Towards a UML Profile for the Simulation Domain

Maouche Mourad and Bettaz Mohamed
Faculty of Information Technology-Philadelphia University, Jordan
E-mail: mmaouch@philadelphia.edu.jo, mbettaz@philadelphia.edu.jo
http:\\www. philadelphia.edu.jo

Keywords: simulation engineering, simulation platforms, model driven approaches, MDA, UML profile.

Received: May 12, 2017

Model driven approaches have recently been exploited to implement simulation systems. Most of the reported contributions have adopted the Model Driven Architecture (MDA), a model driven approach widely used in software engineering. Simulation Platform Description Models (SPDM), which are first citizens MDA models intended for the description of simulation platforms supporting the execution of simulation experiments, are not explicitly considered in the previous works. The purpose of this work is to define a UML profile intended for the modelling of both simulation core concepts and simulation platforms. The contribution of this work is threefold: First we review and synthesize recent contributions in modelling and simulation approaches, practices and platforms; second we propose a resource-oriented approach for the modelling of simulation platform elements; third we consider both component- and workflow-based simulation platforms.

Povzetek: Predlagana je nova plaltforma za simulacije na osnovi virov.

1 Introduction

Recent research works recommended the practice of the Model Driven Engineering (MDE) in the field of the simulation [2, 3, 4]. The works in [5, 6, 7, 11] adopted approaches based on three steps: A conceptual modelling step where scientists or engineers build models, called Computational Independent Models (CIM) capturing the phenomena under study, a design step where simulation engineers build models called Platform Independent Simulation Model (PISM), and an implementation step where software engineers develop models called Platform Specific Simulation Model (PSSM). A series of model transformations allow to derive PISM models from CIM models, and PSSM models from PISM models. Furthermore the Model Driven Architecture (MDA), a variant of the MDE approach standardized by the Object Management Group (OMG) and targeting the software engineering community, emphasizes also another kind of model called Platform Description Model (PDM) [1]. This kind of model is used for the description of platforms that host the developed software applications.

The MDA approach has been adopted, for instance, in the field of the real-time and embedded systems. UML profiles, like UML-MARTE [8], or Domain Specific Languages like AADL [27] have been defined to support the MDA practice in this field. Both provide mechanisms to describe PDM models, and, seem to be good candidates for the modelling of (software) simulation platforms. In a previous work [22], we proposed a software engineering methodology for the development of multiscale modelling and simulation framework based on the UML-MARTE profile where an attempt to model the multiscale platform MUSCLE using the ingredients of the SRM (Software Resource Model) sub-profile of UML-MARTE have been conducted. The SRM sub-profile is dedicated for the modelling of real time operating systems and middleware. Although it offers a wide range of (software) modelling elements and capabilities, most of them target the specific needs of the real time and embedded systems platforms, and with regard to our previous attempt [22], do not meet specific simulation engineering needs. According to our opinion, it is more natural and comfortable for the simulation engineering community to treat and manipulate their specific native entities and concepts as first class modelling elements.

To the best of our knowledge, none of the current works on the MDE practices in the simulation field, addresses the issue of the Simulation Platform Description Model (SPDM), i.e., the description of simulation platforms that support the execution of simulation experiments. The sole work targeting the modelling of simulation platforms is reported in [9]. Discovering the commonalities and variations among a sample of open source multi-physics simulation platforms has been the main motivation of its authors. Although the work in [9] may serve as a reference architecture for simulation platforms developers, it does not offer, in our opinion, explicit mechanisms to develop models describing simulation platforms in the spirit of the MDA approach.

The objective of this work is to define a UML profile for the simulation field intended to support the MDA practices in this field. The proposed profile particularly provides a set of appropriate modelling mechanisms for the description of simulation platforms.

The contribution of this work is threefold: First we review and synthesize recent contributions in modelling and simulation approaches, practices and platforms; second we adopt a resource-oriented approach for the modelling of simulation platform elements; third we consider both component- and workflow-based simulation platforms. These contributions are illustrated by a set of UML stereotype classes capturing core simulation concepts and platforms elements.

The rest of this paper is organized as follows. Section 2 is devoted to the recent developments in modelling and simulation field. Section 3 presents the simulation field from the workflow perspective. Related works are discussed in Section 4. Our contribution is detailed in Section 5. Section 6 outlines a simple example. Finally conclusions and future works are given in Section 7.

2 Recent developments in simulation engineering

Simulation engineering, an emerging discipline that applies the principles of both simulation science and engineering fields, has been widely used to address various complex real-world problems. It mainly involves two complementary activities: 1) a modelling activity where simulation models of physics phenomena or engineering artefacts- are built, 2) a simulation activity where experiments are performed on these simulation models to achieve specific objectives such as understanding of phenomena, predictions, and performance study. The simulation engineering community developed a lot of specific software tools allowing not only to build such models but also to conduct experiments on them. The literature reports various terminology to designate such tools, like simulation frameworks or simulation platforms; simulation platform is the designation that will be used along this paper to designate such simulation tools. A multitude of academic and commercial simulation platforms are available [10]: Some of them are domain dependent while others are generic. MUSCLE [6] and Mapper [12] simulation frameworks proposed generic simulation platforms. Domains where simulation is widely used are numerous: Physics, biology, medicine, and others. Integrated Plasma Simulator (IPS) platform [13], and Virtual Imaging Platform (VPM) [25] are respectively simulation platforms dedicated to the plasma physics and medical imaging domains.

Due to the profusion of concepts, methods, frameworks and tools related to the modelling and simulation field, we present in the following a synthesis addressing advanced issues relevant to this field.

2.1 Modelling and simulation core concepts

A model is an abstract representation of reality. One of the practical uses of models is generating the dynamic of systems from their models. Simulation consists in moving a model over time, given some inputs. Models can be either in a mathematical form, i.e., a system of equations for example, or in an algorithmic form: In the first case the simulation takes the form of a kind of software, named simulator, that implements a solver for this system of equations; in this case models, often specified thanks to domain specific modelling languages, and simulators are separated. Solvers may be categorized according to different criteria such as their application domain and their solving methods. They may be either legacy code or newly developed codes. In the second case, models are specified in terms of algorithmic components; models are embedded in the simulation code. In our work we deal with both cases.

Simulation codes accept well defined scripts as inputs. These scripts specify the set-up and the protocol of the targeted experiments. Simulation engines interpret the input scripts and run the simulation of individual models. Simulation scripts are usually written thanks to specific scripting languages like Python, and Ruby, or in the form of standardized data representation languages like XML.

2.2 Modelling and simulation approaches

Modern modelling and simulation approaches distinguish between the monolithic approach and the partitioned one. In the first approach a single large scale model capturing the whole phenomena under study is built and then its associated simulation code is executed, while in the second one, a complex model is partitioned into a set of single models and then their associated individual simulation codes are coupled and then executed together.

2.2.1 Partitioned methods

A categorization of partitioned methods is given in[26]:

(i) Multiphysics Partitioning
This method is used when the model of the phenomena under study captures multiple physical processes, each of these physical processes belongs to a specific physics such as temperature and viscosity. In this case the model is decomposed into a set of sub-models; each of these sub-models concerns a specific physical process, and all sub-models of the model operate on the same time and space scales.

(ii) Multiscale Partitioning
This method is used when the model of the phenomena under study captures only one physical process; this model, because of its complexity, is decomposed into a set of sub-models that operate on different time and space scales.

(iii) Multiphysics Multiscale Partitioning
Here multi-scale and multi-physics methods are both used. This method is used when the model of the phenomena captures multiple physical processes that don’t operate on the same scales.

Partitioned simulations encompass not only the performance of a set of single simulation experiments but also the interactions between these single simulation experiments. It presupposes the availability of specific mechanisms, called coupling mechanisms, having the mission to drive these interactions. Two issues need to be addressed when coupling single experiments:

(i) The format of the data exchanged between coupled simulation experiments,
(ii) The interaction pattern governing the interaction between coupled simulation patterns.

The same approach, based on a usual programming technique called *wrapping*, is generally used on almost all simulation platforms that deal with the experiments coupling issues. The wrappers are pieces of code that embodies the simulation code of single experiments. For instance the layered architecture of the Integrated Plasma simulation platform described in [13] distinguishes between data wrappers and coupling wrappers:

The data wrapper takes in charge the data conversion from the internal data format used by single experiments into a common exchange data format. The European Fusion research community suggested a generic data structure, named Consistent Physical Objects (CPO), as a common format for the data to be exchanged between single experiments. Data wrappers are not simulation platforms dependent.

The coupling wrapper takes in charge the data motion as well as the pattern of the interaction between coupled single experiments during their data exchanges. Coupling wrappers, contrarily to data wrappers, are simulation platform dependents.

2.2.2 Coupling issue

In [14] the authors laid the foundations of multi-scale computing. Their formalization of the multi-scale coupling reveals two complementary features related to this concept:

(i) Coupling template: Specifying the information flow that may occur between any pair of coupled (single) experiments. Unidirectional as well as a bidirectional data flows are admitted.
(ii) Coupling topology: A graph representing the couplings (edges) between pair of single sub-models (nodes) belonging to a partitioned model. The graph edges are labelled by coupling templates. Two kinds of coupling topology are identified:
    a. Acyclic topology: It is characterized by an absence of cycles in the coupling topology. In this case coupled simulation codes can be ordered and executed sequentially; this kind of coupling is also named loose coupling.
    b. Cyclic topology: It is characterized by the presence of cycles in the coupling topology. In this case the order of the execution of individual simulation codes is not predefined; this kind of coupling is also called tight coupling.

Figure 1.a and Figure 1.b [14] depict respectively the loose and the tight coupling of three sub-models belonging to a partitioned model. The arrows show the direction of their interactions.

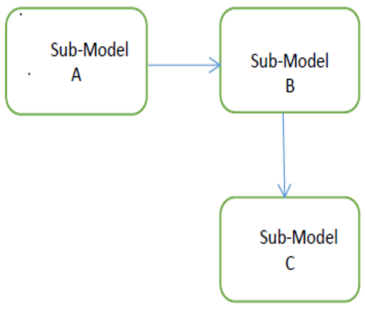

Figure 1.a: Loose Coupling of Sub-Models.

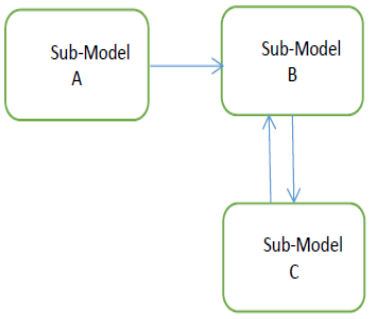

Figure 1.b: Tight Coupling of Sub-Models.

2.3 Orchestration of coupled simulations

Three ways to coordinate and orchestrate a set of coupled single experiments are commonly used:

(i) Centralized mode: A dedicated engine orchestrates and coordinates the enacting of single experiments according to predetermined patterns.
(ii) Master/Slave mode: One of the single experiments plays the role of a master. First, the master experiment is enacted and then the master experiment orchestrates the enactment of the other single experiments, called slave experiments, in a sequential way.
(iii) Component based mode: The coordination is distributed over all the participating single experiments.

3 Simulation from the scientific workflow perspective

The workflow technology, mainly used by the business community, seems to be one of the promising approaches adopted by the scientific community; the concept of scientific workflows emerged as an alternative to the conventional concept of business workflow. There are similarities as well as differences between the two kinds of workflows. For example, business workflows are control-flow oriented, while scientific workflow are mainly data-flow oriented. The readers interested in more details may refer to [15].

3.1 Scientific workflows

A workflow is a pre-defined set of work steps with a partial order on these steps [17]. Work steps represent tasks to be carried out when they are enacted by workflow engines.

Scientific workflows Management Systems have been developed during the last two decades. They are intended to manage, enact and monitor scientific workflows which are a composition of a series of computation and/or data manipulation [13]. Scientific workflows are enacted and orchestrated by specific engines, called workflow engines, forming the core components of scientific workflows Management Systems. Some examples of known scientific workflows management systems are Taverna, Kepler, and Vistrails [16].

Generally, workflows describe control flows and/or data flows. Scientific workflows are usually classified into two categories: Abstract and concrete workflows [19]. Quoting the authors of [18]: “An abstract scientific workflow is a definition of a scientific process with emphasis on the analytical operations or function to be performed rather than the mechanisms for performing these operations”. In opposite, concrete scientific workflows bind the work steps to resources that execute the corresponding tasks.

3.2 Simulation workflows

Simulations of scientific or engineering models are seen as kinds of scientific workflows. Simulations of models are often described by scientific workflows. These workflows follow specific patterns/motifs and include various kinds of steps: Data processing steps, solving/simulation step, visualization step, and data exchanges step. In [24] the authors elaborated catalogues of common motifs for both scientific workflows and data operations that may be performed when conducting scientific experiments.

The iterative pattern is one of the most used control patterns to describe the workflow of individual experiments. For instance structured loops are a kind of iterative pattern.

In the case of a multi-experiment the workflows of the participating individual experiments are coupled. Their coupling is performed thanks to a set of data exchanges constrained by specific interaction patterns. The authors of [20] suggest the concept of “choreography”, borrowed to the business management community, to couple the workflows of single experiments. Every single experiment is realized as an orchestration of scientific services and the whole multi-experiment is described by choreographies without a centralized control.

4 Related works

The literature reports two different directions regarding the development of simulation frameworks:

(i) Component based approaches inspired from the software component-based design and programming methods,
(ii) Workflow based approaches inspired from the workflow based business systems. Recent works with respect to each of these two research directions emphasize the MDA practices.

In [7] the authors proposed a simulation framework based on the hierarchical component-based approach. Their framework is supported by well-defined meta-models capturing Conceptual Simulation Models (CSM) as well as Platform Independent Simulation Models (PISM). However they did not define meta-models that capture Platform Specific Simulation Models (PSSM); in fact these are considered as implementations of PISM models. PISM and PSSM terminology used in the simulation field corresponds respectively to the PIM and PSM terminology used in the software engineering field. It is worthwhile to note that the work in [7] does not consider the simulation platform description models as primary models.

The authors in [21] adopted a workflow based approach for the simulation framework they developed. Their approach, based on an MDA approach too, relies on three distinct levels: A conceptual level at which the modellers describe the models that capture the phenomena under study; an abstract level at which PSSM models, independent form the computing infrastructures are conceived; a concrete level at which models are strongly dependent from the computing infrastructure intended to host the simulation experiments; these last models, called Platform Description Models (PDM) refer to the hardware infrastructure rather than to the simulation workflow framework. Conceptual models are first transformed into specific intermediate representations which are themselves converted to abstract workflows to be enacted by a targeted scientific workflow framework.

Both research works does not consider the modelling of simulation platforms. To the best of our knowledge, the sole research work that investigated the issue of simulation platform modelling is described in [9]. Its authors aimed at discovering commonalities and variations among a sample of open source multi-physics simulation platforms, and proposing a feature model capturing the discovered commonalities and variations using the feature-oriented modelling approach. According to the authors, one of the possible uses of their produced feature model is to serve as a reference for simulation platforms developers.

Our research work, contrarily to [9], targets the modelling of simulation platforms in the context of the MDA approach for the simulation domain, i.e., providing a UML profile intended to build Simulation Platform Description Models (SPDM) for simulation experiments; in opposite to [21], PDM models here refer to simulation platform models rather than to computing infrastructure models.

The present work considers scientific workflows for the description of scientific experiment behaviors, and relies on the concept of generic resources as defined in [8] to model elements of simulation platforms.

5 The proposed UML profile

In this section we develop our UML profile intended for the simulation field. A set of UML stereotypes intended to capture core concepts of the simulation domain are exposed.

5.1 Linking PISM and SPDM models

The proposed profile focuses on the SPDM modelling. Figure 2 depicts the well-known relationship between the PISM, and PSSM models. Elements of PISM models are mapped to elements of SPDM leading to PSSM models.

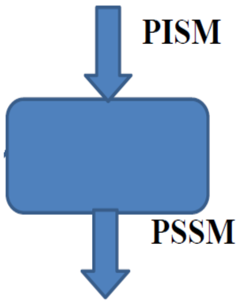

Figure 2: Linking PISM and PSSM.

Our approach relies on two first class UML model elements to describe simulations:
-*Experiment*: intended to describe the simulation of either a monolithic model or the simulation of a single model (member of a partitioned model).
-*Simulation*: intended to describe the architecture of the simulation of a whole model (either monolithic or partitioned model) according to a desired simulation approach (monolithic/partitioned) and design (component-based/ workflow-based).

5.2 PISM model elements

In this section we identify and define a set of UML stereotypes that constitutes the main PISM model elements of our profile.

5.2.1 Simulation stereotype

The simulation and experiment concepts, as defined above, are modelled as stereotypes. Both extend the UML *BehavioredClassifier* metaclass which is a UML classifier that owns behaviors.
A. The class diagram depicted in Figure 3.a describes the *Simulation* stereotype and the hierarchy of its refined stereotypes covering various kinds of simulation approaches.
Comments:
(i). Simulation Stereotype includes at least two properties.
IdentifierElts reports a set of required elements that may identify and characterize conducted simulations such their identification number, their date, the target domain, the version number.
SimulParam is used to report some parameters related to the simulation itself; for instance the duration of the simulation, the space dimension of the simulated model and others.
(ii). PartitionedSimulation and *MonolithicSimulation* are refinements of the *Simulation* Stereotype. *Expnumber* property defined in *PartitionedSimulation* records the number of single experiments participating to a partitioned simulation instance.
(iii).MultiscaleSimulation is a refinement of the *PartitionedSimulation stereotype.* Its *scales* property records the kinds of scale dimension (time, space, time and space) characterizes a simulation instance.
Dimension is an enumeration type intended to carry various kinds of scales.
Dimension == time | space | time&space |…..

A.The class diagram shown in Figure 3.b presents a hierarchy of various multiscale simulation design approaches according to the way coordination and orchestration of coupled single experiments are done.
Comments:
(i). *CompBasedMsc* stereotype represents multiscale simulations designed according to the component based approach. *Conf* property records the configuration of multiscale simulations i.e., its topology (refer to section 2.2.2). We introduce the stereotype *Coupling* as an extension of the UML Association metaclass to model the simulation configuration. The details of this stereotype are shown in Figure 3.d.

(ii). *WrkFlowBasedMsc* stereotype represents multiscale simulations designed according to the workflow based approach. It is mainly characterized by two properties. *Wbeh* property specifies the abstract workflow associated with the workflow based multiscale simulation. The *WorkFlowBeh* stereotype is defined in the part A of section 5.2.3

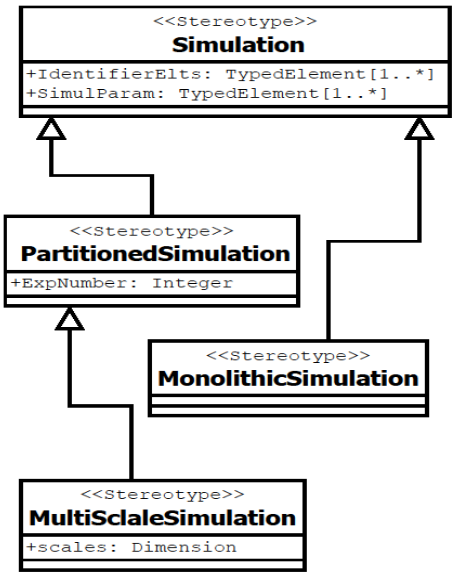

Figure 3.a: Simulation Stereotype and its Refined Stereotypes. Remark: UML TypedElement refers to a pair (named element, its associated type). TypedElement [0..*] means zero or more.

Map property specifies the mapping between workflow nodes and their corresponding workflow *call actions*. The *Mapping* class is a datatype that records (workflow node, action to be called) pairs. The concept of *UML call action* is detailed in the part B of section 5.2.3.

(iii). *CentralizedMsc* stereotype represents multiscale simulations designed according to the centralized version of the workflow based approach. It refines *WrkFlowMsc* stereotype. Its *coord* property (instance of the Coordinator class) is intended to represent the central coordinator that orchestrates the whole simulation workflow. The *Coordinator* class is not detailed in this paper.

(iv). *MasterSlaveMsc* stereotype represents multiscale simulations designed according to the master/slave version of the workflow based approach. It refines the *WrfFlowMsc* stereotype. Its *Master* property records the single experiment that plays the role of master in the whole multiscale simulation.

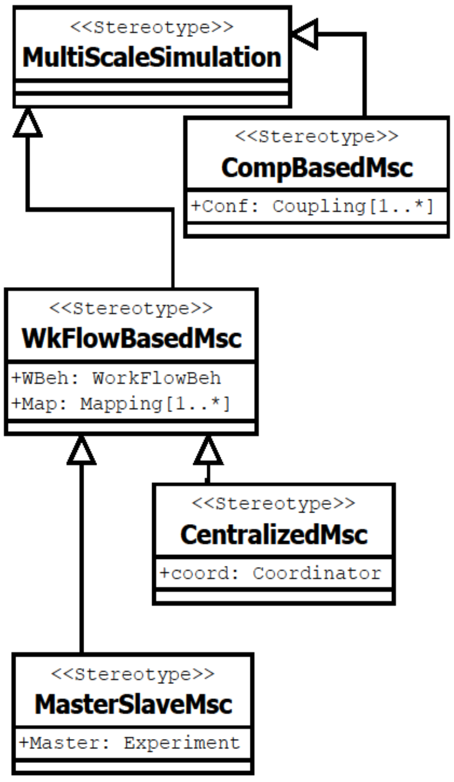

Figure 3.b: Hierarchy of MultiScale Design Approaches.

Figure 3.c shows the relationship between the stereotypes *Monolithic/Partitioned* and *Experiment* stereotypes (more details on the *Experiment* stereotype are given in the section 5.2.3)

Monolithic simulations include only one single experiment whilst partitioned simulations include more than one single experiment.

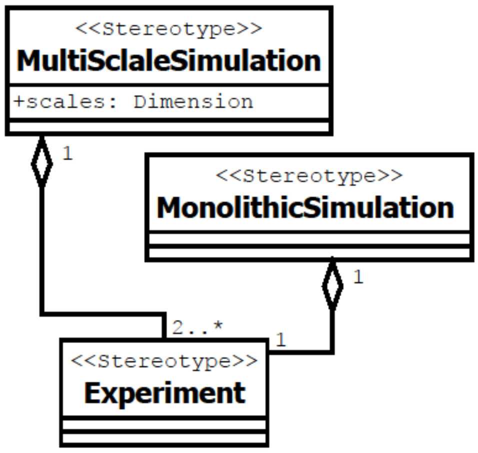

Figure 3.c Linking Simulation with Experiments.

5.2.2 *Coupling* stereotype:

Various kinds of couplings are identified:

- Direct coupling between pairs of experiments participating to component based multiscale simulations. This kind of coupling may various forms. For instance the designers of the MUSCLE multiscale platform use the term “coupling template” to refer to these coupling forms.
- Indirect coupling between slave experiments through a master experiment in case of master-slave multiscale simulations.
- Indirect coupling between experiments through a coordinator in case of centralized multiscale simulations.

Figure 3.d shows the specification of the proposed *Coupling* stereotype. This stereotype extends the UML association metaclass and it is characterized by the following properties:

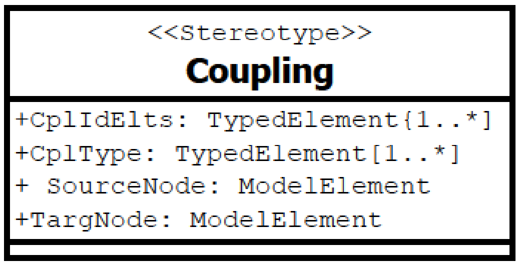
<<Stereotype>>
Coupling
+CplIdElts: TypedElement{1..*]
+CplType: TypedElement[1..*]
+ SourceNode: ModelElement
+TargNode: ModelElement

Figure 3.d: *Coupling* Stereotype.

+ *CplIdElts*: Specifies suitable information susceptible to identify its instances.
+ *CplIType:* Set of suitable typed elements allowing to specify the kind of the coupling.
+ *SourceNode*, *TargetNode*: These attributes play the role of the UML association end. They specify the model elements that are coupled.

5.2.3 Simulation behavior stereotype

Instances of both *Experiment* and *Simulation* stereotypes own their specific behaviours. The stereotype

SimBehavior is intended to capture various simulation and experiment behaviors.

A. Figure 4.a shows a class diagram depicting the usual behaviors met in the simulation world. *The SimBeh* stereotype is intended to model the behavior of experiments and simulations. Two categories of behavior are identified. The opaques ones characterized by their unknown structure, and the regular ones characterized by well-defined, regular and known structures. For instance, workflows and automata-like structures are kinds of regular behavior.

Comments:

(i) Opaque behaviors, as defined in the UML infrastructure, are usually characterized by their body (body source plus the language used to express the source); in the context of our work, *Opaque Experiment* stereotype represents experiments driven by simulation engines. Here we adopt the UML Opaque Behavior metaclass as a base class.

(ii) Automata-based behavior which are explicitly described by automata-like formalisms such as Cellular Automata or others. Such kind of behaviors may, for instance, characterizes the behavior of single experiments that participate to multiscale simulations. Here we adopt the UML State Machine metaclass as a base class.

(iii) Workflow-based behaviors which are explicitly described by abstract workflows. Such kind of behavior may for instance characterizes the behaviour of monolithic simulation as well as multiscale simulations. These are often expressed in terms of Petri nets or UML activity diagrams. The authors of [23] defined a profile for scientific workflows by proposing a refinement of the UML Activity metaclass tailored to their own abstract workflow language. In our work we define the *WorkFlowBeh* stereotype to represent abstract simulation workflows by extending the UML Activity metaclass. *SimMotif* is one of the properties associated with the *WorkFlowBeh* stereotype. It is intended to specify the abstract motif/pattern of simulation workflows. Abstract simulation workflows are composed by sets of workflow nodes assembled according to a particular structure. We assume the availability of a library of UML model elements regrouping a catalogue of usual simulation workflow motifs.

B. More on Workflow based Experiments

Workflow-based experiments are usually composed of work steps structured and organized according to a specific workflow motif/pattern. In order to be independent from specific abstract workflow language, we adopt a solution, used by some workflow engines, that uncouples the workflow motif nodes from the task to be performed at the node level. To achieve this objective, we rely on the UML Behavior metaclass infrastructure to define the *SimulationWorkflowStep* stereotype.

This stereotype extends the UML *Call Operation* and *Call Behavior* metaclasses which are themselves two refinements of the UML *Execution Action* metaclass:

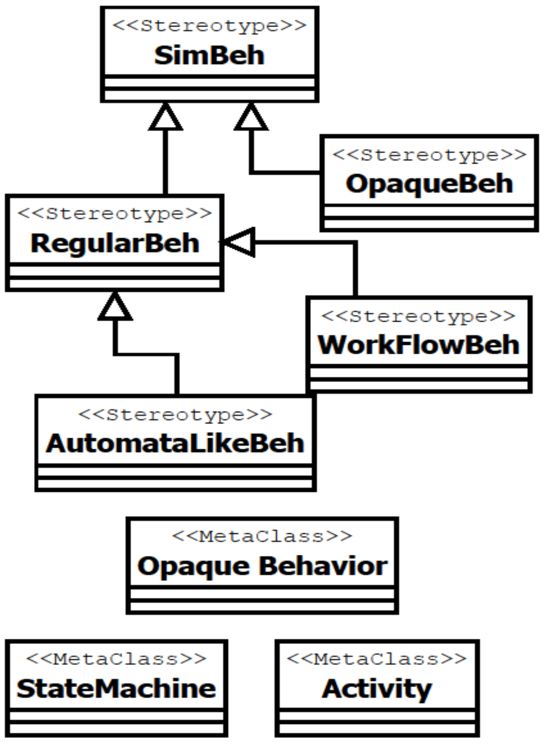

Figure 4.a Typology of Simulation Behaviors

- *Call Operation* is used to trigger atomic operations that correspond to simulation processing steps, like solving, data processing or data interaction steps.
- *Call Behavior* is used to trigger behaviors that correspond to potential sub-workflows contained in simulation workflows (hierarchical workflow motifs). It is useful to handle the master/slave approach (a master experiment enacting a slave experiment) and the centralized approach (a coordinator enacting the workflow of single experiments).

Figure4.b shows two refinements of the *SimulationWorkflowStep* stereotype:

SimAction stereotype representing various kinds of atomic simulation actions call (solving, data processing, data interaction operations) that may be associated with nodes of abstract workflow motifs. It extends the UML *Call Operation* metaclass.

WrkFAction stereotype representing sub-workflows with call action that may be associated with nodes of workflow motifs. It extends the UML *Call behaviour* metaclass.

5.2.4 Experiment and simulation model stereotypes

The *Experiment* stereotype represents PSIM elements. Figure 5 shows the features of this stereotype.

(i) *IdentifierElts* property records any useful information susceptible to identify the experiment (identifier number, experiment date, version, and eventually others).

(ii) *ExpParam* property records experiment parameters (experiment duration, and eventually other parameters).

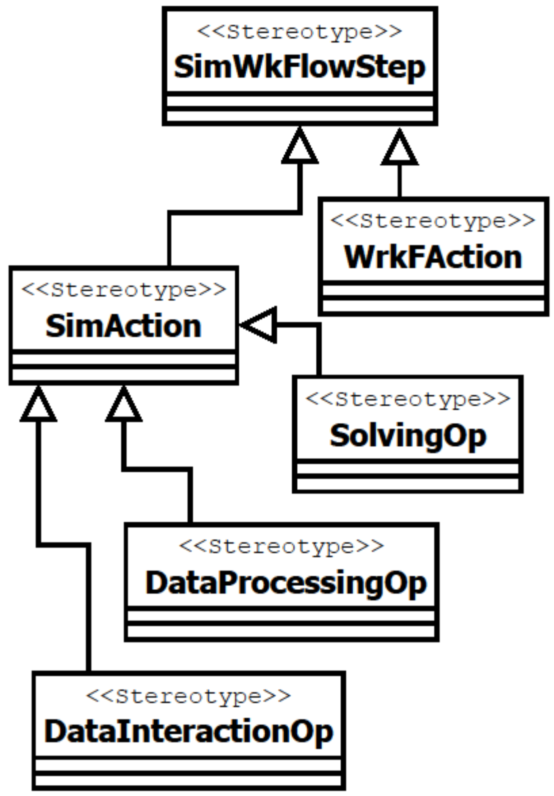

Figure 4.b: Simulation WorkFlow Step Stereotypes Hierarchy**.**

(iii) *ArchElts* property is intended to record any useful information related to the various simulation design approaches. The type of *ArchElts* property type is kept flexible in order to describe various simulation design approaches (monolithic, component based, centralized workflow based, master/slave workflow design approaches). *ModelElement* is a UML-MARTE defined metaclass that refers to any UML classifier.

(iv) *Smod* property specifies the simulation model targeted by the experiment. It may be either a whole simulation model (monolithic simulation) or a single simulation model (partitioned simulation).

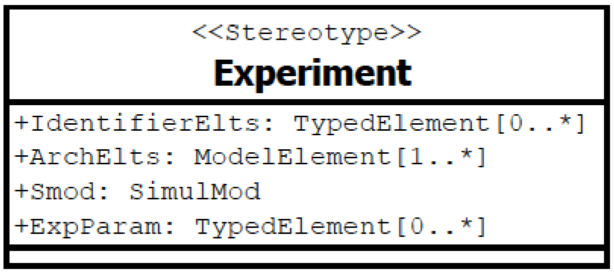
<<Stereotype>>
Experiment
+IdentifierElts: TypedElement[0..*]
+ArchElts: ModelElement[1..*]
+Smod: SimulMod
+ExpParam: TypedElement[0..*]

Figure 5. *Experiment* Stereotype.

Figure 6 shows the features of the *SimulModel* class:

(i) *Field* property specifies the application domain concerned by the simulation (engineering, physics, biology, and others). *Domain* class represents the various domains where simulations may be conducted.

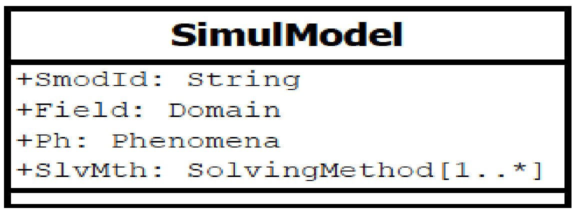

Figure 6. *SimulModel* Class.

(ii) *Ph* property specifies the domain specific phenomena targeted by the simulation.

(iii) *SlvMth* property specifies the set of mathematical methods that may be used to solve the simulation model. We define *SolvingMethod* a stereotype as an extension of the UML *OpaqueExpression* metaclass.

5.3 SPDM model elements

Simulation and experiments, as previously mentioned, are hosted and executed by simulation platforms.

UML-MARTE profile provides the concept of *Resource* to model in a uniform way hardware as well as software elements. Resources are abstract entities that provide services and they are themselves composed of other resources. We refine the concept of abstract resource to concrete (software) elements of simulation platforms.

In the present work we focus on only two core stereotypes that may be used to model PDSMs: Engines and Data Processor resources.

5.3.1 Engine resources

The concept of “engine’ is often used in the simulation field as well as in the workflow technology. Here engines represent virtual computing resources that interpret and run scripts or workflows written in specific formalisms. *Engine* refines the abstract *Resource* stereotype class defined in UML-MARTE profile.

This abstract resource provides a set of services common to all kinds of resources.

Figure 7 shows two kinds of engines: Simulation and Workflow engines.

A. SimulationEngine: An engine that interprets opaque simulation code written in specific formalism/language. It may also be a simulation tool, called simulator, that performs solving methods; simulators accept models and simulation scripts as inputs.

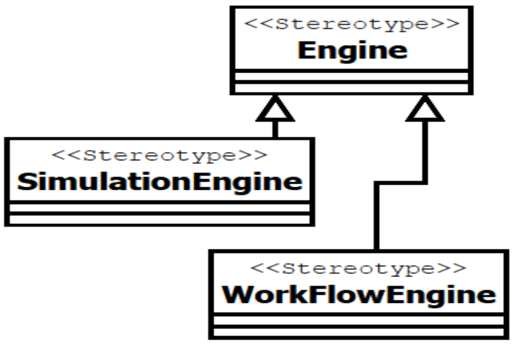

Figure 7. Simulation and Workflow Engines Stereotypes.

(i) I*nterpreter:* Specifies the formalism that is interpreted by the simulation engine,
(ii) K*ind:* Specifies the type of simulation engine. *SimulEngineEnumeration==* simulator| embedded simulation code |….
(iii) *Slv-method*: Specifies the set of numerical method that are supported by the simulation engine.
(iv) *Computation:* Specifies if the engine performs sequential or parallel computations.

Figure8 represents the main features of the *Simulation Engine* stereotype.

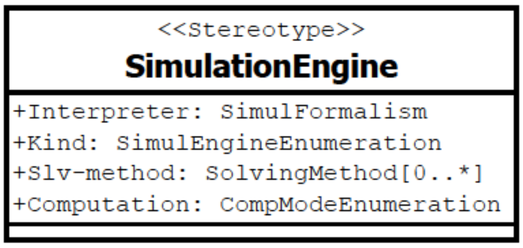
<<Stereotype>>
SimulationEngine
+Interpreter: SimulFormalism
+Kind: SimulEngineEnumeration
+Slv-method: SolvingMethod[0..*]
+Computation: CompModeEnumeration

Figure 8: Simulation Engine Stereotype.

*B. WorkflowEngine***:** An engine that is responsible for the interpretation of executable workflow and the orchestration of workflows. It is a kind of scheduling resource. Workflow steps may be either basic/atomic tasks or sub-workflows. Modellers specify their workflows using either a human readable textual script or a diagram-based workflow language (Front-

End workflow language), while workflow engines interpret platform readable and executable workflow languages (Back-End language).
Figure 9 depicts the main features of the Workflow Engine stereotype.

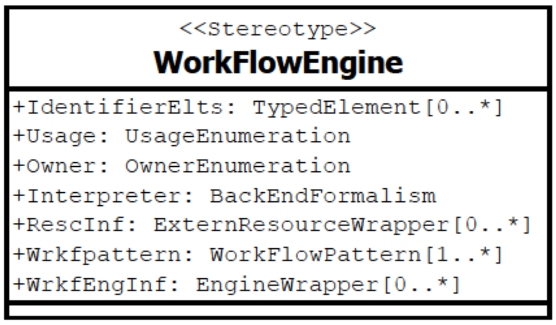
<<Stereotype>>
WorkFlowEngine
+IdentifierElts: TypedElement[0..*]
+Usage: UsageEnumeration
+Owner: OwnerEnumeration
+Interpreter: BackEndFormalism
+RescInf: ExternResourceWrapper[0..*]
+Wrkfpattern: WorkFlowPattern[1..*]
+WrkfEngInf: EngineWrapper[0..*]

Figure 9. *WorkflowEngine* Stereotype.

(i) W*orkFlowPattern is a* sub-class of the *Control Node* meta-class. It includes the usual set of control nodes found in simulation workflows like sequence, loop, and parallel.
(ii) *ExternResourceWrapper*, and *EngineWrapper* are derived from the UML *Adapter* pattern. *External ResourceWrapper* refers to wrappers that encapsulate data processing operators, and *EngineWrapper* refers to wrappers that encapsulate simulation engines in case of cooperation between workflow engines.

5.3.2 Data processor elements

In the following, we present a set of stereotypes aiming to model a set of specific computing resources that are able to support the execution of specific operations: data operation, and data interaction. We model these resources as kinds of virtual processor.

Our approach to categorize the data operations is slightly different from the one reported in [20]. We differentiate the data processing operations that may operate inside individual experiments, the intra-experiment case, from the operations on data that are performed along the data motion from one single experiment to another experiment, the inter-experiment case. A categorization of these Data processors is shown in Figures 10a, 10b, and 10c. The following kinds of data processor are identified:

A. Inter-Experiment Data Processor

Data are potentially subject to manipulation during their motion between single experiments. Each kind of manipulation is described by a specific (mathematical) function or algorithm. Two kinds of manipulations are identified:

(i) Data transformation: filtering,
(ii) Data combination: usually carried out by operators called Mappers.
 (a) Data aggregation: aggregating multiple data sources to one data source,
 (b) Data dis-aggregation: separating one data source into multiple data sources.

B. Intra-Experiment Data Processor

Usually the input data need to be set into a specific format before to be submitted to simulation engines. The output data (produced by simulation engines) need also to be set in specific formats before to be visualized to the modellers. Commercial and academic libraries provide such data processors.

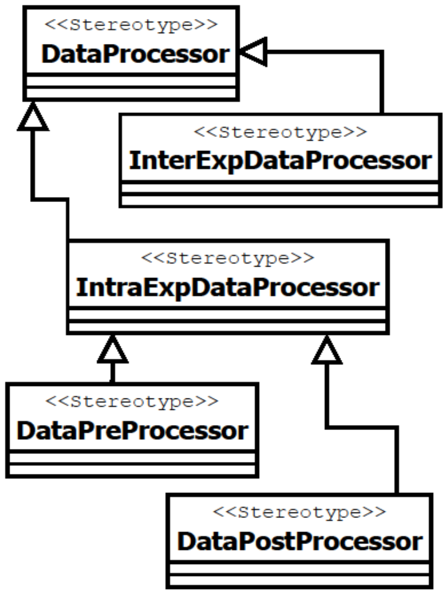

Figure 10 a. Data Processors Classification.

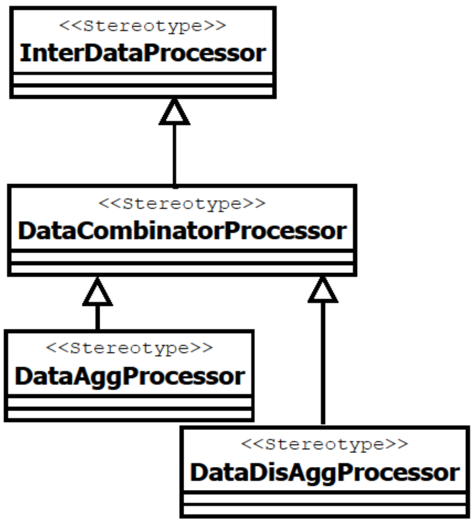

Figure 10 b. Data Combinator Processors Classification.

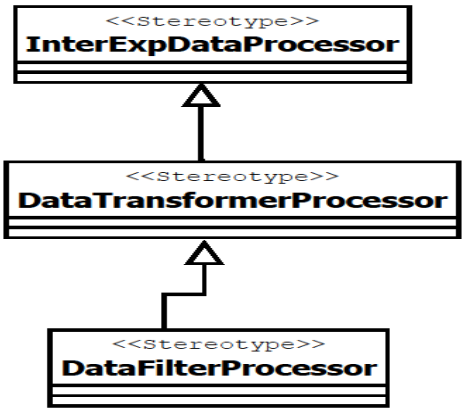

Figure 10.c Data Transformer Processor.

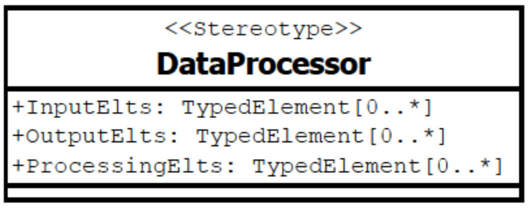
<<Stereotype>>
DataProcessor
+InputElts: TypedElement[0..*]
+OutputElts: TypedElement[0..*]
+ProcessingElts: TypedElement[0..*]

Figure 11: Data Processor Stereotype.

C. The stereotype *Data Processor* inherits from the *Resource* class. Its main features are:

(i) *InputElts*: Specifies the number and types of inputs which depend from the kind of data processor,

(ii) *OutputElts*: Specifies the number and types of outputs depend from the kind of data processor,

(iii) *ProcessingElts*: Specifies an algorithm (body) that implements the analytic (mathematical) operation to be performed as well as a set of appropriate parameters qualifying its performance.

5.3.3 Data interaction operator

Single experiments participating to multiscale simulations are coupled according to specific coupling mechanisms. They exchange data either in a direct way, in case of a component based multiscale simulation approach or in an indirect way in case of master/slave and centralized multiscale simulation approaches.

Our profile provides a stereotype class named *DataInteractionOperator* intended to run various kinds of coupling (data motion according to specific templates). It represents an abstraction of the so-called coupling wrappers mentioned in the section 2.2.1. We adopt and refine the UML Adapter pattern to define this stereotype.

6 Example

In this section we introduce a simple example to illustrate the (partial) use of our proposed profile. The example exposes only the PISM model elements.

The example presents a component based multiscale simulation which consists of two single scale experiments namely C1 and C2 interacting through two couplings namely Cp12 from C1 to C2 and Cp21 from C2 to C1.

C1 is the experiment on the simulation model Mod1 and C2 is the experiment of the simulation Mod2. Both Mod1 and Mod2 are single scale models of the partitioned simulation model Mod.

a. Instantiation of the stereotype *Simulation* with the following tags:

+ *IdentifierElts* =
SimlId: String
SimDt: Date
SimVersion: String
SimDm: Domain /Domain: a data type/

+ *SimulParam* =
SimDuration: Time
SimSpace: Space /Space: a data type/
SimMd: SimulMod / Simulation Model/

b. Instantiation of the stereotype *MultiscaleSimulation* with the following tags:
+ *scales* = time&space /value of Dimension Enumeration type/
+*ExpNumber* = 2 /property of *PartitionedSimulation*

c.Instantiation of the stereotype *CompBasedMsC*
+*Conf* =
Cp12: Coupling
Cp21: Coupling
Two couplings in our example Cp1 and cp2

d. Instantiation of the stereotype *Coupling*
+ *CplType=*
CcplK: CouplingKind */CouplingKind:* enemuration data type*/*
CpT: *CouplingTemplate* /CouplingTemplate is a *data type/*
+ *CpMeth: OpaqueExpression* /coupling code algorithm

+ SourceNode =
Src: InPort /InPort: a UML model element/
+ TargNode =
Targ: OutPort /OutPort: a UML model element

For Cp12 Instance
CplK = directcoupling /direct coupling between two experiments/
CpT = tempX / templateX is one instance of CouplingTemplate/
CpMeth = MethX /MethX: a coupling algorithm for tempX/
Src =out1
Targ= in2

```
    For Cp21 Instance
    CplK =  directcoupling   /direct coupling between two experiments/
     CpT  =    tempY    / templateY is one instance of CouplingTemplate
    CpMeth = methY      /MethY a coupling algorithm for  tempY /
    Src =out2
    Targ= in1
```

e. Instantiation of the stereotype *Experiment*

From the architectural point of view Experiment instances are seen as components owning an internal behavior and characterized by a set of input and output ports for their interaction (coupling) with other experiments. In this example we use the SEL (SubModel Execution Loop) behavior borrowed from the MUSCLE multiscale framework.

```
+ IdentifierElts =
      ExpId: String
      ExpDt: Date
      ExpVersion: Integer

+ArchElts =
 ExpBeh: SEL      / SEL:  Behaviour of specific cellular automata/
 ExpIn: InPort [1..*]                  / InPort: model element/
 ExpOut:OutPort [1..*]               /  OutPort: model element/

     For C1 Experiment instance:
      ExBeh =sel1   \ an instance of the SEL data type\
      ExpIn = {in1}
      ExpOut = {out1}
     For C2 Experiment instance:
     ExBeh =sel2   \ an instance of the SEL data type\
     ExpIn = {in2}
ExpOut = {out2}

+ExpParam =
 ExpTimeScale: Time              /Time scale of the Experiment/
 ExpSpaceScale: Space           /space scale for the experiment

     For C1 Experiment instance:
      ExpTimeScale = t1
      ExpSpaceScale = sp1
      For C1 Experiment instance:
      ExpTimeScale = t2
      ExpSpaceScale = sp2

+SMod =_Mod1     (Mod2 for the C2 Experiment instance).
```

Realistic and complete case studies are currently under construction.

7 Conclusion and future works

In this work we present a synthesis of recent contributions in the modelling and simulation field encompassing up-to-date simulation topics. Model driven approaches for the simulation field are discussed. Multi-scale and multi-physics simulation methods and their related issues are outlined. Modern simulation platforms adopting a component- as well as a workflow-based approach are exposed.

We also propose modelling mechanisms intended for the description of simulation platforms, thus making possible the development of a kind of MDA primary model called SPDM. For this purpose we define a UML profile including a set of useful UML stereotypes that capture core simulation concepts as well as core simulation platforms elements such as simulation engines, workflow engines, and simulation data processors. In this work, a resource-based approach, similar to the one used for the UML-MARTE profile, is adopted for the modelling of simulation platforms elements.

As a first future work we plan also to develop UML meta-models for a set of widely used simulation model specification formalisms, thus enabling PISM-to-PISM transformations.

Acknowledgement

This research work is fully supported by the Deanship of the Scientific Research - Philadelphia University (Jordan).

References

[1] A.-R. Da Silva, Model Driven Engineering: A Survey supportedby the Unified Conceptual Model, *Computer languages, Systems and Structures*, Vol. 43, pp. 139-155, Elsevier, 2015.

[2] S. Wagner, D. Pfluger, and M. Mehl, Simulation Software Engineering: Experiences and Challenges, in*Proc.of the International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering*, pp. 1-4, 2015.

[3] O. Topcu, U. Durak, H. Oguztuzun, and L. Yilmaz, Distributed Simulation: A Model Driven Engineering Approach, *Simulation Foundations, Methods and Applications, Springer*, 2016. https://doi.org/10.1007/978-3-319-03050-0

[4] G. Wagner, Model-Driven Engineering of Second-Life-Style Simulations, in *the Proc. of the Winter simulation conference*, 2010.

[5] A. Yang, and W. Marquard, An Ontological Conceptualization of Multiscale Models, *Computer ChemicalEngineering*, Vol. 33, pp. 822-837, (2009). https://doi.org/10.1016/j.compchemeng.2008.11.015

[6] J. Borgdorff, M. Mamonski, B. Bosak, K. Kurowski, M. Ben Belgacem, B, Chopard, D. Groen, P.-V. Covery, and A.-G. Hoekstra, Distributed Multiscale Computing with MUSCLE2, the Multiscale Coupling Library and Environment, *Journal of Computational Science*, Vol. 5, Issue. 5, pp. 719-731, Elsevier, 2014.

[7] D. Cetinkaya, A. Verbraeck, A., and D.-M Seck, Applying a Model Driven Approach to Component Based Modeling and Simulation, in *Proc. of the Winter Simulation conference*, 2010.

[8] OMG, A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, Beta 2, *OMG Document Number: ptc/2008-06-09,* 2008.

[9] O. Babur, V. Smilauer, T. Verhoeff, and M.V-D. Brand, A survey of Open Source Multiphysics Frameworks in Engineering", *Procedisa Computer Science*, Vol. 151, pp. 1088-1097, Elsevier, 2015.

[10] O. Babur, T. Verhoeff, and M.G.-J. Van Den Brand, Multiphisics and Multiscale Sofware Frameworks: An Annotated Bibliography, Computer Science Reports, Technische University Eindhoven, Vol. 1501, 2015.

[11] Y. Zaho, C. Jiang, and A. Yang, Towards Computer-Aided Multiscale Modeling: An Overarching Methodolgy and Support of Conceptual Modeling, *Computer and Chemical Engineering,* No.36, pp. 10-21, Elsevier, 2012.

[12] M. Ben Belgacem, and al, Distributed Multiscale Computations Using the MAPPER Framework", *Procedia Computer Science*, Vol. 13, pp. 1106-1115, Elsevier, 2013.

[13] O. Hoenen, D. Coster, S. Petruczynik, and M. Plociennick,Coupled Simulations in Plasma Physics with the Integrated Plasma Simulator Platform, *Procedia Computer Science*, Vol. 5, pp. 1138-1147, Elsevier, 2015.

[14] J. Borgdorff, J.-L. Falcone, E. Lorenz, C. Bona-Casas, B. Chopard, and A.-G. Hoekstr, Foundations of Distributed Multiscale Computing: Formalization, Specification and Analysis, *Journal of Distributed Computing*, Elsevier, Vol.73, pp. 465-483. 2013. https://doi.org/10.1016/j.jpdc.2012.12.011

[15] U. Yildez, A. Guabtni, and A.H-H. Ngu, Business versus Scientific workflow: A Comparative Study, Research Report No. 2009-3, Project DAKS, Department of Computer Science, UC Davis University of California, 2009.

[16] T. Buchert, L. Nusbaum, and J. Gustedt, A Workflow-Inspired, Modular and Robust Approach to Experiments in Distributed Systems, Project-Team Algorille, Research Report n0 8404, