6th European Conference on Computational Mechanics (ECCM 6)
7th European Conference on Computational Fluid Dynamics (ECFD 7)
1115 June 2018, Glasgow, UK

MUPIF: MULTI-PHYSICS INTEGRATION PLATFORM

BOŘEK PATZÁK[1], VÍT ŠMILAUER[1] AND MARTIN HORÁK[1]

[1] Czech Technical University in Prague, Faculty of Civil Engineering, Department of Mechanics
Thákurova 7, 166 29 Prague 6, Czech Republic borek.patzak@fsv.cvut.cz,
vit.smialuer@fsv.cvut.cz, martin.horak@fsv.cvut.cz

Key words: Simulation platform, Integration, Distributed computing, Interoperability

Abstract. Paper describes the desing of distributed integration platform enabling to create complex coupled workflows from existing simulation tools. The main features as well as selected real world applications are presented.

1 INTRODUCTION

A reliable multi-scale and multi-physics numerical modeling requires including all relevant physical phenomena along the process chain and across multiple scales. The complexity of problems requires combination of knowledge from different fields. This brings in also the challenges for software engineering and design. There is a strong need for integration platforms, that enable inter-operable integration of existing simulations tools and databases into a complex simulation work-flows, providing capability to exchange information and efficiently use available computing resources.

The interoperability by definition implies standards. Traditional approaches have been based on syntactic interoperability, based on specific communication protocols and conversion tools. The more attractive are approaches based on semantic interoperability, where data are exchanged together with their meaning, which allows for automated machine interpretation, translation, and verification [1].

Integration platform should enable integration of existing simulation tools allowing to perform the data exchange and steering. Regarding the development in hardware and software technology in last decades, the integration platform should enable integration of distributed and parallel applications over the network in an efficient and scalable way, based on standardized communication protocols. Standard methods, such as "naming services" or a "service discovery" that allow individual component lookup based on name and/or services should be provided, as well as services for job and resource allocation on remote computing resources.

The aim of presented paper is to describe the design of distributed, object-oriented simulation platform MuPIF [2, 3] and illustrate its use on several use cases. The MuPIF is an open source project developed in Python [5, 6], licensed under LGPL license.

2 DESIGN OF THE PLATFORM

MuPIF is designed using object-oriented approach, where classes are introduced for each identified entity. The entities include, among others, individual simulation tools and exchanged data. The strength of object-oriented approach is in ability to express similarities between classes using inheritance, a concept allowing to derive one class from another by inheriting attributes and services of the parent class. Derived class can extend parent class by i) adding new attributes or methods and ii) provide its own implementation of inherited methods (specialization). In this way parent class can declare behavior or interface (in terms of defined services) that is common to all derived classes. Typically, the role of parent classes is only to declare the common interface, the individual methods are implemented by the derived classes representing specific objects, then such a parent class is called an abstract class. One can create a hierarchy of abstract classes allowing to define the interfaces in a hierarchical and structured way.

In MuPIF, the generic abstract classes are introduced for models (simulation tools) and generic data types, such as properties, spatial fields, time steps, etc. The definition of abstract interfaces for models as well as for high level data types is one of the unique features of the MuPIF. It allows to achieve true plug&play architecture, where individual application as well as data representations can be plugged into existing work-flows and be manipulated using the same interface. All classes are derived from top level abstract class, MuPIFObject, declaring a generic interface common to all components. It defines the services allowing to attach the additional information about the individual object, so called metadata. The metadata play important role, as they allow to track the origin of the data, information about data units, etc. Some metadata have to be defined by user or simulation tool, some can be automatically collected by the platform.

The interoperability in the MuPIF is achieved by standardization of application and data component interfaces, it is not relying on standardized data structures or protocols. Any existing data format can be plugged in and transparently used, provided the corresponding data interface is implemented. At present, the platform is being integrated into Business Decision Support System (BDSS) in the frame of EU H2020 Composelector project [4]. The aim of the framework is to demonstrate the business decision process during the selection of materials and manufacturing processes for which data are available. However, the developed BDSS platform will offer the possibility to explore new material designs by offering the possibility to investigate, "develop" and evaluate, in physical terms, new types of materials and check their potentials with regard to the required property profile and KPIs. In the Composelector project, the BDSS enables the actors from business and/or technical departments within the manufacturing organization to drive decisions on material modeling, whilst bringing the various modelling tools together within an automated and seamless workflow. A graphical web editor based on standard on Business Process Model and Notation (BPMN) and ESTECO technology [7] enables the modeling and execution of business workflows. It integrates the material information management system, GRANTA MI [8], an enterprise level, web-accessible platform for materials and process data/information to demonstrate the management of physical and

