Towards an Engineering Methodology for Multi-Model Scientific Simulations

Alessandro Margara*, Mauro Pezzè*† Igor V. Pivkin* and Mauro Santoro*
*Università della Svizzera italiana (USI), Lugano, Switzerland
†University of Milano Bicocca, Milano, Italy
Email: {alessandro.margara, mauro.pezze, igor.pivkin, mauro.santoro}@usi.ch

***Abstract*—Complex physical phenomena are characterized by sub-systems that continuously interact with each other, and that can be modeled with different computational models. To study such phenomena we need to integrate the heterogeneous computational models of the different sub-systems to precisely analyze the interactions between the various aspects that characterize the phenomenon as a whole.**

While efficient methods and consolidated software tools are available to build and simulate single models, the problem of devising a general and effective approach to integrate heterogeneous models has been studied only recently and is still largely an open issue.

In this paper, we propose an engineering methodology to automate the process of integrating heterogeneous computational models. The methodology is based on the novel idea of capturing the relevant information about the different models and their integration strategies by means of meta-data that can be used to automatically generate an efficient integration framework for the specific set of models and interactions.

In this position paper we discuss the various aspects of the integration problem, highlight the limits of the current solutions and characterize the novel methodology by means of a concrete biological case study.

I. Introduction

Computational science focuses on building models and analysis techniques to simulate and study complex physical phenomena. Often, such phenomena are characterized by several aspects and sub-systems. Each sub-system can be best captured with a computational model that may substantially differ from the models suitable to analyze other sub-systems. Models may work at a different scale, for example, a large number of biological systems involve physical, chemical and mechanical processes interacting across diverse spatiotemporal scales. Moreover, heterogeneous types of models may be required to represent each aspect of the phenomenon under analysis, for example, biological systems may involve continuum-based models as well as particle-based models.

While efficient methods and consolidated software tools are available to solve each model in isolation, devising an effective way to integrate multiple models vastly remains an open issue.

In computational science, models are usually integrated with ad-hoc solutions that apply only to the specific target models. Such integration encodes a significant amount of relevant knowledge about the models, their communication strategies and the software and hardware infrastructure used for simulation. However, such information remains hidden in the implementation details of the specific solution, and thus cannot be reused to integrate other models.

General integration strategies and middleware platforms abstract away from application domain details that are essential when dealing with the integration of computational models, such as platform-dependent performance requirements that characterize computational simulations. Thus, general solution may not be effective in the context of computational science.

As acknowledged in the literature, a methodology to support an efficient and seamless integration of heterogeneous models and model simulation tools is needed to speed up, ease and automate the integration process [1]. The recent interest of the scientific community has led to the definition of some integration strategies and platforms [2], [3], [4]. Current solutions often consider single aspects of the integration problem, such as the presence of multi-scale model, or work only with some types of models, such as particle-based models, and do not generalize to the whole set of aspects yet.

A general integration methodology requires both a clear definition of the interactions between the different computational models and an efficient integration of the software tools used for simulating the different models. In particular, it requires detailed information about (*i*) the format and semantics of the data elements exchanged among the different simulation and analysis tools, (*ii*) the specification of when and how the communication of information takes place, i.e., the communication protocol, and (*iii*) the definition of how the different tools co-exist in a coherent platform deployment, i.e., how the different tasks are distributed and scheduled over the computational nodes.

This is a arguably a complex task, which requires (*i*) a deep understanding of the phenomenon under analysis and its modeling, (*ii*) a precise knowledge of the computational tools and infrastructure used for simulation and (*iii*) a comprehensive vision on the different choices in terms of communication, deployment and scheduling of computational tasks that may significantly impact on the efficiency of the overall solution.

In this paper, we introduce a methodology for automatically integrating heterogeneous computational models. The core idea of the proposed methodology is the definition of a suitable abstraction that captures the details required to integrate heterogeneous models and that we defined as *meta-data*. The new meta-data driven integration methodology is grounded on three key pillars: (*i*) meta-data information to

describe the computational models and the aspects related to their integration; (*ii*) a standardized middleware layer to enable inter-model communication; (*iii*) the automated generation, design, configuration and deployment of integrated and multi-model simulations for the specific problem.

