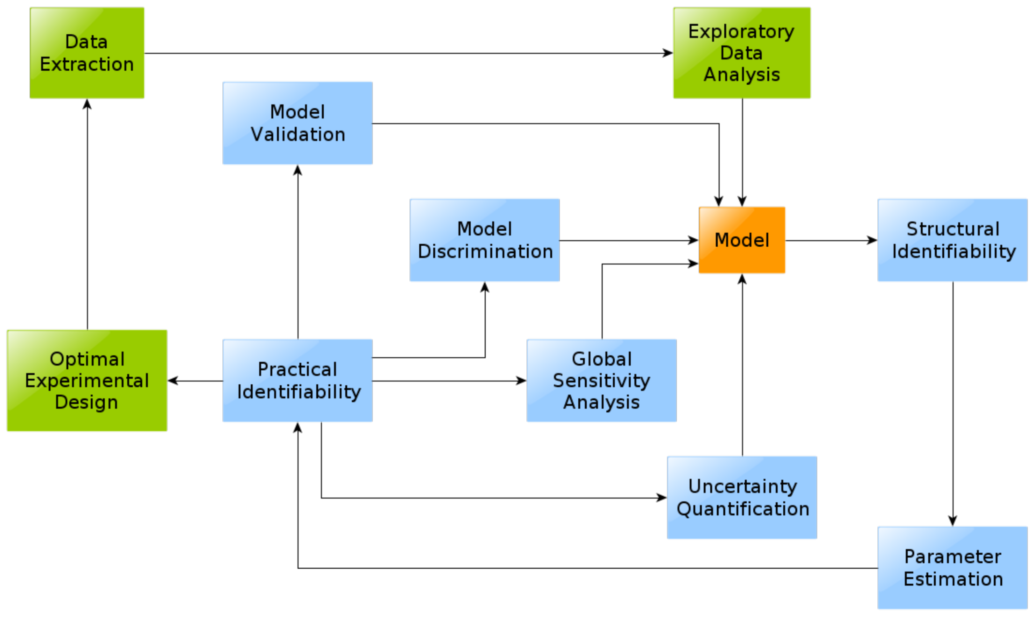
As a starting point for this analysis we take the official `BioPreDyn` description of work. In this document, the integrated software suite requirements are briefly described in two distinct deliverables (three if we consider D7.2, which is the prototype - or proof of concept - of the two other deliverables):

> D3.4) Integrated Suite of Tools: Integrated software-suite for iterative multi-scale model building providing tools for all the steps in the modeling cycle; documentation describing the suite, incl. algorithm comparison and applications
>
> D7.2) Prototype Software for Testing: User-friendly version of prototype software for testing in a setting for industrial applications.
>
> D8.3) Integrate Software Suite: Integrated software suite implementing the methods and tools developed during this project in an interoperable, user-friendly and well-supported way

A few keywords can be extracted from these descriptions:

- **Integrated software suite.** The objective of those deliverables therefore consist in combining several systems biology modeling tools together. Most of those tools are developed or maintained by consortium partners, but third-party tools will be considered. Tools used by consortium members include `DataRail`[2], `CellNOpt` [3], `AMIGO`[4], `COPASI`[5], `Cytoscape` [6]...
- **Modeling cycle.** The systems biology modeling cycle (as defined by the consortium) is consists in an eleven step cycle covering the main aspects of modeling, from data generation to model validation.

[1] http://www.biopredyn.eu/
[2] http://code.google.com/p/sbpipeline/wiki/DataRail
[3] http://www.cellnopt.org/
[4] http://www.iim.csic.es/~amigo/index.html
[5] http://www.copasi.org/tiki-view_articles.php
[6] http://www.cytoscape.org/


Data Extraction
Exploratory Data Analysis
Model Validation
Model Discrimination
Model
Structural Identifiability
Optimal Experimental Design
Practical Identifiability
Global Sensitivity Analysis
Uncertainty Quantification
Parameter Estimation

Specifications [1]: eleven step systems biology model building cycle

The software suite should therefore cover most of these steps, through the tools it will aggregates.

- **User-friendly.** The chosen implementation of the `BioPreDyn` software suite will have to be easily usable by non-developer people; a graphical user interface is required here.

3.2. Case study

As a pre-study for the software suite development, several tools and work flows used in the consortium were investigated. It aimed at detecting additional needs or constraints not expressed in the original description of work, and include them in the final design.

3.2.1. AMIGO

Summary. `AMIGO`[7] is a `Matlab`[8] tool box for systems biology model identification. It accepts `Matlab`, `Fortran` and SBML files as input model files, and `Matlab` files as input data files. Model identification steps can be generated in C or `Fortran` before being executed in order to improve performances.

Languages. `Matlab`, C, `Fortran`

Model building cycle steps. The following steps of the systems biology model building cycle are implemented in `AMIGO`:

- 4. *A priori* identifiability analysis
- 5. Model fitting / Parameter estimation
- 6. *A posteriori* identifiability analysis
- 9. Optimal experimental design (OED)
- 10. Global sensitivity analysis

Needs. `AMIGO` developers expressed the following needs for the integrated software suite:

- Data alignment standards

[7] http://www.iim.csic.es/~amigo/index.html
[8] http://www.mathworks.com/products/matlab/index.html

- Portable task, experiment and report generation
- Interoperability with third-party solvers

3.2.2. Fly suite

Summary. Collection of tools dedicated to developmental gene regulatory network[9] (GRN) study in *Drosophila melanogaster* embryos. The different tools cover the analysis from the embryo picture analysis to the GRN weight inference from experimental data:

- `fly_gui` is a `Java` software tool for gene expression profile extraction from stained fly embryo pictures. It generates custom data files containing the gene expression data for each gene, time point and cell nucleus in the selected stripe.
- `fly_sa` is a `C++` tool implementing a parallel version of the Lam simulated annealing algorithm applied to GRN inference.
- `Python` scripts are used besides those tools in order to compute intermediate values such as the promoter strength, diffusion parameters... and write them in the data file.

Languages. `Java`, `C++`, `Python`.

Model building cycle. The following steps of the systems biology model building cycle are implemented in the fly suite:

- 1. Data extraction
- 2. Exploratory data analysis
- 5. Model fitting / Parameter estimation

Needs. Fly suite developers expressed the following needs regarding the integration of their tools in the `BioPreDyn` software suite:

- Interoperability with third-party solvers
- Standard data-alignment tools (compatibility with SBML)

3.2.3. COPASI

Summary. `COPASI`[10] is a `C++` software tool for biochemical networks analysis; it implements a dedicated language called `CopasiML`[11] for encoding all the aspects of a `COPASI` simulation. `COPASI` accepts SBML as an input format for biochemical models.

Languages. `C++`, `Qt`.

Model building cycle. The following steps of the systems biology model building cycle are implemented in COPASI:

- 4. *A priori* identifiability analysis
- 5. Model fitting / Parameter estimation
- 6. *A posteriori* identifiability analysis

Needs. `COPASI` developers expressed the following needs regarding the integration of their tools in the `BioPreDyn` software suite:

[9] http://en.wikipedia.org/wiki/Gene_regulatory_network
[10] http://www.copasi.org/tiki-view_articles.php
[11] http://www.copasi.org/static/schemadoc/

- Interoperability with third-party tools (AMIGO)

3.3. Summary

With only three software tools / work flows investigated, several common characteristics appear:

- **Programming languages.** The described tools use a wide spectrum of programming languages: `C`, `C++`, `Java`, `Matlab`, `Fortran`...
- **Data / model alignment.** Portable standard formats for exchanging models and / or data are a major concern for all the development teams we met during this preliminary investigation; one model format seems to emerge: `SBML`. Besides, a format encoding the model, the simulation, the initial conditions and the results in a same file is requested.
- **Systems biology cycle steps.** Several steps of the model building cycle are covered by the tools described here, and several are not. Most developers / users requested the possibility to use alternative tools for specific steps in their analysis pipelines.

4. Use cases

In this chapter, we formalize the main features to be captured by the integrated software suite into use cases.

4.1. Systems biology model building cycle implementation

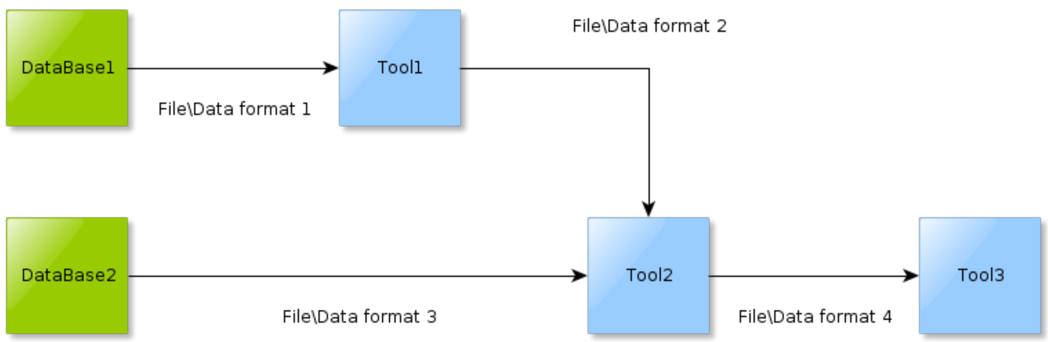
Model building steps. The chosen architecture should consist in an implementation of the systems biology model building cycle. It should allow modelers to execute all the steps of this cycle on a given model with a given data set.

Integration of `BioPreDyn` tools. As a part of the `BioPreDyn` project, the resulting software suite should use as many software tools developed within the consortium as possible, and integrate them as building blocks for the steps they cover.

Graphical user interface. As a tool intended to non-developers, the `BioPreDyn` software suite should be as user-friendly as possible, and give access to all its functionalities through a convenient graphical user interface.

4.2. Modularity

Sparse model building cycle. Not all the biochemical models use all the steps described in the systems biology model building cycle; most of them use only a subset of those steps. As a consequence, the chosen design should allow any combination of operation on the input model file.



Use cases [1]: representation of a generic analysis pipeline

Third party components. The `BioPreDyn` integrated software suite should not be limited to consortium tools only, and should allow third-party tool integration as model building blocks.

4.3. Simulation file standard

Standard formats. In order to be easily interfaced with other tools, the integrated software suite should be compatible with reference biochemical model and data formats.

Simulation formats. Model and data are closely related in the described model building cycle; each output model file should therefore be associated with the data used for model calibration. Similarly, the conditions according to which the model should be simulated should be specified in the model file, along with the results of this simulation. Such model / data structures are described both in SED-ML and SBRML languages.

4.4. Validation cases

Systems biology model building cycle prototype. A simple example of systems biology model (such as the one described in [3]) should be implemented as a proof of concept of the `BioPreDyn`software suite.

`BioPreDyn` use cases. The `BioPreDyn` description of work defines four biological modeling problems:

- Animal developmental gene regulatory networks
- Large-scale models of microorganisms
- Signaling and regulatory networks in cells
- Biotechnological production processes

As a validation step, each `BioPreDyn` use case analysis pipeline should be executed using the integrated software suite.

5. References

- **[1]** Balsa-Canto, E., Banga, J. R. (2011). AMIGO, a toolbox for advanced model identification in systems biology using global optimization.[12] Bioinformatics (Oxford, England), 27(16), 2311–3. doi:10.1093/bioinformatics/btr370
- **[2]** Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., et al. (2006). COPASI--a COmplex PAthway SImulator.[13] Bioinformatics (Oxford, England), 22(24), 3067–74. doi:10.1093/bioinformatics/btl485
- **[3]** Ashyraliyev, M., Fomekong-Nanfack, Y., Kaandorp, J. a, Blom, J. G. (2009). Systems biology: parameter estimation for biochemical models.[14] The FEBS journal, 276(4), 886–902. doi:10.1111/j.1742-4658.2008.06844.x
- **[4]** Jaqaman, K., Danuser, G. (2006). Linking data to models: data regression.[15] Nature reviews. Molecular cell biology, 7(11), 813–9. doi:10.1038/nrm2030
- **[5]** Chu, K.-W., Deng, Y., Reinitz, J. (1999). Parallel Simulated Annealing by Mixing of States.[16] Journal of Computational Physics, 148(2), 646–662. doi:10.1006/jcph.1998.6134

[12] http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3150037/?report=abstract
[13] http://www.ncbi.nlm.nih.gov/pubmed/17032683
[14] http://www.ncbi.nlm.nih.gov/pubmed/19215296
[15] http://www.ncbi.nlm.nih.gov/pubmed/17006434
[16] http://linkinghub.elsevier.com/retrieve/pii/S0021999198961344

- **[6]** Köhn, D., Novere, N. Le. (2008). SED-ML–an XML format for the implementation of the MIASE guidelines. [17] Computational Methods in Systems Biology, 176–190.
- **[7]** Dada, J. O., Spasic, I., Paton, N. W., Mendes, P. (2010). SBRML: a markup language for associating systems biology data with models.[18] Bioinformatics (Oxford, England), 26(7), 932–8. doi:10.1093/bioinformatics/btq069

[17] http://www.springerlink.com/index/N67N137071431XT7.pdf
[18] http://www.ncbi.nlm.nih.gov/pubmed/20176582