D 8.3 Second prototype with demonstration

Project acronym: *MAPPER*

Project full title: Multiscale Applications on European e-Infrastructures.

Grant agreement no.: 261507

Due-Date:	30 september
Delivery:	
Lead Partner:	Cyfronet
Dissemination Level:	Public
Status:	
Approved:	Q Board, Project Steering Group
Version:	1.1

DOCUMENT INFO

Data and version number	Author	Comments
06.07.2012 v0.1	Katarzyna Rycerz	Plan of the document
24.07.2012 v0.2	Katarzyna Rycerz	General overview; Draft of evaluation of efficiency
14.08.2012 v0.3	Tomasz Gubala	Added parts concerning MaMe
05.09.2012 v0.4	Grzegorz Dyk	Sections: GridSpace connection with AHE and QCG clients, Provenance
06.09.2012 v0.5	Katarzyna Rycerz	Sections about GridSpace Experiment Workbench and Execution Engine
06.09.2012 v0.6	Alexandru Mizeranschi	SBML toolbox section
06.09.2012 v0.7	Joris Borgdorff	Sections about the jMML library
06.08.2012 v0.8	Daniel Harezlak	sections about MAD
07.08.2012 v0.9	Katarzyna Rycerz	Minor corrections, formatting
10.09.2012 v 1.0	Katarzyna Rycerz	Minor corrections, formatting
18.09.2012 v 1.1	Katarzyna Rycerz	Corrections after project internal review.

TABLE OF CONTENTS

1 Executive summary 5
2 Contributors 5
3 Glossary of terms 6
4 General architecture of the second prototype. 8
5 Value added since first prototype 9
5.1 User Interfaces and visual tools 9
5.1.1 Changes to MML and jMML Library 9
5.1.2 Multiscale Application Designer (MAD) 9
5.1.3 GridSpace Experiment Workbench and Result browsing 11
5.2 Programming tools 13
5.2.1 MaMe: MAPPER Memory and xMML Repository 13
5.2.2 The SBML toolbox 16
5.3 Execution Tools 18
5.3.1 Execution Engine Second Prototype 18
5.3.2 Connection with QCG Client Second Prototype 20
5.3.3 Connection with AHE client second prototype 21
5.3.4 Browsing Results of MAPPER Applications 22
5.4 MAPPER Provenance data collector and storage 23
6 Prototype availability 25
6.1 jMML library 25
6.2 MAD 25
6.3 GridSpace Experiment Tools 25
6.4 MaMe 27
6.5 SBML toolbox 27
6.6 Provenance System 27
7 Evaluation of efficiency of WP8 tools 28
8 Conclusions 32
9 References 33

LIST OF FIGURES

Fig. 1. Architecture of the second prototype of the programming and execution tools in the context of the full design presented in D8.1. 8

Fig. 2. A snapshot of the current implementation of MAD showing sample multiscale application composed from single scale submodules. 10

Fig. 3. A single scale model presented by MaMe Model Registry. 14

Fig. 4. Any element of MAPPER application (scale model, a mapper or a filter) may have its implementations registered in MaMe. Scale models and mappers may have ports, which are the means of putting together complex applications into a single workflow processing. 14

Fig. 5. Main view of the Experiment (xMML) Repository part of MaMe showing the list of recorded experiment (application) descriptions, each being an xMML document in the specific version of this notation, with the list of application's elements and their interconnections. 15

Fig. 6. Class diagram of MultiGrain 17

Fig. 7. The current provenance architecture. 23

Fig. 8. Ontology used for describing provenance 24

Fig. 9. Process of constructing multiscale application - the steps of the process are indicated as rectangles; supporting tools are indicated as circles. The following colours were used: MML support - orange, tools - blue, external services - green 28

Fig. 10. Sample user answers to the SUS questions during first seasonal MAPPER school. 32

LIST OF TABLES

Tab. 1. Registered interpreters in GridSpace Execution Engine 20

Tab. 2. Availability of GridSpace software modules 25

Tab. 3. Multiscale application creation steps. For every step, we describe if and how it can be supported by the tools instead of a user manual work. 29

Tab. 4. Status of using WP8 tools in MAPPER applications 30

1 Executive summary

This document describes the functionalities and possible access to demonstrations for the multiscale programming and execution tools in the MAPPER project. More specifically, D8.3 is the second prototype of the tools facilitating creation and execution of multiscale applications with structure described in Multiscale Modelling Language. The presented tools support composition of multiscale applications from existing single scale submodules installed on e-infrastructures. After being composed, such applications are executed. The second prototype contains improved programming and execution tools which includes: the application composition tool called Multiscale Application Designer (MAD), Registry for application modules description implemented as MAPPER Memory (MaMe), tools supporting high level stage of execution: GridSpace (GS) Experiment Workbench (EW) and GS Execution Engine. As planned in D8.1 and D8.2, we have also present prototypes of new tools: xMML Repository, Provenance and Result Management. Additionally, we also present a status of the integration of GridSpace Execution Engine with AHE interoperability layer on the API level.

We present architecture of the current implementation, the detailed description of the presented tools, their current functionality, list of changes from the first prototype described in D8.2 and links to the prototype, code repository and/or demo film.

The document is organized as follows: In Section 4 we briefly describe architecture of the prototype and its relation to design in D8.1 and status presented in D8.2. The detailed information about each tool prototype can be found in Section 5. In Section 6 we list links to prototypes, code repositories and/or demonstration videos. Section 7 outlines current evaluation of efficiency of WP8 tools. We conclude in Section 8. The status of the actual MAPPER applications and their preliminary usage of the tools can be found in D7.2.

2 Contributors

Below we list the institutions and names of the contributors. Their exact role in this deliverable is depicted in the document info table at the beginning of the document**.**

Cyfronet: K. Rycerz, G. Dyk, T. Gubała, D. Harężlak, E. Ciepiela, M. Bubak

PSNC: M. Mamoński ,T. Piontek

UvA: Joris Borgdorff

UU: Alexandru Mizeranschi

UCL: Stefan Zasada

UNIGE: M.Ben Belgacem

3 Glossary of terms

In this document we will use terminology listed below. Additional glossary of terminology can be also found in Section 3 of D 8.1.

Application Hosting Environment (AHE): a framework supporting running applications on Grid infrastructures hosting Globus, UNICORE, QCG-Computing or GridSAM middleware.

Car-Parrinello Molecular Dynamics (**CPMD):** package containing a parallelized plane wave / pseudopotential implementation of Density Functional Theory, particularly designed for ab-initio molecular dynamics.

CxA: Ruby-based file format that describes a MUSCLE application: (1) modules parameters (2) couplings between modules.

Executor: a common entity for hosts, clusters, grid brokers etc. It's anything that is capable of running software which is already installed on it (represented as Interpreters).

Experiment host: host where GridSpace experiment is executed.

Filter: in MML terminology one-to-one type of connection between submodels.

gMML: see MML.

GUI: Graphical User Interface.

jMML : java library supporting MML.

GridSpace experiment: set of snippets in various script languages stored in XML file.

Interpreter: a software package accessible from any script language available on any infrastructure accessible by MAPPER community. An example of interpreter can be MUSCLE (See D 8.1) or LAMMPS[1] tools. We assume that the software is installed in WP4.

JobProfile: see QCG JobProfile.

Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS): package supporting classical molecular dynamics simulations.

Loosely coupled and **tightly coupled**: a collection of submodels instances is loosely coupled if there is no cycle between them in the coupling topology, and tightly coupled otherwise.

Mapper: in MML terminology mapper is one-to-many type of connection between single scale submodels. Note the difference between mappers and the MAPPER project.

MAPPER memory (MaMe): semantics-aware persistence store for MAPPER metadata based on xMML description.

Multiscale Application Designer (MAD): MAPPER application composition tool.

Metadata: data about data (e.g. link to actual file, but not file itself).

Multiscale model: the model of a multiscale process.

[1] http://lammps.sandia.gov/

Multiscale Modelling Language (MML): the high level concept of the language that describes single scale submodels and their connections. The connection can be realized by mappers (one-to-many type of connection) or filters (one-to-one type of connection with data filtering). It is a concept for modelers and has several representations. The one described in this document are xMML and gMML:

- **xMML**: the XML representation of MML that contains all information about application structure. The latest version of xMML specification can be found on http://napoli.science.uva.nl/xmml/xmml.tar.gz.
- **gMML** - the graphical representation of MML that contains only part of information about application structure, useful for modellers and application developers.

Multiscale Coupling Library and Environment (MUSCLE): a communication library that can be used to connect modules implementing single scale models into a multiscale simulation. The structure of the application is described in CxA file.

Submodel (Single scale model): a model of a single scale process. In the context of a multiscale model, a submodel.

Snippet: a piece of code in a script language.

Synchronization points: points during execution that one submodel instance will need to synchronize with another (including itself), by requiring input.

System Biology Markup Language (SBML): XML-based language for representing models. It's oriented towards describing systems where biological entities are involved in, and modified by, processes that occur over time.

QosCosGrid (QCG): a resource and task management system aiming to provide supercomputer-like performance and structure to cross-cluster large-scale computations that need guaranteed level of Quality of Service (QoS).

QCG JobProfile: XML-based language describing how to execute an application using QCG middleware.

Repository: place where multiscale applications' description files are stored and managed (e.g. xMML files).

Registry: place where information (metadata) about some entities (in our case simulation modules) are registered (but modules themselves are not stored!).

Task graph: an acyclic directed graph representation of the submodel instances and their synchronization points as they unfold over time. It may include each of the operators of the SEL as nodes.

User Interface machine (UI): machine accessible directly (via ssh) by a user from which he can access other (Grid, PBS) resources.

xMML: see MML.

4 General architecture of the second prototype.

The general architecture of the second prototype is shown in the Fig. 1, which is enhanced version of the architecture presented in D8.1 and D8.2 according to evolving user requirements.

