Hypermodelling Technology for Multiscale Simulations

Debora Testi[1], Daniele Giunchi[1], Xia Zhao[2], Gordon Clapworthy[2]

[1]SCS, Italy; [2]University of Bedfordshire, UK

Correspondence: d.testi@scsitaly.com, Via Parini 1, 40033 Casalecchio di Reno (BO), Italy

1. Introduction

The Virtual Physiological Human (VPH) [1] is a methodological and technological framework that, once established, will enable collaborative investigation of the human body as a single complex system, integrating the efforts of different biomedical disciplines, ranging from bioinformatics to medical imaging. This requires the interconnection of different computational models and simulations algorithms not only from different domains, but also at different scales. This is revealing the need, amongst other things, for a software technology that allows an easy inclusion of models into the whole system simulation and the execution of workflows (which will be complex in nature) for the choreography and orchestration of the different algorithms to be executed. Henceforth, we will refer to this as *hypermodelling technology*.

A hypermodel is a generic composition of existing predictive models into a new model. Each sub-model represents the process at a particular scale or within a specific sub-system; the hypermodel would thus represent the whole biological process across different scales and sub-systems. In order for it to be as general as possible, the hypermodelling technology allowing the hypermodel to be executed has to be designed to be flexible and modular so to adapt to different levels of patient specificity, and in particular to the amount of data available for the subject on a case by case basis. The aim of this paper is to describe a new ICT hypermodelling technology, its architecture, and its implementation together with a deployed used case.

2. The hypermodel

When designing a hypermodel technology, the most important aspects to be considered are:

- the possibility to connect models which can be developed with different software tools or libraries and which can be deployed on different hardware hosted in different physical locations;
- the communication between the models, which can be classified into:
  - control flow: the set of instructions that needs to be passed from one sub-model to another or to the system for its execution;
  - data flow: the data input-output of each sub-model.

Generalising, a typical scenario might have a pre- and post-processing software tool, which allows the user to load/import his/her data into a central repository, to pre-process them, to store them back to the data repository and which provides the possibility to launch and configure the hypermodel, and finally receive the results for eventual post-processing. This operation to launch the hypermodel execution should automatically suggest which sub-models can be run according to the available data and some pre-defined workflows, but the user may be allowed to choose a different workflow if the available data allow it. The application should also allow the authentication of the user into the system.

3. The hypermodelling technology

The hypermodelling technology described here has been implemented as an extension to the third version of the open-source Multimod Application Framework (MAF3) [2] to which specific components have been added. This software environment relies on the concept of the "wrapper", which is used not only to expose existing sub-model codes, but also for the other services composing the infrastructure. If the new module exposes the same API, its integration into the platform is straightforward and there is the possibility to add and/or remove other services according to the upcoming needs. Moreover, with this approach, each module may not know anything about the other modules. The architecture is represented in Figure 1, and its components are briefly described below.

- *Data repository*. Where input/output data are stored (including intermediate results coming from the sub-models); based on PhysiomeSpace services [3], it also includes services for data translation and transformation.
- *Authentication service*. This allows the user to be authenticated into the system with features for accounting and granting permissions only to certain parts of the hypermodel. The services have been implemented using the Biomed Town [4] identity-provider service and the Apache *mod_auth_tkt* module [5]. The mechanism is based on a session cookie with an expiry time, which is passed to all services.
- *Taverna*. The MAF3 hypermodel workflow is based on a client-server paradigm in which a Taverna Server represents the orchestrator of the hypermodel. Taverna [6] is an open source and domain-independent Workflow Management System – a suite of tools used to design and execute scientific workflows. On the client side, Taverna Workbench enables the user graphically to create, edit and run workflows locally, while the Taverna Server is the remote workflow execution service that enables a dedicated server to be set up for executing workflows remotely.

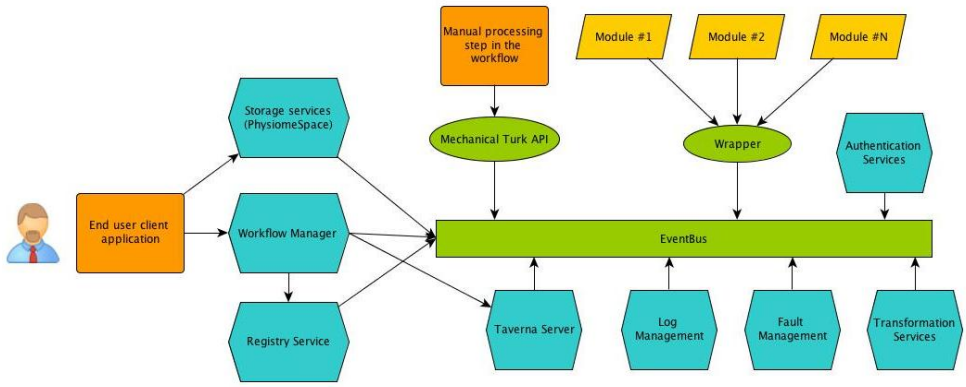

Figure 1. The hypermodelling technology architecture

- *Workflow service*. Along with the communication service (see below), the workflow service has a central role in the hypermodel, as it takes care of launching the sub-models in the correct order and obtaining information about when the sub-model ends its execution. It is also the interface between the client and the other hypermodel technology components. The implementation of the Workflow Manager is based on a Flask Python Microframework [7]. Its aim is also to share the active session among all of the modules/services that form part of the execution of a workflow: a Postgres SQL database is used to map a user to his/her running workflow and related session cookie; it also posts the workflow and input definition to the Taverna Server, and it launches the workflow execution, returning the workflow status and its output.
- *Communication service*. This is the message-exchange service between the different services and sub-models. It has been implemented as part of MAF3 (*mafEventBus*) and it allows objects to become a signal emitter or observer in order to communicate to each other in a dynamic way using the signal/slot mechanism implemented inside the Qt framework [8].
- *Wrapper*. Each service and/or sub-model is exposed with this software component, which allows each service to be considered as a black box without needing the details on how it is implemented or how it works internally. To execute an algorithm/module inside the hypermodel environment, the code needs to be wrapped as a MAF3 operation, which can then be started by the *mafEventBus* call coming from remote requestor. The wrapper starts the algorithm execution and passes all the parameters coming from the *mafEventBus* call. The current implementation of the wrapper is represented by the *mafAlgorithm* operation, which is registered into the MAF3 factory and allows a shell script to be started with relevant parameters. The Wrapper also takes care of downloading all input resources needed from the remote storage server, creating a database to store the log information, and eventually uploading output resources to the remote storage server.
- *Registry service*. This contains information on all the available sub-models to compose a workflow, together with information on the type of data in the input and output. This information includes the description and status of the module, the response time, topology, permission level, etc.
- *Log Management*. This module requests, from each other module, a detailed log by polling them periodically, and saves the result of the call in a database record. It gathers information that can be then queried by other modules (such as the Workflow Manager) and provides APIs for accessing the database.
- *Fault Management*. This manages exceptions or errors in the execution of the sub-models. Should one occur, the user can obtain the information (and any intermediate results) as early as possible. If an exception occurs in one model, then other models can free their resources or switch to other tasks (this can be managed in conjunction with the workflow service).
- *Mechanical Turk*. This is a common API, which is used to integrate into the hypermodel those sub-modules, that require human intervention during the execution.

4. A deployed use case

The mentioned above, this hypermodel technology has been used to deploy the workflows for the prediction of fractures in osteoporotic patients within the VPHOP project [9]. In this specific use case, the hypermodel is composed of five sub-models representing different spatial scales:

1. *Body Level*, which provides the boundary conditions for the organ level model and aims to deliver the patient's specific load spectrum;
2. *Organ Level*, which is always executed, as it calculates the risk of fracture for the chosen organ;
3. *Tissue Level*, which derives homogenised orthotropic mechanical properties at the continuum level;
4. *Cell Level*, which evaluates the effect of remodelling and/or pharmaceutical treatment;
5. *Constituent Level*, which predicts the effect of surgical augmentation treatment.

The VPHOP models require high performance computing services in order to run within a reasonable time frame (of the order of a couple of days), so all the associated algorithmic and simulation models have been installed on a HPC facility (IBM-PLX, http://www.cineca.it/en/hardware/ibm-plx) and wrapped as previously described. The wrapped modules are of different types: commercial or open-source codes (such as Ansys, Octave, etc.), in-house algorithms or databases. The user can launch the hypermodel execution from end-user interfaces specifically designed for clinicians (via a web-browser) or for researchers (with a client application which also allows the monitoring of the execution of a single model). The hypermodelling technology presented is currently in use by VPHOP partners in order to process the patients' data collected within the project from four clinical centres.

5. Discussions

The creation of domain specific workflows and their execution have been topics of high interest in multiscale modelling in the last years. A number of technological solutions are thus becoming available like those provided by the MAPPER and Virolab projects [10-13]. In the designing phase, already available technologies have been compared to the MAF3-implementation proposed and implemented in VPHOP. MAF was then chosen has showing advantages in terms of memory allocation and CPU usage in the specific application domain.

However, the hypermodelling technology has been designed to be very general so as to make it possible to adopt it in other biomedical contexts. In order to completely achieve this, despite the successful results obtained within the VPHOP project, extensions and improvements will be needed in the future to manage, for example, the execution of highly coupled modules; a caching mechanism for the data might be added to the wrapper for this purpose. Moreover, the wrapping mechanism should be simplified so that it can be easily carried out by non-developers. It should be also considered that the tests performed were with modules hosted on a single hardware installation, even if the storage services were remote. Distributed communication has thus been tested only partially, and further work might be necessary on this aspect in the future.

Acknowledgements

This work was partially funded under the VPHOP EC funded project (FP7-223865).

References

1. Virtual Physiological Human, http://en.wikipedia.org/wiki/Virtual_Physiological_Human
2. Multimod Application Framework, http://www.openmaf.org
3. PhysiomeSpace data sharing service, http://www.physiomespace.com
4. Biomed Town community portal, http://www.biomedtown.org
5. Apache mod_auth_tkt, http://search.cpan.org/~gavinc/Apache-AuthTkt-0.08/AuthTkt.pmt
6. Taverna workflow management system, http://www.taverna.org.uk/
7. Qt, http://qt.nokia.com/
8. Flask micro framework, http://flask.pocoo.org/
9. VPHOP, Osteoporotic Virtual Physiological Human project, http://www.vph-op.eu
10. MAPPER project, http://www.mapper-project.eu
11. Virolab project, http://www.virolab.org
12. Borgdorff J, Lorenz E, Hoekstra AG, Falcone J-L, and Chopard B. A Principled Approach to Distributed Multiscale Computing, from Formalization to Execution. in e-Science Workshops (eScienceW), 2011 IEEE Seventh International Conference on. 2011.
13. Hoekstra AG and Coveney P. Towards Distributed Multiscale Computing for the VPH. in VPH2010. 2010.