To meet the needs of computational multi-model simulation and analysis, our methodology must fulfill two conflicting requirements: (*i*) it has to be *flexible* and *general*, to capture different types of models and integration strategies, and (*ii*) must produce *efficient* simulation infrastructures, with limited overhead with respect to ad-hoc solutions. Our goal is to find a good balance between these two requirements.

The remainder of this paper is organized as follows. Section II surveys existing approaches to the problem of model integration and discusses engineering issues in the definition of a general solution; Section III presents our methodology based on meta-data and exemplifies it using a case study from system biology. Finally, Section IV concludes the paper presenting current and future research directions.

II. Background and Related Work

Currently, computational models are developed with ad-hoc approaches, which lack mature software engineering methodologies and development processes [1]. Finding a suitable software development process for computational models remains an open problem, which requires to identify and understand the characteristics of scientific software tools [5].

The development and integration of computational models is vastly delegated to computational scientists that have a deep understanding of the application domain. Computational scientists internalize a model of software development, but may neither realize nor make explicit the contextual factors that make such model successful [6].

In this paper we focus on the problem of defining a methodology to automatically integrate heterogeneous computational models. Devising a suitable methodology for the integration of heterogeneous models involves finding a good trade-off between generality of the methodology and efficiency of the integrated simulation it produces [7].

The need for ease of development, reusability and faster adaptation of existing solutions to new scientific problems is driving computational scientists to the definition of new integration strategies. The recent attempts have introduced integration frameworks, domain specific languages or integration libraries.

The most prominent integration framework is the computational solution and the software and services developed within the MAPPER project to enable the integration of models at different spatiotemporal scales across disciplines [2]. Some recent work [8] propose languages to abstract all the information required to integrate models at different scales. Several libraries and toolkits have been developed to enable an efficient integration of either homogeneous or heterogenous models at different scales [3], [9], [10], [11], [12], [13], [14].

Our methodology attempts to formalize the key factors that influence multi-model integration into a coherent framework of meta-data, by exploiting the guidelines defined by the above contributions to devise effective integration strategies.

An orthogonal research line focuses on the definition of a software development process for computational models. Bartlett proposes methodologies based on the Agile development processes to facilitate the integration of software written by different expert groups [15].

III. A Meta-Data Driven Methodology

This section presents a novel approach to define an effective methodology for the integration of heterogeneous computational models. The distinctive feature of our methodology consists in introducing the concept of meta-data to capture the key features of the models, their simulation and their interactions, thus enabling the automated generation and deployment of an integrated simulation.

A. Overview

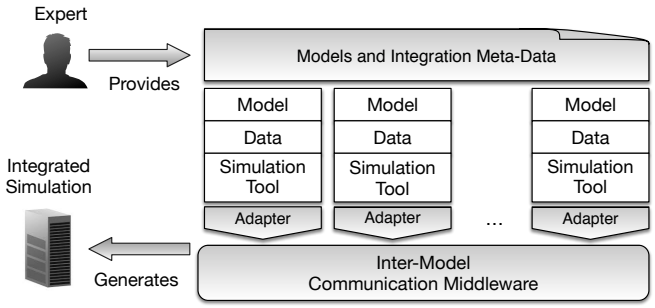

Fig. 1: Meta-data driven methodology: an overview

Fig. 1 shows the main elements that characterize our methodology: the grey boxes represent the core components of our methodology that interact with computational models, data and simulation tools (white boxes).

Domain experts *provide* the knowledge about the models involved in the simulation and the constraints on their interactions and execution in the form of *meta-data*. The methodology standardizes the inter-model communication by means of a *communication middleware* that mediates the interactions across models and a set of *adapters* that enable the reuse of simulation tools, thus enabling the automated *generation* of an integrated simulation.

Models and integration meta-data.

Over the years, computational scientists have built efficient simulation tools, each of them optimized for a specific kind of models. In absence of an engineering methodology, the integration of such tools is usually designed ad-hoc for the specific problem at hand.

In our methodology, domain experts shall characterize the models used in simulation in the form of meta-data, thus providing the information required to integrate existing tools within a coherent workflow. The meta-data include the following types:

Computational models that describe each computational model involved in the integrated simulation. In particular the following information will be encoded within this meta-data category: the *type and scale of model* adopted –for instance, continuum-based model, particle-based model, agent-based model and their specific spatial temporal scale–; the content and format of the *data* they consume, process and generate; the *semantics* of the data they process –for instance, in a particle-based model, the semantics specify that particles are represented using type, position and velocity in a tridimensional space–.

Workflow that specifies how the different computational models are organized in an execution flow that takes into account the spatial region they cover and their time scale. For example, two models can be simulated in iterative steps, that is, the result of a model is used as an input parameter for the other model in the next iteration. The workflow also constraints the architecture of the integrated simulation –for instance, some simulation steps can run in parallel only if they do not depend on each other to proceed–.

Data flow that specifies the data items that are exchanged between models during the execution of the workflow. For example, this can be useful to gain information on the volume of data exchanged between models to optimize their integrated deployment.
The data flow also defines the *translation* functions that transform the data produced by one model into data that can be understood and processed by another model –for instance, information about particles in a particle-based model are aggregated into continuum functions analyzed by a continuum-based model–.

Communicating conditions that specify when the communication takes place. For example, the communication can be interval-driven, when models exchange data after a specified number of simulation steps, or event-driven, when models exchange data upon specific event occurrences. This can be useful to gain information on the frequency of data exchanged and on the computational cost of each model, to optimize the allocation of the available resources across simulation tools.

The classes of meta-data presented above should be sufficient to fully specify the interaction between computational models. Our methodology enables computational scientists to provide additional meta-data to characterize the hardware infrastructure used for simulation, and to define how software can take advantage of specific hardware components, such as GPUs. In particular, the additional meta-data are organized within the following categories:

Computational infrastructure that describes the details of the hardware infrastructure in terms of computational and communication resources, as well as availability of specific hardware accelerators.

Hardware constraints that specify whether specific hardware is required or suggested in a specific simulation step. This information can be used to optimize the allocation of computational resources to individual software modules.

Data locality information that specifies locality in data production and consumption. This may enable optimizations in the deployment, for instance, allocating software modules that communicate frequently on well connected computational resources.

Inter-model communication middleware and Adapters.

Meta-data define which data elements are exchanged between computational models, how such elements are translated from one model to another and when the communication takes place. In other words, they define the communication protocol that governs the interaction between models.

To concretely implement such protocol, our methodology relies on a middleware that acts as a communication bus between simulation tools, and standardizes the communication interfaces. To enable the reuse of well established simulation tools, we rely on adapters that should be implemented for each specific simulation tool once, and that translate from a tool-specific data format to the standard middleware interface. The middleware and the adapters simply define a common playground for executing models without adding data processing and communication steps with respect to ad-hoc solutions. Therefore, we do not expect them to introduce visible overhead.

Adapters can integrate within a simulation tool at various levels. Depending on the integration level, they may enable different degrees of optimization in the integrated simulation. For example, if an adapter exposes functions to directly access its results from the main memory, then it enables for faster data communication with software components that run on the same physical node, allowing for data locality optimizations.

B. Case Study

We are currently studying and evaluating the feasibility and benefits of our methodology within a real world project on angiogenesis, which is the process of generating new blood vessels from pre-existing vasculature. In this context, our methodology aims to use meta-data to integrate models of the development of vascular networks with models of the blood flow and blood components within veins. This scenario presents three main integration challenges: (*i*) different aspects of the domain are encoded with different types of models, (*ii*) different models are simulated with different software tools, each of them presenting specific deployment and execution strategies, (*iii*) different models work at different spatiotemporal scale.

We analyzed the integration problems that arise in the angiogenesis case study jointly with domain experts, focusing

on the domain specific information about the computational models, their interaction and their simulation, and we organize this information according to the conceptual framework presented in the previous section. Next, we show the results of this analysis, presenting the information included within each meta-data category. We are currently investigating the usage of different meta-data formats suitable to encode such information.

The project adopts two *types of models* that consider different types of *data*: a continuum-based model computes the geometry of the vascular network, relying on the MRAG simulation tool [16], a particle-based model describes the flow through the network, relying on the LAMMPS tool for simulation [17], a highly-scalable MPI-based code capable of exploiting GPUs.

The time scale of these two models are very different and therefore can be separated during simulation. Development of vascular network takes hours, while the passage of individual red blood cells through the network takes seconds. Thus, the two models can be coupled through the exchange of boundary conditions, as discussed later.

In terms of *semantics* of data, particles are represented using their type –fluid, blood cells or vessel walls–, position and velocity in a tridimensional space, while the continuum model uses partial differential equations that model the effects of oxygen and nutrient diffusion, and the proliferation and propagation of endothelial cells that form the vessel walls.

Both models partition the domain of analysis into sub-domains, with an instance of MRAG and LAMMPS being responsible for each sub-domain, potentially on different processing nodes. LAMMPS works at a finer granularity than MRAG, and the division of the domain is not uniform, for processing efficiency. Interactions between sub-domains are governed by boundary conditions.

The *workflow* defines an iterative execution of LAMMPS and MRAG within each sub-domain. MRAG adopts the flow information provided by LAMMPS to update the shape of the network. LAMMPS performs simulation according to the geometry of the vascular network as computed by MRAG.

Data flow between LAMMPS and MRAG occurs within each sub-domain. LAMMPS requires data relative to the geometry of the vascular network and parameters in the surrounding tissues to produce data about the distribution of oxygen and nutrients in the tissue, as well as data about the distribution of wall shear stress. Within each computational iteration, both the values of the geometry of the network and the parameters in the surrounding tissue are assumed to be constant. MRAG takes into account updated distribution of oxygen and nutrients in the tissue, as well as distribution of WSS, which will be used to further advance development of the network. Meta-data include particle-to-mesh and mesh-to-particle *translation* functions to enable the communication of the two computational model.

The *communicating conditions* are event-based: the data between models is transferred when specific conditions occur.

The *computational infrastructure* used within the project consists of PIZ-DAINT and PIZ-DORA[1] at the Swiss National Super Computer center. While the project does not currently exploit the support for GPUs in LAMMPS, the plans to take advantage of GPUs will introduce *hardware constraints* on the deployment of the simulations. The nature of the problem does not enable optimizations related to *data locality* across different models. Indeed, each instance of MRAG spans a domain that is several order of magnitude larger than an individual region analyzed by an instance of LAMMPS.

This case study confirmed that synthesizing and abstracting meta-data information about computational models and their interaction is indeed feasible. It shows that the classes of meta-data presented in the previous section represent a valid starting point to drive our methodology that captures relevant aspects about multi-model integration.

We are collaborating with domain expert in angiogenesis to concretely compute the cost of specifying meta-data. We plan to compare different meta-data formats at different levels of detail and granularity to identify a convenient cost benefit trade-off of our methodology.

IV. Conclusions and Future Work

In this paper we tackled the problem of integrating heterogeneous computational models within a simulation by proposing an engineering methodology whose prominent novelty is the introduction of meta-data to capture and encode the salient elements that drive an efficient integration of models. The methodology aims to balance the generality of the integration approach with the efficiency of the integrated simulation.

In this position paper we presented the main elements of the methodology, whose validity is confirmed by our preliminary investigation. Our research agenda includes four main items required for the development of an integration platform.

First, we are extending our analysis of computational models, to (*i*) better define the boundaries of our methodology and (*ii*) refine the set of features that serve to fully characterize a computational model and its interaction with other models.

Second, we are investigating suitable languages and formalisms to encode meta-data information. We foresee the use of layered languages, in which only minimal information is required to define a working integrated simulation. Additional information can be provided, if needed, to enable finer grained tuning and optimizations.

Third, we are defining a suitable interface for the inter-model communication middleware. Such interface has to be simple enough to enable a fast implementation of working adapters for existing simulation tools. Yet, it needs to provide enough flexibility to support specific optimizations when needed. For example, it should enable computational scientists to provide guidelines on low level details such as memory layout and accesses.

Fourth, we are working on efficient algorithms to plan an optimal or nearly optimal allocation of resources based on the available meta-data information.

[1] http://www.cscs.ch/computers

REFERENCES

[1] J. C. Carver, "Software engineering for computational science and engineering," *Computing in Science & Engineering*, vol. 14, no. 2, 2012.

[2] S. J. Zasada, M. Mamonski, D. Groen, J. Borgdorff, I. Saverchenko, T. Piontek, K. Kurowski, and P. V. Coveney, "Distributed infrastructure for multiscale computing," in *Procs. of the International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2012.

[3] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. Ben Belgacem, B. Chopard, D. Groen, P. V. Coveney, and A. G. Hoekstra, "Distributed multiscale computing with muscle 2, the multiscale coupling library and environment," *Journal of Computational Science*, 2014.

[4] P. M. Sloot and A. G. Hoekstra, "Multi-scale modelling in computational biomedicine," *Briefings in bioinformatics*, 2009.

[5] C. Crabtree, A. Koru, C. Seaman, and H. Erdogmus, "An empirical characterization of scientific software development projects according to the boehm and turner model: A progress report," in *Procs. of the Workshop on Software Engineering for Computational Science and Engineering*, 2009.

[6] J. Segal, "Some challenges facing software engineers developing software for scientists," in *Procs. of the Workshop on Software Engineering for Computational Science and Engineering*. IEEE Computer Society, 2009.

[7] V. R. Basili, J. C. Carver, D. Cruzes, L. M. Hochstein, J. K. Hollingsworth, F. Shull, and M. V. Zelkowitz, "Understanding the high-performance-computing community: A software engineer's perspective," *IEEE Softw.*, vol. 25, no. 4, 2008.

[8] J.-L. Falcone, B. Chopard, and A. Hoekstra, "Mml: towards a multiscale modeling language," *Procedia Computer Science*, vol. 1, no. 1, 2010.

[9] E. T. Ong, J. W. Larson, B. Norris, R. L. Jacob, M. Tobis, and M. Steder, "A multilingual programming model for coupled systems," *International Journal for Multiscale Computational Engineering*, vol. 6, no. 1, 2008.

[10] I. Sbalzarini, J. Walther, M. Bergdorf, S. Hieber, E. Kotsalis, and P. Koumoutsakos, "Ppm a highly efficient parallel particlemesh library for the simulation of continuum systems," *Journal of Computational Physics*, vol. 215, no. 2, pp. 566 – 588, 2006.

[11] Z. Keming, K. Damevski, V. Venkatachalapathy, and S. G. Parker, "Scirun2: a cca framework for high performance computing," in *High-Level Parallel Programming Models and Supportive Environments, 2004. Procs. Ninth International Workshop on*, 2004.

[12] S. P. Zwart, S. Mcmillan, D. Heggie, J. Lombardi, P. Hut, S. Banerjee, H. Belkus, T. Fragos, J. Fregeau, M. Fuji, E. Gaburov, E. Glebbeek, D. Groen, S. Harfst, R. Izzard, M. Juri, S. Justham, P. Teuben, J. Bever, O. Yaron, and M. Zemp, "A multiphysics and multiscale software environment for modeling astrophysical systems," 2008.

[13] M. Bernaschi, S. Melchionna, S. Succi, M. Fyta, E. Kaxiras, and J. K. Sircar, "Muphy: A parallel multi physics/scale code for high performance bio-fluidic simulations," *Computer Physics Communications*, vol. 180, no. 9, 2009.

[14] P. Dadvand, R. Rossi, M. Gil, X. Martorell, J. Cotela, E. Juanpere, S. R. Idelsohn, and E. Oate, "Migration of a generic multi-physics framework to hpc environments," *Computers & Fluids*, vol. 80, 2013.

[15] R. A. Bartlett, "Integration strategies for computational science & engineering software," in *Procs. of the Workshop on Software Engineering for Computational Science and Engineering*. IEEE Computer Society, 2009, pp. 35–42.

[16] D. Rossinelli, B. Hejazialhosseini, D. G. Spampinato, and P. Koumoutsakos, "Multicore/multi-gpu accelerated simulations of multiphase compressible flows using wavelet adapted grids," *SIAM Journal on Scientific Computing*, vol. 33, no. 2, 2011.

[17] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of computational physics*, vol. 117, no. 1, 1995.