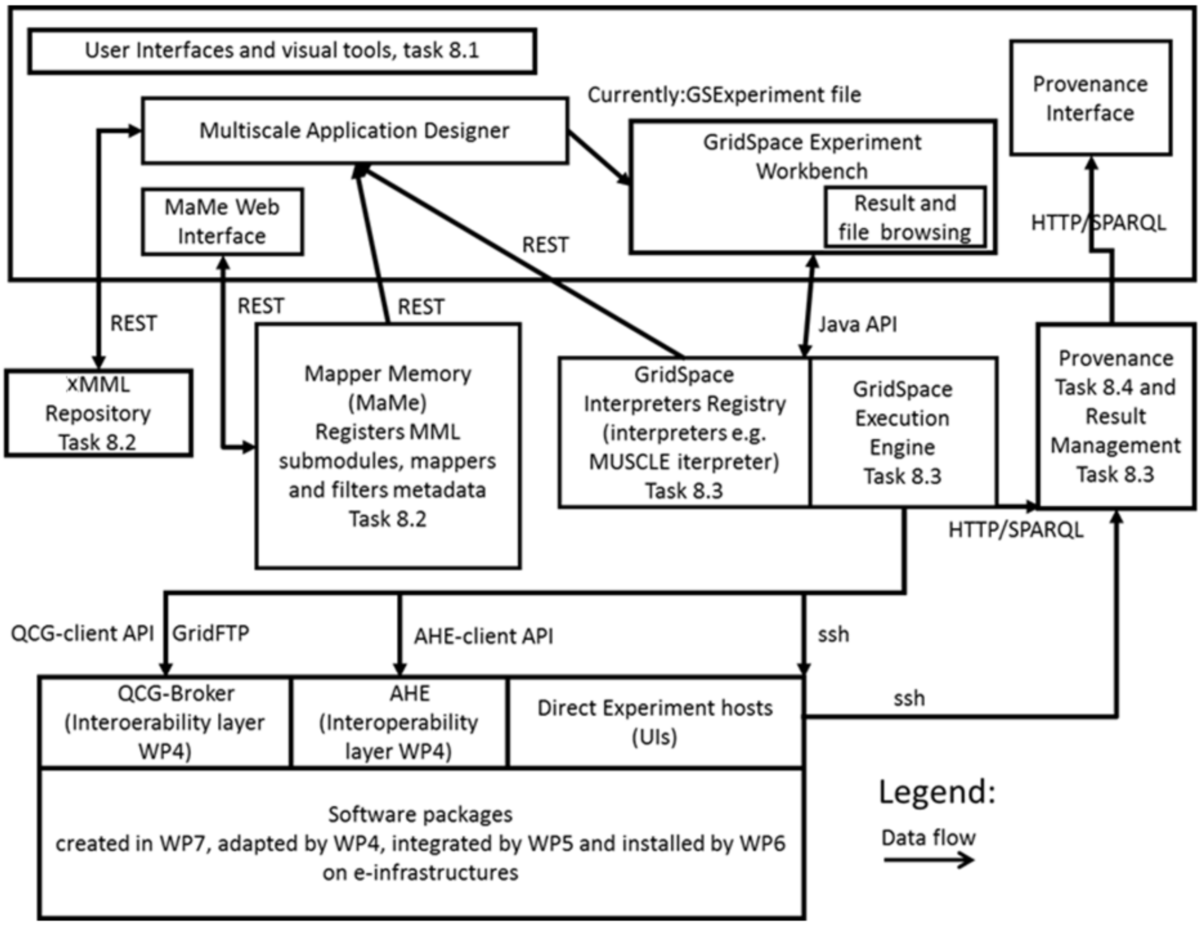

User Interfaces and visual tools, task 8.1
Multiscale Application Designer
Currently:GSExperiment file
GridSpace Experiment Workbench
Result and file browsing
Provenance Interface
MaMe Web Interface
REST
REST
REST
REST
HTTP/SPARQL
Java API
xMML Repository Task 8.2
Mapper Memory (MaMe) Registers MML submodules, mappers and filters metadata Task 8.2
GridSpace Interpreters Registry (interpreters e.g. MUSCLE iterpreter) Task 8.3
GridSpace Execution Engine Task 8.3
Provenance Task 8.4 and Result Management Task 8.3
HTTP/SPARQL
QCG-client API
GridFTP
AHE-client API
ssh
ssh
QCG-Broker (Interoerability layer WP4)
AHE (Interoperability layer WP4)
Direct Experiment hosts (UIs)
Software packages created in WP7, adapted by WP4, integrated by WP5 and installed by WP6 on e-infrastructures
Legend:
Data flow

Fig. 1. Architecture of the second prototype of the programming and execution tools in the context of the full design presented in D8.1.

The second prototype contains improved version of the multiscale programming and execution tools presented in D8.2. This includes: Multiscale Application Designer (MAD), Modules Registry implemented as Mapper Memory (MaMe), GridSpace (GS) Experiment Workbench and GS Execution Engine with its Interpreters Registry. As planned, we have also present prototypes of new tools: xMML Repository, Provenance and Result Management. xMML repository stores xMMLs of applications and provides them to MAD for reusability. Result Management is a realized as a part of Provenance system that is able to save snapshots of experiment results together with their metadata. Therefore a user is able to view experiments results and their metadata using Provenance Interface. Apart from improving integration with of GridSpace Execution Engine with QCG-Broker, we also implemented integration with AHE interoperability layer on the API level.

We will subsequently improve the presented prototype by improving modules according to evolving user requirements.
Next chapters present the detailed description of the presented tools, contain architecture of the current implementation, its current functionality, list of changes from the first prototype described in D8.2 and links to the prototype, code repository and/or demo film.

5 Value added since first prototype

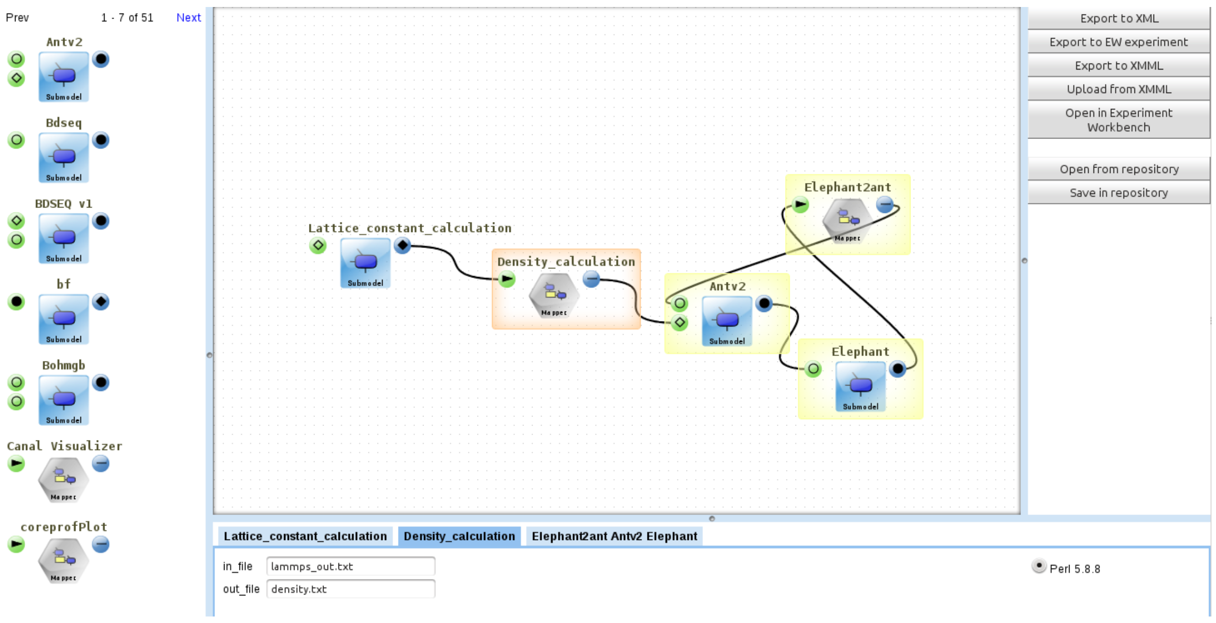
5.1 User Interfaces and visual tools

5.1.1 Changes to MML and jMML Library

The jMML library is a Java library that handles Multiscale Modeling Language (MML) [BORGDORFFa]. The first prototype of jMML is described in detail in Subsection 5.1.1 of D8.2. The MML specification is described in detail in Subsection 4.2 of D8.1 and has few changes. The major difference is the addition of a new computational element called a terminal. A terminal can directly terminate a conduit. There are two types of terminal: a source and a sink. A sink will read any data that comes over a conduit and a source will generate any data. Natural sources and sinks that come to mind are file readers and writers, so that messages received over a conduit are actually the contents of files.
In addition to what was decribed in Subsection 5.1.1 in D8.2, the jMML library can now output a MUSCLE configuration file for a given xMML file. It can also generate a directory structure with preliminary code and base code already filled in based on the xMML file.

5.1.2 Multiscale Application Designer (MAD)

Multiscale Application Designer (MAD) is a graphical tool enabling easy multiscale application composition out of individual components of the MML language. The tool is available to users through a web browser and uses convenient drag-and-drop techniques to build applications. Since the first prototype was published and described in deliverable D8.2 in section 5.1.2 a number of changes were implemented in the second prototype. The most important ones include user interface changes, xMML repository usage and implementation of a flexible module management mechanism. The changes are described in detail below.



in_file lammps_out.txt
out_file density.txt

Fig. 2. A snapshot of the current implementation of MAD showing sample multiscale application composed from single scale submodules.

GUI enhancements

The second prototype of MAD is presented in Fig. 1. The bottom part of the interface is now occupied by a property editor which allows to change properties (including simulation parameters) of MML components. The tabs of the editor correspond to individual nodes in the workspace area, unless a tightly-coupled section is present in which case a single tab combines the section components' properties. The editor fills in the property forms with default values coming from the MaMe registry. If present, different MML component implementations can be chosen in the property editor. Another feature of the property editor is handling of global parameters which are defined in MaMe in an MML component namespace. This may introduce conflicts if the same global property is imported by a few components. Such cases are discovered and a proper warning is produced. The user may pick a global parameter value of one of the conflicting components or specify a new one. When an application is saved all changes applied in the editor are saved and restored when the application is being imported into MAD.

xMML repository

As described in the Subsection 5.2.1 an xMML repository was made available and allows for storing and retrieving of multiscale applications written in the xMML format. MAD utilizes this by providing xMML repository widgets. This allows users to compose their applications and afterwards save and share them with other users. Stored applications are annotated by a name and a short description. If an application with the same name is saved the old one is overwritten by putting it into the archive for later recovery. Saved applications cannot be removed directly from MAD but it is possible to be done through MaMe by providing valid credentials.

Module management

First prototype of MAD generated an executable form of tightly-coupled sections by using (among others) low-level properties which were specific for an execution environment (infrastructure site) and were defined in MaMe. This posed interoperability problems on the modeling level. Additionally, the feature of executing particular parts of tightly-coupled sections on different execution sites brought extra difficulties in generating proper mapping descriptions. In the second prototype mapping between parts of tightly-coupled sections and execution sites was moved to GridSpace Experiment Workbench (EW) which enabled the modeling layer to operate on abstracted dependencies. This however required the EW experiment generation process to be modified. The modification is in place and passes correspondences between MML components belonging to a tightly-coupled section and abstract module names. This enables different mapping strategies to be implemented in EW in order to optimally use the available infrastructure and keeping the application descriptions abstract.

5.1.3 GridSpace Experiment Workbench and Result browsing

Since previous release of the first prototype, which was thoroughly described in Section 5.1.3 of D8.2, a considerable effort has been devoted to further development of GridSpace platform in order to embrace requirements and characteristics of multiscale applications. In particular, GridSpace Experiment Workbench (EW) being a web-based user interface to the platform has been refactored and enriched in order to better support scientific application developers in composing and running multiscale, distributed, cross-site and time-consuming applications.

Support for long-running experiments.

One of the major deficiencies of Experiment Workbench in the first prototype was lack of proper handling of long-running experiment. EW didn't give means to run experiments in an unattended way so the users had to be logged into EW during all experiment run time, keeping the session with EW open for long time. In the second prototype EW handles experiment execution on the server side thus allowing user to log out from EW and re-log again anytime later to see the status of the execution that is carried out continuously "in background". Technically speaking, it was achieved by decoupling user sessions with EW server from underlying execution service that can operate without user being logged in.

When a user logs out form EW, all his or her experiment executions are being detached from user session and kept “parked” on the server side, what happens without interrupting the computations, until a user re-log again. Then, experiment executions are brought back form server-side “parking” and reattached to a user’s session. In consequence, user’s workbench became persistent in a way that one can log in and log out at any time, and even when session expires the state of workbench is persisted on the server side and can be retrieved when user re-log in again. Moreover, users are notified when the session expiration time is about to be reached and asked if the session has to be prolonged.

Improved URL handling.

Following up improvements aimed at supporting long-running experiments, the URL handling mechanism has been reworked in order to enable preserving EW user interface state. Now, URL encodes the state of the user interface, thus subsequent references to a given URL retrieve EW user interface in exactly the same state. That enables users to save user interface state of their interest as an URL and retrieve it any time by navigating to that URL. What is more, the web browser history management is now better supported as “back” and “refresh” buttons work as web browser users got used to.

One click opening of MAD-generated experiments.

EW now can open experiments generated by external applications, most notably Multiscale Application Designer (MAD), using dedicated HTTP endpoint (/workbench/open). Making HTTP POST request to such URL results in opening the experiment provided as a content of the request. If user is not yet authenticated the login form is displayed. After successful authentication EW is opened with the experiment that has been passed from external application. Owing to this feature, MAD can now offer one click way to redirect to EW and open the experiment generated on the fly by MAD.

Login with proxy certificate generated on the fly.

Since some executors such as AHE and QCG use grid proxy certificates for authentication and authorization purposes, a new way of logging is now enabled in EW. Users are no longer required to generate proxy on their own, manually or using specific tools. Instead, EW login form can accept a pair of grid certificate and corresponding certificate key and generate proxy certificate on the fly within web browser using dedicated built-in applet. This way, users’ sensitive personal credentials are not being sent outside their computers: proxy is generated within their web browsers and after that used by EW for authentication and authorization.

Pluggable openers for visualizing and editing data files.

In order to deliver capabilities of visualization and editing of complex data files the idea of, so called, openers has been introduced. Opener is a web application powered by as e.g. applet, javascript, flash, or any other rich internet application technology. Openers are served by EW and run within users' web browsers sharing the same secured web session with EW. Openers are pluggable and can be added to EW anytime by its provider. Owing to this, openers can be developed and delivered independently. In particular, already existing web-based tools can be reused and adopted to fit in EW architecture. Openers interoperate with EW using its REST interface dedicated for openers: each data file stored on an executor is assigned with HTTP URL and can be fetched and saved using HTTP GET and PUT requests, respectively. Openers are then enabled to GET data file in a given supported file format and deal with it in its own specific way e.g. visualize or open for edition. Modified file can be then PUT back on the servers and stored in its original location.

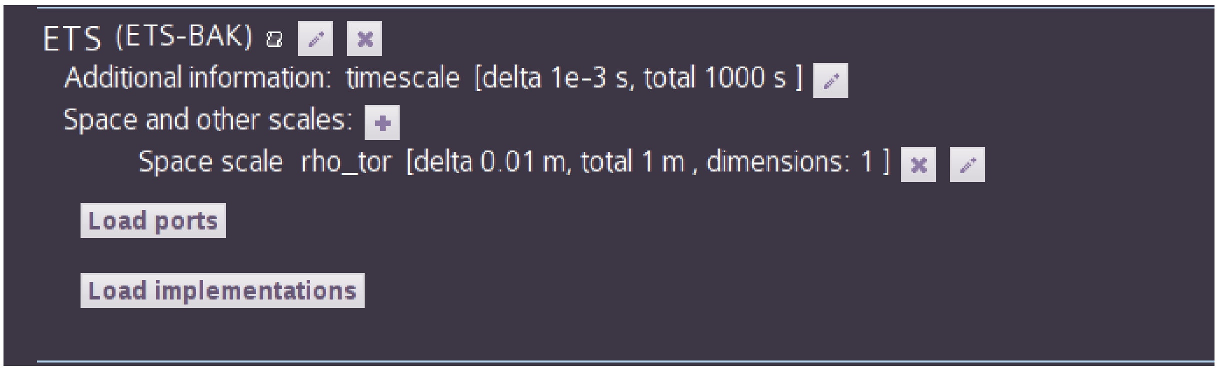
5.2 Programming tools

5.2.1 MaMe: MAPPER Memory and xMML Repository

Mapper Memory (MaMe) is a core element of Mapper multiscale applications toolbox. It serves as a persistence layer and a domain-aware registry for metadata regarding structure, implementation and execution of multiscale applications on computational infrastructure. As described by Section 8.2 of D8.1, MaMe stores information on scale modules, mappers and filters, together with their ports, implementations and other constituting elements and attributes.
The first prototype of MaMe was described in Section 5.2 of D8.2. This section of the document is intended to provide the description of the progress made in the design and development of this component. The most important addition is the entirely new xMML application description repository. Also, in the models registry, part of MaMe, there were some additions and developments made. The following subsections describe them in detail.

Changes in Models Registry

In comparison to the description of the first prototype of the Model Registry of MaMe (Section 5.2.1.1 of D8.2), there were only minor changes made - mainly in order to get a better user experience and smoother workflow across the user interface.



ETS (ETS-BAK)

Additional information: timescale [delta 1e-3 s, total 1000 s]

Space and other scales:

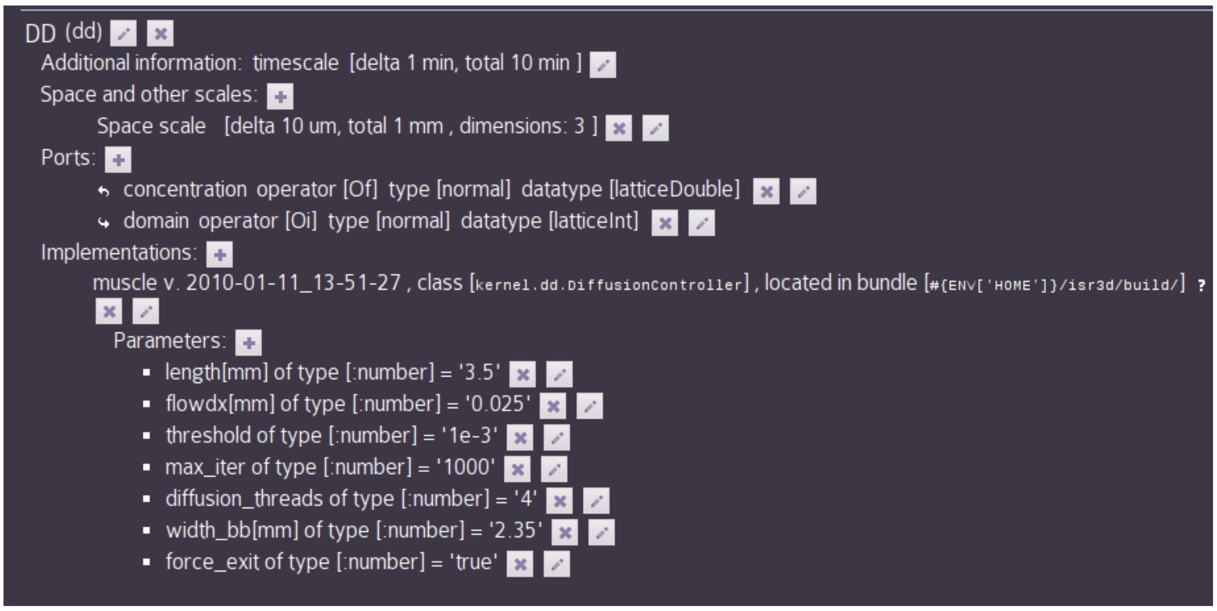
Space scale rho_tor [delta 0.01 m, total 1 m , dimensions: 1]

Load ports

Load implementations

Fig. 3. A single scale model presented by MaMe Model Registry.

As presented in Fig. 3, due to growth of the number of stored models and mappers, the element list was made more concise. Only the most general metadata is being loaded right away when browsing the list of available models, mappers and filters. Any following information and dialog boxes (ports, implementations) are loaded on-demand with AJAX asynchronous calls. The user may get full information on a given multiscale application element by loading additional information with a click (see Fig. 4).



DD (dd)

Additional information: timescale [delta 1 min, total 10 min]

Space and other scales:

Space scale [delta 10 um, total 1 mm , dimensions: 3]

Ports:

concentration operator [Of] type [normal] datatype [latticeDouble]

domain operator [Oi] type [normal] datatype [latticeInt]

Implementations:

muscle v. 2010-01-11_13-51-27 , class [`kernel.dd.DiffusionController`] , located in bundle [`#{ENV['HOME']}/isr3d/build/`] ?

Parameters:

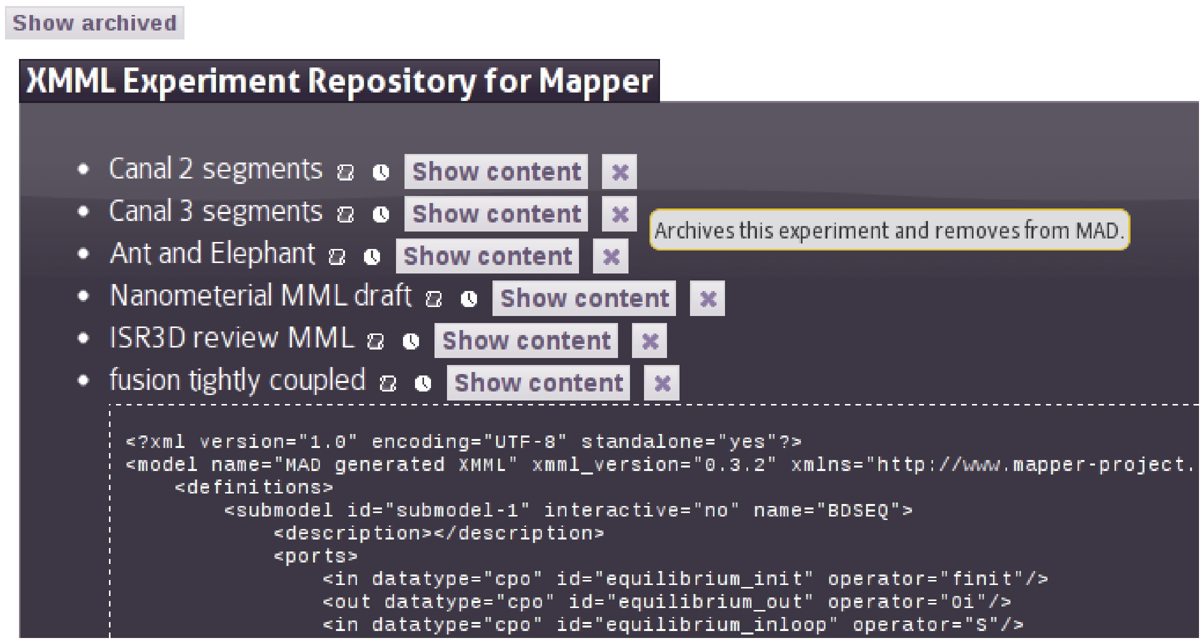
- length[mm] of type [:number] = '3.5'
- flowdx[mm] of type [:number] = '0.025'
- threshold of type [:number] = '1e-3'
- max_iter of type [:number] = '1000'
- diffusion_threads of type [:number] = '4'
- width_bb[mm] of type [:number] = '2.35'
- force_exit of type [:number] = 'true'

Fig. 4. Any element of MAPPER application (scale model, a mapper or a filter) may have its implementations registered in MaMe. Scale models and mappers may have ports, which are the means of putting together complex applications into a single workflow processing.

Further changes to the user interface of the Models Registry concern finer-grained interaction model. Now the administrator of MAPPER infrastructure or the developers of application modules are able to add and remove not only the models, their ports and their implementations, but also scales and the specific parameters of these implementations. Also, any element might be modified in-situ, so no longer the "remove-and-reregister" procedure is forced upon the user of MaMe.

New xMML Repository

As stated by Section 5.2.1.2 of D8.2 the previous (initial) version of the xMML Repository of multiscale application descriptions was very rudimentary. Afterwards it was completely rewritten to provide better usage experience for users and developers of other MAPPER tools, according to the design (Section 8.2.1 of D8.1). Currently the Repository (also referred to as Experiment Repository) constitutes another important building block of MaMe.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<model name="MAD generated XMML" xmml_version="0.3.2" xmlns="http://www.mapper-project.
    <definitions>
        <submodel id="submodel-1" interactive="no" name="BDSEQ">
            <description></description>
            <ports>
                <in datatype="cpo" id="equilibrium_init" operator="finit"/>
                <out datatype="cpo" id="equilibrium_out" operator="Oi"/>
                <in datatype="cpo" id="equilibrium_inloop" operator="S"/>
```

Fig. 5. Main view of the Experiment (xMML) Repository part of MaMe showing the list of recorded experiment (application) descriptions, each being an xMML document in the specific version of this notation, with the list of application's elements and their interconnections.

As presented in Fig. 5, the Experiment Repository holds descriptions of multiscale applications. These description may be written by the users by hand but the preferred and better method of planning applications is to use the MAD tool to design a new multiscale application in a user-friendly way. MAD is able both to load saved xMML description and to save the new ones using this repository.

In order to provide the method of removing unnecessary, older versions of experiments, the experiment archive is being provided - removing any xMML experiment description from MaMe will move it to the archive (instead of deleting it completely). Such archived experiments do not clutter MaMe and MAD displays but are easy to retrieve back with the use of Reload from Archive capability of MaMe.

In order to accommodate MAD and similar tool using MaMe internally, a HTTP/REST interface was developed for the Experiment Repository. Currently it contains three operations:

- experiments_list - returns the full list of available (i.e., not archived) experiments and their metadata, in JSON format,
- experiment_content - returns the full xMML document for a specific experiment saved in MaMe,
- save_experiment - performs an "upsert" operation on a given experiment: that is it saves it when no such experiment existed or it updates its content, if it had been already recorded by MaMe in the past.

The capability of being able to parse ready xMML description in order to retrieve and automatically register all models, mappers and filters mentioned there, as described by Section 5.2.1.2 of D8.2, was preserved and is still available in the current version of MaMe.

5.2.2 The SBML toolbox

Compared to the status described in the Subsection 5.2.2 of D8.2 deliverable, the SBML[2] toolbox was improved in Year 2 of the MAPPER project in a number of ways:

- The previous ordinary differential equation (ODE) solver (XPPAUT[3] , Ermentrout, 2002) has been replaced with a native Java library, resulting in increased performance (reduced computation times),
- SBML models can now be automatically created for a given number of genes and for a given rate law,
- A larger number of ODE-based rate laws is supported,
- A component to visualize the GRN models represented in the SBML format has been added.

As previously described in deliverable D8.2, the SBML toolbox has been developed as a set of Java classes and tools, used by the MAPPER Computational Biology application which we named MultiGrain ("Multiscale Gene Regulation Modelling Tool"). Based on the developments in Year 1 of the MAPPER project, we have redesigned some aspects of our Java-based gene regulatory network (GRN) modelling and simulation tool, in order to match our current needs. These changes are described in deliverable D 7.2, while here we focus on

[2] The systems biology markup language: http://sbml.org

[3] XPPAUT, a tool for solving stochastic, differential, and difference equations: http://www.math.pitt.edu/~bard/xpp/xpp.html

the SBML toolbox exclusively. Figure 1 shows the class diagram representing the current status of the application.

A first important change was replacing XPPAUT with Michael Thomas Flanagan's Java scientific library[4]. Since our GRN tool is also implemented in Java, this resulted in a considerably increased performance, due to the fact that ODE solving is an important step in simulating our SBML models of GRNs. The class structure was also changed to enable this, eliminating the previous Xppaut class and adding others such as:

- GRNModel, containing information about the GRN model which is being reverse-engineered
- DifferentialEquation, an abstract class containing general information
- DifferentialEquationSystem, aggregating
- A class implementing the DifferentialEquation class for every type of rate law supported.

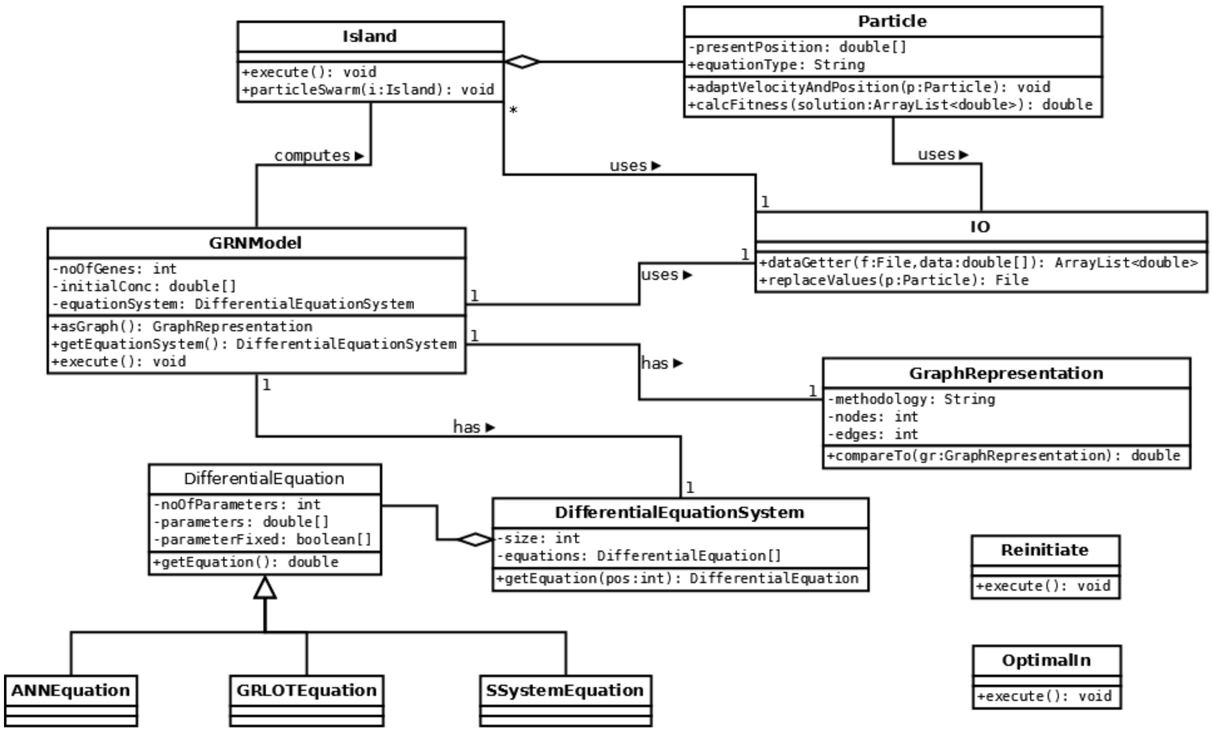

Island
+execute(): void
+particleSwarm(i:Island): void
Particle
-presentPosition: double[]
-equationType: String
+adaptVelocityAndPosition(p:Particle): void
+calcFitness(solution:ArrayList<double>): double
computes ▶
uses ▶
uses ▶
*
1
IO
+dataGetter(f:File,data:double[]): ArrayList<double>
+replaceValues(p:Particle): File
GRNModel
-noOfGenes: int
-initialConc: double[]
-equationSystem: DifferentialEquationSystem
+asGraph(): GraphRepresentation
+getEquationSystem(): DifferentialEquationSystem
+execute(): void
uses ▶
has ▶
has ▶
GraphRepresentation
-methodology: String
-nodes: int
-edges: int
+compareTo(gr:GraphRepresentation): double
DifferentialEquation
-noOfParameters: int
-parameters: double[]
-parameterFixed: boolean[]
+getEquation(): double
DifferentialEquationSystem
-size: int
-equations: DifferentialEquation[]
+getEquation(pos:int): DifferentialEquation
Reinitiate
+execute(): void
OptimalIn
+execute(): void
ANNEquation
GRLOTEquation
SSystemEquation

Fig. 6. Class diagram of MultiGrain

Second, we decided to add support for generating SBML models from within our toolbox, requiring the user only to provide the number of genes in the GRN and to choose a rate law. In contrast, the user was previously required to provide an SBML file created in an external tool such as COPASI [HOOPS] or CellDesigner [FUNANHASHI]. In this way, we are able to enforce a common naming system for the different parameters that appear in our equations, simplifying the process of adding (implementing) new methods.

[4] Michael Thomas Flanagan's scientific library: http://www.ee.ucl.ac.uk/~mflanaga/java/

After initially focusing on only one type of rate law (the Artificial neural network introduced by [VOHRADSKI], we added support for four additional types of ODEs used in systems biology literature: the S-system method [SAVAGEAU], the generalized rate law of transcription method [MENDES], the Hill equation and the generalized mass action model [VOIT]. For each rate law, we created a Java class extending the abstract class DifferentialEquation, containing a method which generates the ODE when needed by the ODE solver.

Finally, we added support for visualizing GRN models as graphs. We created the GraphRepresentation class for this, using the Java Jgraph library[5] for the implementation. Our aim is to also implement a means for comparing two GRNs based on their structure similarity. A score will be given to represent how similar a graph is to another, useful for example for knowing how well a reverse-engineered model matches the structure of a real GRN, in cases when this structure is known beforehand.

We are now in the course of preparing and running scientific experiments employing the MultiGrain tool. As a result of this, we will present a practical, scientific use of the SBML toolbox as part of a concrete systems biology tool. Also, due to the modular, fine-grained nature of our components we expect them to be easily reused into other similar projects.

5.3 Execution Tools

5.3.1 Execution Engine Second Prototype

GridSpace Execution Engine constitutes a back-end facility for submission of experiments developed and run through GridSpace Experiment Workbench (EW) described in Subsection 5.1.3 of this document. In order to support functionality offered by EW and other MAPPER tools in the second prototype the following functionality has been added since the first prototype release of Execution Engine.

Support for long-lasting connections with executors.

Implication of the requirement of supporting long-running experiments (explained in [4.1.3]) was ensuring long-lasting connections with executors that rely on connection-oriented protocols (e.g. SSH, GridFTP). For this purpose Execution Engine periodically triggers dedicated “keep alive” call against all connected executors. Depending on concrete implementation of an executor, keep alive call may be implemented in various, executor-specific ways. This call must be handled by the executors who are to perform all necessary activities in order to avoid session expiration and check connectivity with external systems the executor relies on, if any.

[5] The JGraph library: http://www.jgraph.com/

Isolated executor client modules.

Since Execution Engine is intended to embrace unrestricted range of executors the mismatches and software conflicts between executor client modules were expected to occur and indeed have happened. AHE and QCG client modules proved incompatible with each other what has been remedied by introducing isolation between executor client modules enabled by Java class loader mechanism. Using different class loaders per individual executor, the isolation between executor classes has been achieved. Executors share only the minimum amount of classes that are loaded by Execution Engine, while the rest is loaded by the executors independently using isolated and separated class loaders. That avoids name conflicts of in Java classes induced by version conflicts of modules that multiple executors depend on. Such low-level isolation puts robust foundation that strengthens extendability of EE in terms of a quantity of executors that can coexist in single Execution Engine instance.

Compound assets.

As it was encountered in the cases of some multiscale application modules, they may use not only single input and output files but whole directories with a given layout as well. Therefore, GridSpace experiments had to support directories as their input/output assets. First, application developers have been enabled to specify a path to an input/output directory using terminating slash character. In this case, EW deals with such-defined input/output as with a directory. Second, it can be specified that a given directory is a compound of a set of given single input/output files. Copying single files to and from compound assets is handled transparently by EW.

Parameterized interpreter arguments.

After the first prototype of Execution Engine it was found that when using scientific software packages (configured as interpreters in GridSpace) on different computing sites (configured as executors in GridSpace) some interpreter- or executor-specific parameters have to be set when submitting computations. For example, a queue name in the PBS batch system where to submit jobs is an executor-specific parameter, specific to a given computing site. Configuration file of a given scientific software package is an interpreter-specific parameter that needs to be provided on whatever site this software is to be run. For this purpose, EE allows for specification of a program invocation command that contains interpreter- and executor-specific parameters to be provided in the experiment and experiment execution descriptor, respectively.

New resources configured in interpreters registry.

Working on porting Mapper multiscale application portfolio to the Mapper framework, new interpreters have been registered prior to the release of the second prototype. Interpreters were indicated by the depending applications, installed on indicated sites and e-infrastructures and then registered as indicated in Tab. 1.

Tab. 1. Registered interpreters in GridSpace Execution Engine

Interpreter Id	Used by application	Available on executors
CPMD-3.13-2	Nano	AHE, Mavrino site, Zeus site
CPMD2CUBE-3.13-2	Nano	Mavrino site
MSI2LMP_POT-30Sep11	Nano	Mavrino site
LAMMPS-30Sep11	Nano	AHE, Mavrino site, Zeus site
CanalVisualizer-1.0	Canals	Zeus site, Grass1 site
Mencoder-4.1.2	Canals	Zeus site, Grass1 site
Helena	Fusion	Zeus site
Ilsa	Fusion	Zeus site

5.3.2 Connection with QCG Client Second Prototype

The first prototype of the integration between GridSpace and QCG was already described in Subsection 5.3.3 of D8.2. It supported the execution of example Multiscale applications. The second prototype: (1) improves stability and reliability, (2) increases the flexibility of the solution to enable execution of more varied multiscale applications.

Regarding stability and reliability the integration tests that were set up for the first prototype helped to detect a number of problems and errors. Most of them concerned the usage of GridFTP protocol for file staging in and out.

Running a generic MUSCLE-based application with QCG

The first prototype of QCG integration with GridSpace allowed to run example mutli-scale applications. However, it still lacked ability to support completely new, composed ad-hoc application (for example using MAD). This issue was caused by the fact that not all machines within QCG infrastructure had all required kernel implementations installed. In other words, the executing infrastructure should be aware of all dependencies that are required by each module and on which sites they can be resolved.

For example, a multiscale application may require kernels A,B and C. There are kernels A and B installed on machine X and kernel C installed on machine Y. Therefore, kernels A and

B can be run on machine X and kernel C on machine Y. However, infrastructure has to be aware what kernels are required by the application and where they are available.
Following improvements were made to solve aforementioned problem:

- in MaMe dependencies are only abstract module names instead of full paths to jars,
- MAD does not add the dependencies listed in MaMe registry to CxA scripts. Instead, they are passed as interpreter parameters to experiment,
- using list of dependencies for each kernel passed in interpreter parameters in experiment file EW constructs a proper job profile for QCG that has these dependencies listed. One should note that at this point dependencies are still seen as abstract names. Moreover, EW does not have to write the machine addresses that should be used for execution. These will be resolved by QCG (see next point),
- QCG broker accepts a job profile, reads required dependencies for each kernel and refers to its internal registry to decide which sites will meet the requirements.

Detailed technical description of GridSpace-QCG integration can be found in Annex A of D5.2 (living document update in M24).

5.3.3 Connection with AHE client second prototype

The connection of GridSpace with AHE was described in section 5.6.3 of D 4.2 (M18) of the project. Since then, the following improvements have been made:

- working execute operation, allowing for users to run code snippets with GridSpace web interface using AHE infrastructure,
- the execution process can be interrupted by user. In such case he or she can start another execution on AHE infrastructure,
- interpreters that are relevant for running sample multiscale applications have been configured, in particular the LAMMPS and CPMD tools installed on AHE machines are available through GridSpace .

In order to validate prepared integration mechanisms, especially the execute operation, we prepared integration tests for the AHE Executor. They invoke simple tasks on working AHE infrastructure, check whether they were successful and compare the actual output with expected one. Three main scenarios are taken into account:

- a task that writes only to standard output (not output files and therefore not staging in/out is required),
- a task that writes to a file - checks task execution and tests file stage out,
- a task that reads from a file and writes to another - as previous one but also checks staging in.

The tests are run periodically. Developers are immediately informed about any failures by email.

5.3.4 Browsing Results of MAPPER Applications

In MAPPER, result management is done in three aspects:

- storing, reading and browsing the actual file that was created as a result of experiment execution
- storing metadata that describes the experiment outputs
- browsing results and searching using associated metadata

As described in Section 8.1.2 of D8.1 two first areas were already partially supported by the first prototype by GridSpace. Since then, the metadata and file handling has been improved and more browsing capabilities where added. Below, we describe the changes in more details.

Physical file handling is provided by file browser in GridSpace by giving access to files that are present in a location related to specific experiment executor using SSH and GridFTP. The improvements in this field are introduced by a new provenance tool. It is capable of performing copies of experiment inputs and outputs and versioning them. This additional persistence allows users to read experiment results even if they were changed or deleted on original machine. The copying can be disabled for certain files (for example if they are too big to be transfered over the network). Currently two protocols are used for transferring the files:

- SSH - in this case the GridSpace server works as a intermediary between the executor storage and versioning storage. This is required because usually establishing a direct SSH connection between machines by a third party is not possible
- GridFTP - if the machine that runs the experiment supports this protocol, GridSpace uses the third-party-copy feature and files are copied directly between source machine and target.

Metadata handling. A new provenance system significantly expands the metadata that is available for experiment results. It collects detailed information about each execution of a GridSpace experiment, including what results (files) were created, when and by which execution steps (snippets). It also has links to their copies (mentioned in previous subsection) therefore it allows for easy searching and browsing. More details are available in section Provenance.

Browsing. Basic result browsing was already provided by GridSpace and its file browser. It enables browsing a typical directory-tree that is present on an infrastructure that runs the experiment. More sophisticated browser is provided for provenance and it is called QUaTRO.

It is a web-based application that allows a user to construct queries for RDF database containing provenance. The results of such query may be, among others, copies of files produced by experiment. Therefore, the provenance browser can be used as a experiment results browser.

5.4 MAPPER Provenance data collector and storage

As described in D8.2 in Section 5.4 the provenance system provides the following functionality:

- tracking - collecting metadata about experiment execution
- storing - providing a special database for holding metadata as well as creating snapshots of experiment inputs and outputs
- browsing and querying collected data using convenient user interfaces

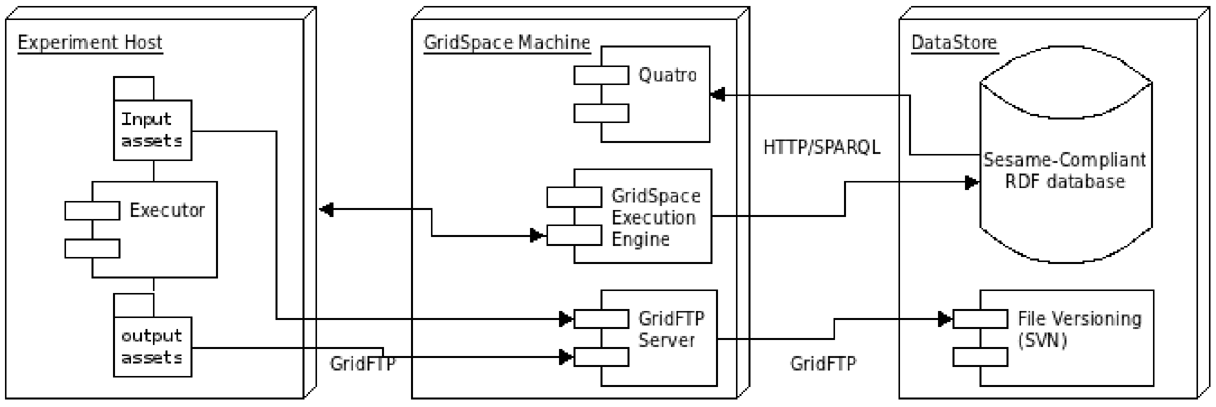


Fig. 7. The current provenance architecture.

The architecture of the provenance system is shown in Fig. 7. The Experiment Host is an infrastructure that runs the experiment using input and output files (input and output assets). The GridSpace machine hosts GridSpace Execution Engine described earlier in this deliverable together with GridFTP server for accepting file transfers from experiment host that are required to store and version experiment results. The provenance data can be browsed by QUaTRO (provenance interface). the DataStore server keeps provenance-related data, that is the metadata in RDF format and snapshots of experiment inputs and outputs.

Compared to previous design, this architecture differs in following points:

- There is no separate event collector. It is embedded in GridSpace execution engine and writes data directly to the database
- the MEF format concept was abandoned in favor of plain RDF graphs as it turned out to be too restrictive

- The GridFTP server is physically on the same machine as the GridSpace execution engine. This results from the fact that we needed the server to be installed on a machine that is already trusted by other sites

As mentioned before the event collector has been embedded into the GridSpace execution engine. It listens to the experiment execution process and at each step creates a provenance entry compliant with an ontology tailored for describing provenance in GridSpace. Following events are collected and written to a RDF database:

- experiment was started - contains date, experiment name and metadata
- snippet execution was started - contains date, experiment for which this snippet is run, inputs of this snippet (snapshots are created) and snippet code
- snippet execution was canceled - same as above but creates snapshots of snippet outputs instead of inputs. Additionally, the outputs are linked with the snippet that they were created by
- experiment execution finished - contains date and reference to same experiment metadata as the expriment start event

The ontology that is used for describing provenance is based on Open Provenance Model Vocabulary (OPMV)[6]. Most important part of it is presented in Fig. 8. The grayed entities come from the OPMV ontology. The black ones are the extension created for GridSpace.

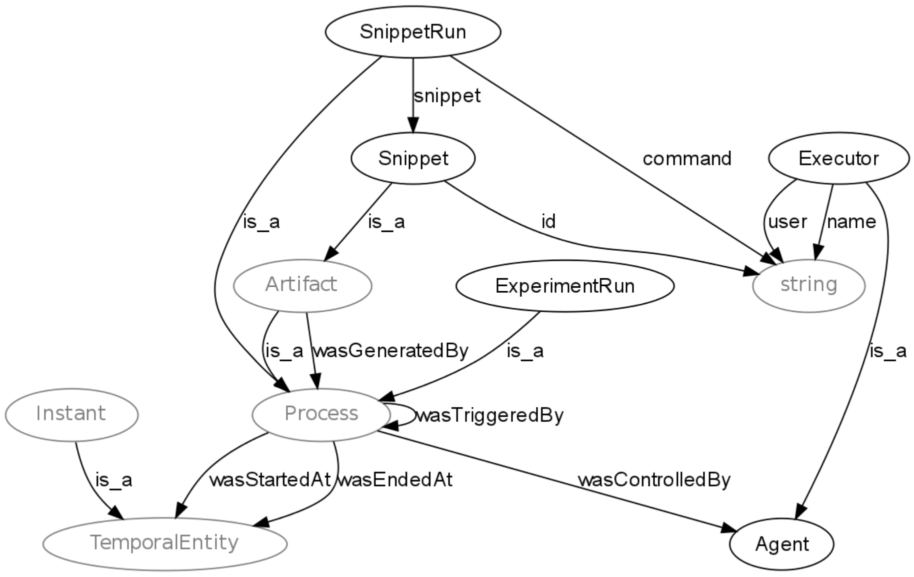


Fig. 8. Ontology used for describing provenance

[6] http://open-biomed.sourceforge.net/opmv/ns.html

A system uses 4store RDF database for keeping provenance medatada and a SVN repository for versioning input and output files of experiment steps (snapshots). Both services are currently operational. Additionally, creating snapshots of scripts was added.
A QUaTRO browser, available in the first prototype, has also been slightly improved by adding resource explorer.

6 Prototype availability

Below we present details concerning availability of prototypes of each tool.

6.1 jMML library

The XML Schema of MML is located at the public repository https://github.com/blootsvoets/xmml and the jMML library is located at https://github.com/blootsvoets/jmml.

6.2 MAD

The second prototype of MAD is available at https://gs2.mapper-project.eu/mad. The code is managed by Maven and is available in an SVN repository at https://gforge.cyfronet.pl/svn/gs2-utils/ibuilder. Access to the repository is possible to third-parties by using anonymous account of username anonsvn and password anonsvn.
A movie presenting MAD can be used can be seen on
http://dice.cyfronet.pl/projects/details/Mapper.

6.3 GridSpace Experiment Tools

GridSpace in its second prototype consists of several fine-grained modules. Experiment Workbench is a Java Enterprise Edition web application that comprises all the modules and can be considered as fully stand-alone. Rest of the modules are Java libraries that depend on each other. Each module source code resides in its separate SVN repository and is built as Maven artifact and then stored in dedicated Maven repository as indicated in Tab. 2.

Tab. 2. Availability of GridSpace software modules

GridSpace Module Name	Description	Source code access - SVN URL	Distribution – Maven artifact
Experiment Workbench	Web-based user interface for creating and running GridSpace experiments	https://gforge.cyfronet.pl/svn/gs2-utils/gs2portalproto/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/

			gs2/ew/
Core (Execution Engine)	Execution engine that handles submission of computational tasks in distributed environment. It uses executors in order to access computational power providers' services.	https://gforge.cyfronet.pl/svn/gs2-utils/cyfronet.gs2.core/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/core/
Executors	Specification of interface between Execution Engine and computational power providers.	https://gforge.cyfronet.pl/svn/gs2-utils/cyfronet.gs2.executors/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/executors/
Experiment	Basic data structures and object model of GridSpace experiments.	https://gforge.cyfronet.pl/svn/gs2-utils/cyfronet.gs2.experiment/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/experiment/
Result Management based on Provenance	GridSpace Provenance subsystem.	https://gforge.cyfronet.pl/svn/gs2-utils/provenance/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/provenance/
SSH Executor	Basic, built-in executor handling execution on sites and single hosts reachable through SSH protocol.	https://gforge.cyfronet.pl/svn/gs2-utils/cyfronet.gs2.ssh-executor/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/ssh-executor/
SPI	GridSpace Common programming utilities.	https://gforge.cyfronet.pl/svn/gs2-utils/cyfronet.gs2.spi/	http://dev-gs.cyfronet.pl/mvnrepo/cyfronet/gs2/spi/

Installation of Experiment Workbench is continuously available at https://gs2.mapper-project.eu/ew/ in its the most recent stable version. Associated interpreters registry is made accessible for external tools (such as MAD) through REST endpoint: https://gs2.mapper-project.eu/ew/gridspace. Experiment Workbench installation includes comprehensive tutorials and user manual. A movie presenting EW working with other tools can be seen on http://dice.cyfronet.pl/projects/details/Mapper.

6.4 MaMe

The current version of MaMe (both the Models Registry and the Experiment xMML Repository) is permanently provided under the link http://gs2.mapper-project.eu/mame/

It has completely open read access to every interested party - so any multiscale application developer or any scientist running such applications is able to read about available models and experiments, developed within the MAPPER Consortium. The write access is, however, blocked for such external people and only the members of the Project are presented with appropriate credentials to be able to modify the contents of the registry.

The same rule applies to the REST API for other tools: the read operations are open and the insert/delete/modify operations are secured. The exact explanations on how to use the REST API and what operations are available that way is given in the help sections of the main MaMe view (see the same MaMe public URL, given above).

The current revision of the MaMe source code is available at two distinct parts:

- https://gforge.cyfronet.pl/svn/sint/trunk/mame (the MaMe tool)
- https://gforge.cyfronet.pl/svn/sint/trunk/sintmodel_mapper (the semantic Mapper data model).

This distinction is made in order to keep the tool design clear and properly decomposed. The MaMe web application is being developed with the Semantic Integration methodology [GUBALA] and the core part of this requires the domain model to be an explicit, external element of the design (so the tool itself is kind of parametrized with the MAPPER domain model, based mainly on the xMML notation).

6.5 SBML toolbox

The codes are available in the svn repository
https://apps.man.poznan.pl/svn/sbml-toolbox/GRNApplication/src/

6.6 Provenance System

The main tool for browsing provenance data and experiment results is available here: https://gs2.cyfronet.pl/quatro/. A movie presenting how it can be used can be seen on http://dice.cyfronet.pl/projects/details/Mapper.

Source codes are available in following locations

- https://gforge.cyfronet.pl/svn/gs2-utils/provenance/trunk/ - source code for provenance event collector that is embedded in GridSpace
- https://gforge.cyfronet.pl/svn/quatro2/trunk/ - latest source code of the QUaTRO browser

7 Evaluation of efficiency of WP8 tools

According to description of Task 8.5 in DoW we measure efficiency of the WP8 tools by comparison of the effort of coupling applications by hand with the effort of coupling them using the tools. The main steps required for constructing the multiscale application either manually or using the tools are indicated in Fig. 9.

First, all submodels of single scale phenomena have to be created. These submodels have to be appropriately described to be composed with each other to form the multiscale application. Also, the corresponding software modules that implement submodels has to be designed. Having created single scale models, as well as the required scale bridging methods, they should be composed into the full multiscale applications. Firstly, this is done on the conceptual level. Next, the connection scheme has to be described. Finally, the application has to be executed on resources that fulfill requirements of all individual components. After execution, the results have to be fetched and presented to the user. All these steps can be done manually.

The WP8 tools support development of multiscale applications and facilitate their execution in abovementioned the process. They facilitate visual joining of modules and automatic generation of the connection scheme from this visual view. They provide sharing of modules and descriptions of their connections and reusing modules in different configurations. Having stored all required data, it is possible to automate the production of an executable from created connection scheme and facilitate choosing required resources from the single entry

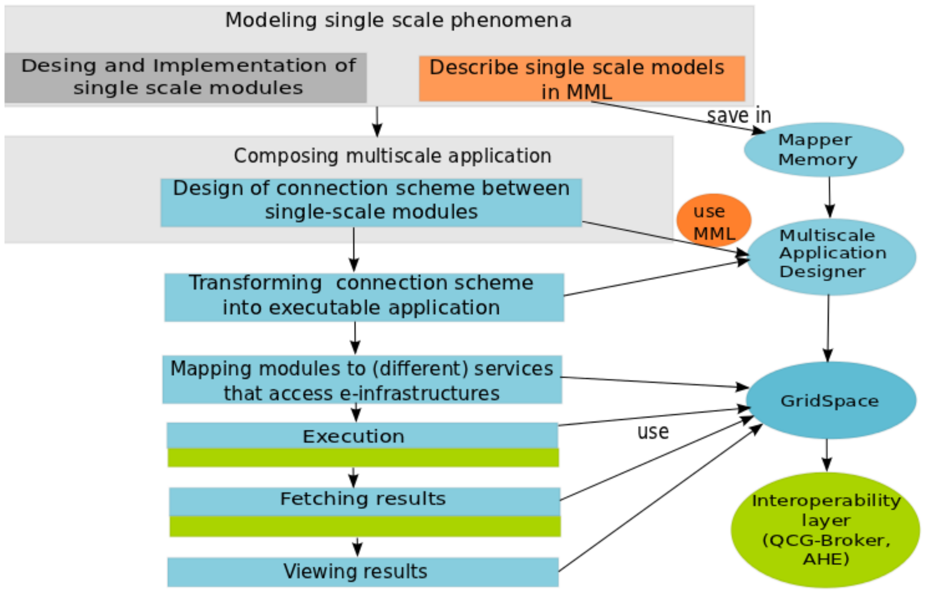

Modeling single scale phenomena
Desing and Implementation of single scale modules
Describe single scale models in MML
save in
Mapper Memory
Composing multiscale application
Design of connection scheme between single-scale modules
use MML
Multiscale Application Designer
Transforming connection scheme into executable application
Mapping modules to (different) services that access e-infrastructures
GridSpace
Execution
use
Fetching results
Interoperability layer (QCG-Broker, AHE)
Viewing results

Fig. 9. Process of constructing multiscale application - the steps of the process are indicated as rectangles; supporting tools are indicated as circles. The following colours were used: MML support - orange, tools - blue, external services - green.

point. After the execution, the output data are fetched and presented to the user. Moreover, history and provenance of produced results are also provided to enable other scientists to search, compare, evaluate and possibly validate results. In Tab. 3 we summarize actions required to create and execute the multiscale applications as described above. For each of the actions, we describe if and how it can be supported by the tools. For most cases, actions are facilitated by providing interactive interface to the user. If the action can be fully automated, we also provide the estimated time of the action.

Tab. 3. Multiscale application creation steps. For every step, we describe if and how it can be supported by the tools instead of a user manual work.

Action	Tools Support
conceptual modeling of single scale phenomena	done by a user
description of single scale models	for regular applications the description is registered to MaMe by using interactive user interface
design and implementation of single scale modules	done by a user; can be done using specyfic frameworks such as MUSCLE
conceptual modeling multiscale phenomena	done by a user
design of connection scheme between singe scale modules	interactive visual design in MAD; the previously designed connection schemes can also be loaded from xMML repository (few milliseconds); automatic generation of connection scheme (few milliseconds)
preparation of executable application from connection scheme	automatic generation of Experiment by MAD (few milliseconds) - this assumes that implementation of single scale modules are already available
mapping modules to (possibly different) external services that access e-infrastructures; setting parameters of these services	done by a user from the single web interface (GridSpace EW)- interactive process that usually takes from a few seconds to a few minutes.
execution of modules	goal is to facilitate application run, not improve performance -initialization of execution is done from a single web interface by pressing the run button; for ssh accessible resources there is a possibility of monitoring standard error and output from GridSpace EW that

	enables user on-line interaction with running modules; convenient usage of features underlying services (QCG-Broker, AHE)
fetching results	automatically fetched and visible in GridSpace EW by means of standard protocols - ssh and GridFTP (time depends on data size and protocol performance)
viewing results	visible in GridSpace EW; additionally QUaTRO browser can search created copies of experiment results
viewing provenance results	QUaTRO browsing and searching provenance data

The tools provide convenient initialization of execution, the possibility of monitoring standard error and output as well as easy browsing of results. The tools themselves do not focus on improving applications performance, but they facilitate usage of the underlying services (i.e. QCG-Broker and AHE) that can also support this goal. Moreover, the performance of the application is not affected by the tools usage as from the application point of view it is executed on remote resources using standard ssh protocol and available service clients (for QCG-Broker see section 5.3.2 and for AHE executor see Section 5.3.3). The tools are transparent for application execution.

To measure tools efficiency we have used metrics defined in DoW:

- **user experience with new MAPPER tools measured by feedback forms**: During the first seasonal MAPPER school http://www.mapper-project.eu/web/guest/first-seasonal-school we have measured usability of MaMe, MAD and GridSpace Experiment Workbench tools based on [BROOKE]. The obtained average SUS score for the tools was 68 points (for 100 possible; standard deviation was 18) . The average was calculated from answers from most active 10 participants. The most common request of the users was to allow changing parameters of application submodules directly in MAD which is done in second prototype (See Section 5.1.2).
- **statistics of successful execution of complete multi-scale simulations.** Currently we are applying our tools within the MAPPER project community - all xMML of applications are stored in xMML repository and are available through MAD https://gs2.mapper-project.eu/mad (press "open from repository" button). The status is described in Tab. 4. Details can be found in D 7.2

Tab. 4. Status of using WP8 tools in MAPPER applications

Application name	Field	Status of using WP8 tools
Instent	physiology	modules registered in MaMe, connected in

restenosis		MAD, executed in EW -using SSH and QCG executors; succesfully applied and described in [BORGDORFFb]
Irrigation canals	hydrology	modules registered in MaMe, connected in MAD, executed in EW -using SSH and QCG executors; succesfully applied in [BELGACEM]- used as a basis for a tutorial used during first seasonal MAPPER school in London http://www.mapper-project.eu/web/guest/mad-mame-ew
clay-polymer nanocomposites	Nano-material science	modules registered in MaMe, connected in MAD, executed in EW by SSH executor and AHE executor
reverse engineering of gene-regulatory networks	Computational Biology)	modules registered in MaMe, connected in MAD, tests of execution in EW
equilibrium-stability	fusion	modules registered in MaMe, connected in MAD, executed in EW
transport turbulence equilibrium workflows	fusion	modules registered in MaMe, connected in MAD, executed in EW
Heme LB	physiology	Modules registered in MaMe; application is currently using of high performance MPWide communication library, which is planned to be integrated with MUSCLE (supported by GridSpace) in the third year. As a result, usage of MAD and GridSpace is expected in the third year.

- **number of single-scale models incorporated and used within MAPPER infrastructure** measured by taking information from models' registry developed in Task 8.2: At present 20 single-scale models, 25 mappers and two filters are already registered in the model registry (MaMe), representing almost all MAPPER applications in addition to a test application. This number can be monitored online at http://gs2.mapper-project.eu/mame

- **number of new scientific results from applications created by MAPPER tools measured by number of publications in well recognized journals/conferences;** List of publications:
    - using WP8 tools with In-stent restenosis application [BORGDORFb]
    - using WP8 tools with clay-polymer nanocomposites application [GROEN][SUTER]
    - using WP8 tools with Irrigation Canals application [BELGACEM]
    - the paper on WP8 tools [RYCERZ]
- **mean time required to train a new user to use MAPPER tools measured during Seasonal Schools in task 2.4.** The tools tutorial done during first MAPPER seasonal school[7] was successful and consisted of 30 minutes presentation and 60 minutes hands-on exercises available on http://www.mapper-project.eu/web/guest/mad-mame-ew. There were no major problems in using the tools by school participants according to their responses to SUS survey. In Fig. 10 we show their answers to the sample questions of that survey showing their opinion if it was easy to learn using the tools.

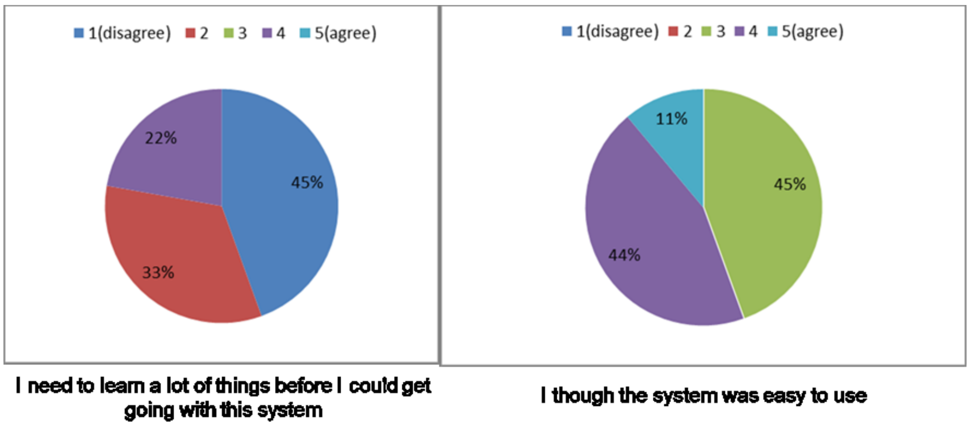


Fig. 10. Sample user answers to the SUS questions during first seasonal MAPPER school.

8 Conclusions

This deliverable presents the second prototype of multiscale programming and execution tools. It shows the current status of implementation according to the design presented in D 8.1. and changes in comparison to first prototype described in D8.2.

7 http://www.mapper-project.eu/web/guest/first-seasonal-school

We conclude that the proposed environment has proven to be quite useful and can help scientists to build their multiscale applications as described in Section 7. The use of common description language (MML) enables development of different applications from a single set of modules and switching between different versions of modules offering specific features. The accessibility of web based tools enables sharing of applications among scientists working on the same area of expertise. Additionally, support for interactivity (i.e. user interaction during application execution) is provided. As can be seen in the usage of tools within MAPPER project community is successful in various research fields.
The current application status can be found in D 7.2. In the next years of the project the presented prototype will be enhanced to provide full functionality described in D 8.1.

9 References

[BELGACEM] M. Ben Belgacem, B. Chopard and A. Parmigiani "Coupling method for building a network of irrigation canals on distributed computing environment" to be published in Proceedings of 10th International Conference on Cellular Automata for Research and Industry, ACRI 2012, Santorini Island, Greece, September 24-27, 2012. Series: Lecture Notes in Computer Science, Vol. 7495
[BORGDORFFa] Bergdorf, J., Falcone, J.-L., Lorenz, E., Bona-Casas, C., Chopard, B., and Hoekstra, A. G. Foundations of Distributed Multiscale Computing: Formalization, Specification, Analysis and Execution. Journal of Parallel and Distributed Computing, submitted, 1–31.
[BORGDORFb] Joris Borgdorff, Carles Bona-Casas, Mariusz Mamonski, Krzysztof Kurowski, Tomasz Piontek, Bartosz Bosak, Katarzyna Rycerz, Eryk Ciepiela, Tomasz Gubala, Daniel Harezlak, Marian Bubak, Eric Lorenz, Alfons G. Hoekstra: A Distributed Multiscale Computation of a Tightly Coupled Model Using the Multiscale Modeling Language. Procedia CS 9: 596-605 (2012)
[BROOKE] John Brooke Usability evaluation in industry, SUS - a quick and dirty usability scale (CRC Press, Boca Raton, FL), pp 189–194 (1996)
[ERMENTROUT] Ermentrout, B.: Simulating, Analyzing, and Animating Dynamical Systems: A Guide To Xppaut for Researchers and Students. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.
[FUNAHASHI] Funahashi, A., Morohashi, M., Kitano, H. & Tanimura, N.: CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. BIOSILICO 1, 159 – 162, 2003.

[GROEN] D. Groen, J. Borgdorff, S. Zasada, C. Bona-Casas, J. Hetherington, R. Nash, A. Hoekstra, P. Coveney, A Distributed Infrastructure for Multiscale Biomedical Simulations, accepted by the Virtual Physiological Human Conference 2012.

[GUBALA] T. Gubała, M. Bubak, P.M.A. Sloot; Semantic Integration of Collaborative Research Environments, in: M. Cannataro (Ed.) Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine and Healthcare, Chapter 26, pp. 514-530, Information Science Reference, 2009

[HOOPS] Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P. & Kummer, U.: COPASI – a COmplex PAthway Simulator, Bioinformatics, 22(24):3067–3074, 2006.

[MENDES] Mendes, P., Sha, W., Ye, K.: Artificial gene networks for objective comparison of analysis algorithms, Bioinformatics 2003, 19(90002): 122-129, 2003.

[RYCERZ] Katarzyna Rycerz, Eryk Ciepiela, Tomasz Gubala, Daniel Harezlak, Grzegorz Dyk, Marian Bubak, Joris Borgdorff and Alfons G. Hoekstra: An Environment for Programming and Execution of Multiscale Applications submitted to TOMACS journal

[SAVAGEAU] Savageau, M.A.: Biochemical systems analysis: A study of function and design in molecular biology, Addison-Wesley, Reading, Mass., 1976.

[SUTER] J. Suter, D. Groen, L. Kabalan and P. Coveney: Distributed Multiscale Simulations of Clay-Polymer Nanocomposites, Materials Research Symposium, San Francisco, United States of America, April 2012.

[VOHRADSKY] Vohradský, J. Neural network model of gene expression. The FASEB Journal 15, 846, 2001.

[VOIT] Voit, E.O. & Schwacke, J.H.: Understanding through modeling: A historical perspective and review of biochemical systems theory as a powerful tool for systems biology. Konopka, A.K. (editor), Systems biology: Principles, methods and concepts, pp. 28-77, 2007.