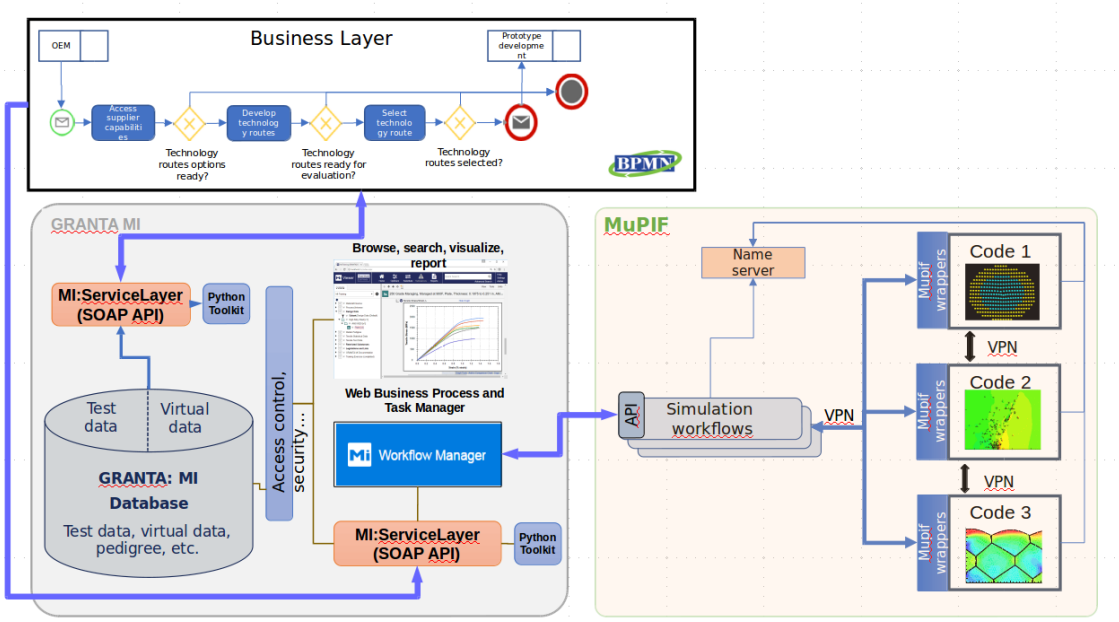

Figure 1: The Composelector BDSS schema

virtual data, reporting and decision-making for materials and process selection. MuPIF serves the simulation platform, where the simulation workflows are defined and executed.

2.1 APPLICATION INTERFACES

The application interface allows to perform data exchange with the simulation tool and steer its execution. The data exchange methods allow to set or get specific data component, that can represent simple data object, such as property, or more complex data objects, representing, for example, spatial fields, computational meshes, force field potentials, etc. The individual application are not supposed to interpret the raw data structures, but rather they interpret the data using the interfaces. of representing data. In this way, the individual simulation tools are abstracted from the details of particular data representation. The advantages of this concept are described in the next chapter. The steering services allow to update the simulation state for given time step, report critical time step, check simulation status, etc. Table 1 lists the most important methods of the application interface.

To connect an existing simulation tool to the platform, one needs to implement the Application interface. This can be implemented in a number of ways, depending on particular application. The implementation consists in implementing a derived class from abstract Application class (defining the application interface) and implementing the required methods accordingly. In general, two different approaches can be distinguished. The first one, called direct approach, is based on direct interaction with the application using its scripting interface or programming interface (requires to link the application as a library). The second option is to communicate with the simulation tool indirectly, usually using input/output files. In this case, the application interface records all settings

Table 1: Application interface definition (simplified)

Method	**Description**
Init (inFile, workdir)	Constructor. Initializes the application.
getDataComponent (dataCompID, time)	Returns data identified by its ID evaluated at given time.
setDataComponent (dataComp)	Registers the given data component in application.
solveStep (tStep, stageID=0, runInBackgr=False)	Solves the problem for a given time step. tStep is object representing solution time step.
isSolved ()	Returns true or false depending whether solve has completed when executed in background.
wait ()	Wait until solve is completed when executed in background.
finishStep (tstep)	Called after a global convergence within a solution time step.

of parameters and variables, and when the solution update is requested, the input file is produced, simulation tool is called and finally, the output file is parsed for output parameters. Up to now, several interfaces have been implemented, including commercial as well as open source simulation tools. Recent applications illustrated connections to Matlab, Comsol, X-stream, Micress and many other simulation tools. The simulation workflows define execution and data flow for complex simulation tasks combining individual simulation tools. The workflows themselves implement Application interface and this is another unique feature of MuPIF platform. This allows to regard and use complex workflows as black-box simulation tools. In addition, a hierarchy of simulation workflows can be naturally defined. Data interfaces

In case of data, the MuPIF is trying to abstract from particular data representation. This is achieved by defining the interfaces for data types as well. For example, consider a spatial fields describing variable with spatial distribution. It can be represented in a number of different ways, including mathematical formula as a function of spatial coordinates or interpolated set of spatially distributed point values, for example. However, it is possible to identify a typical operations, that could be expected from a spatial field, like for example, operation that evaluates the spatial field at given position. The identified operations then constitute the data type interface, allowing to manipulate all instances using the same interface. The data represented by objects bring in several advantages, as object is an encapsulation of raw data and methods operating on the data. Therefore simulation tool does not interpret the data itself, it is the role of the interface to declare the needed methods to interpret the data. The interface allows to abstract from particular representation of data, therefore allowing to naturally support different data storage formats. This allows to create specific data representations targeted for specific purpose, as voxel-based or vector-based micro-structure representations, for example, still

fully compatible with all the components. In many cases, the fact that simulation tools can work with different data representations abstracted by the common interface, can reduce the need for data transformations, which can also reduce numerical errors introduced during transformation. In addition, as the interface should be the only way how to operate with data, the interface can also abstract from actual storage of raw data, the data can be located in memory, in file, database, or even distributed over the network. The standardization in MuPIF is therefore focused on standardization of data interfaces rather than on standardization of data structures.

2.2 DISTRIBUTED DESIGN

Complex simulations are resource and time demanding. Distributed and parallel computing environments provide needed resources and computational power. Common feature of these environments are distributed data structures and concurrent processing on distributed processing nodes. This brings in an additional level of complexity that needs to be addressed. The important role of the simulation framework is to provide a communication mechanism that will take care of the network communication between the objects. An important feature, particularly for the end user, is transparency, hiding the details of remote communication to the user and allowing to manage local and remote objects using the same interface.

In MuPIF, the communication layer is built on a transparent distributed object system fully integrated into Python [5, 6]. It takes care of the network communication between the objects when they are distributed over different machines on the network, hiding all socket programming details. One just calls a method on a remote object as if it were a local object – the use of remote objects is (almost) transparent. This is achieved by the introduction of so-called proxies. A proxy is a special kind of object that acts as if it were the actual object. Proxies forward the calls to the remote objects, and pass the results back to the calling code. In this way, there is no difference between simulation workflows for local or distributed cases, except for the initialization, where, instead of creating local application interfaces, one has to connect to their remote counterparts. From the position of the end user, the transparent communication eases the development and maintenance. Security can be granted using VPN with the encryption and authorization services. The VPNs work on a lower level of communication (OSI Layer 2/3) by establishing "virtual" (existing on the top of other networks) network, where all nodes have the illusion of direct communication with other nodes through TCP or UDP, which have IP addresses assigned in the virtual network space. The VPN itself communicates through existing underlying networks, but this aspect is not visible to the nodes; it includes data encryption, compression, routing, but also authentication of clients which may connect to the VPN. Supporting tools for monitoring platform infrastructure are available.

3 APPLICATIONS

Recently, the MuPIF platform has been used in modeling fire impact on steel structures (combing CFD based fire simulation tool with nonlinear structural analysis) and

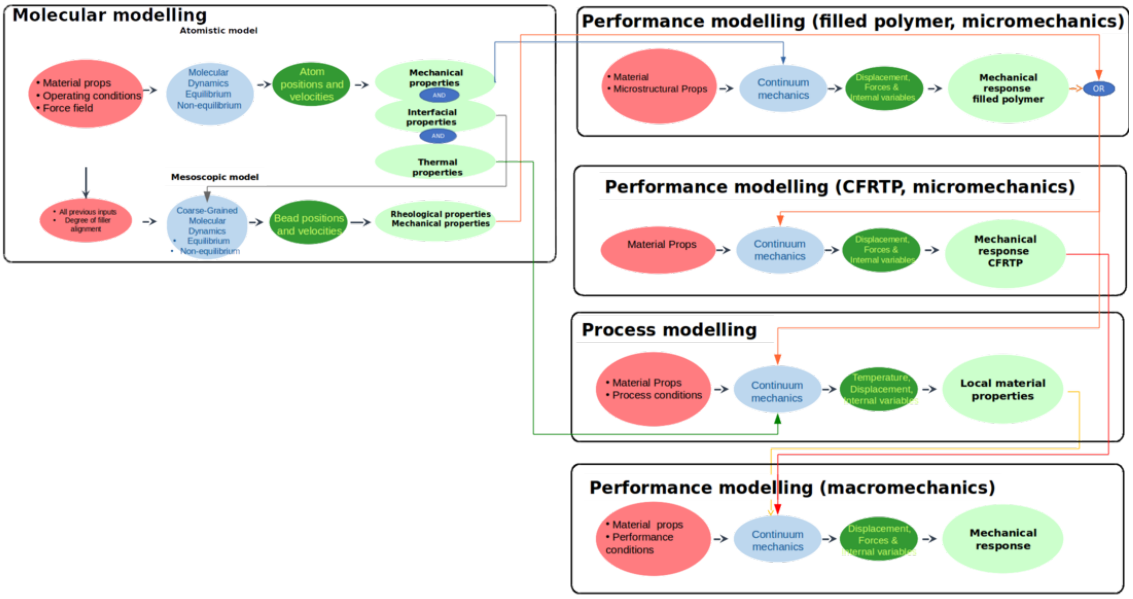

Figure 2: Moda diagram of the simulation workflow considered in Composelector project

in frame of EU 7FP MMP [9] to model production and performance of novel solar cells and phosphor LEDs. At present, it is used in the frame of EU H2020 Composelector to design innovative composite materials with applications in aerospace (design of composite airplane frame, see Fig. 2 for modeling workflow) and automotive industry (design of composite leaf springs and tires). The complex simulation workflows involve combination of several simulation tools starting on atomistic/molecular scale to determine basic material properties, bridging several resolution scales and ending up on component/structural scale, including commercial and open source modeling tools distributed and executed across different locations and computing resources.

We demonstrate MuPIF capabilities on a linked CFD-thermomechanical task, consisting of a steel beam placed into a furnace with gas burners, see the geometry in Fig. 3. The furnace is of dimensions 3.0 m x 4.0 m and 2.2 m height and heated by 8 natural gas burners. The simply-supported beam is made from steel I profile 3.8 m long, placed just under the furnace ceiling.

Two codes are used for the a linked simulation; Fire Dynamics Simulator software (FDS), developed at NIST [10] and OOFEM used for thermo-mechanical analysis, developed at Czech Technical University in Prague [11]. Figure 4 shows their interaction using MODA diagram coined by EMMC. MuPIF interconnects and orchestrates all the continuum models. The results were computed end exported to VTU files. Figure 5 left shows a cross-section through the middle of the furnace. One can see the two FDS meshes, one coarse on the whole furnace and one finer around the beam, which is necessary for the interpolation close to the ceiling. Figure 4 right provides detailed normal stress σ_x at time t=576 s.

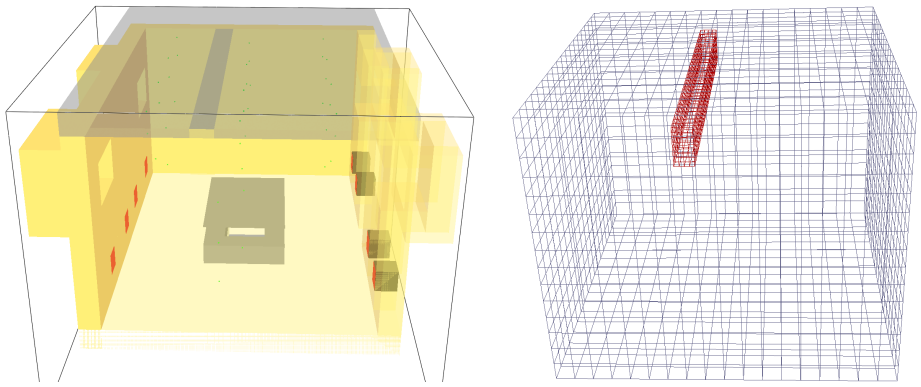

Figure 3: FDS furnace model from Smokeview (left) and meshes from FDS and OOFEM (right).

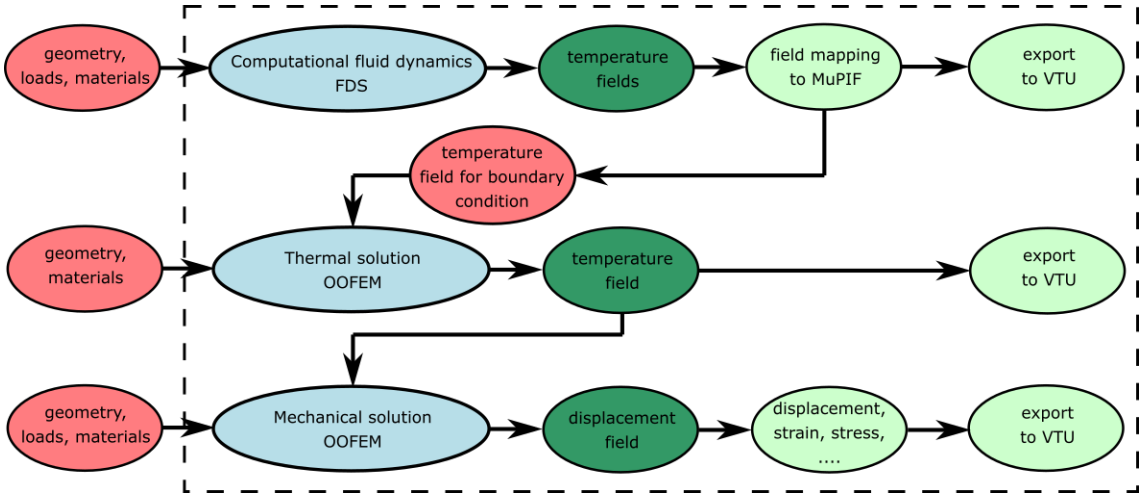

Figure 4: Data flow using MODA diagram.

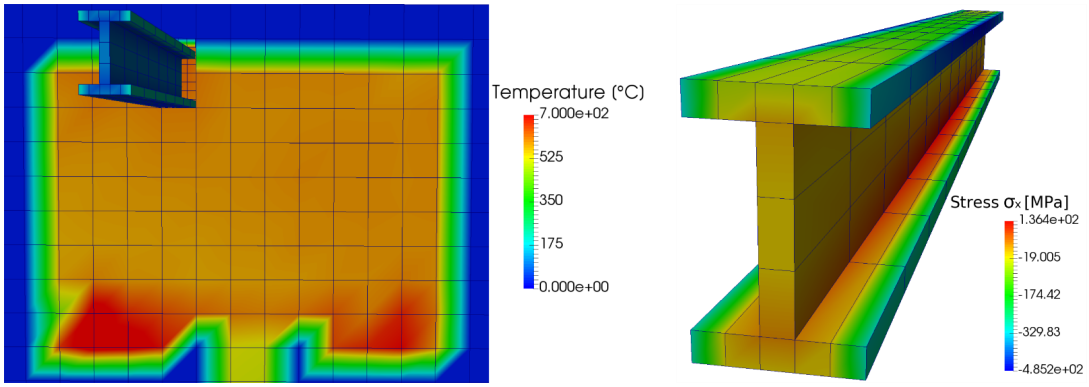

Figure 5: Temperature field at t=576 s with a cross-section in the middle of the furnace (left) and normal stress σ_x (right).

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of EU H2020 Composelector project (GA no: 721105).

REFERENCES

[1] European Materials Modelling Council, www.emmc.info, 2017.

[2] Patzák,B. and Rypl, D. and Kruis, J. *Mupif – a distributed multi-physics integration tool*. Advances in Engineering Software, 60–61(0):89 – 97, 2013.

[3] Patzák, B. and Šmilauer, V. and Horák, M.: *MuPIF object-oriented integration platform*, www.mupif.org, 2018

[4] Composelector: Multi-scale Composite Material Selection Platform with a Seamless Integration of Materials Models and Multidisciplinary Design Framework, EU H2020 project no. 721105, www.composelector.net, 2018

[5] Python Software Foundation. Python Language Reference, version 2.7., [Online]. Available: http://www.python.org.

[6] Pyro - Python Remote Objects, [Online]. Available: http://pythonhosted.org/Pyro.

[7] BeePMN, the collaborative BPMN editor powered by ESTECO, www.beepmn.com.

[8] GRANTA MI: system for enterprise materials information management, Granta, https://www.grantadesign.com/products/mi/system.htm.

[9] Multiscale Modelling Platform: Smart design of nano-enabled products in green technologies, EU FP7 project no. 604279, http://mmp-project.eu

[10] NIST: FDS, Fire Dynamics Simulator, home page. https://pages.nist.gov/fds-smv/, 2017.

[11] Patzák, B. *OOFEM – an object-oriented simulation tool for advanced modeling of materials and structures*. Acta Polytechnica, 52(6):59-66, 2012.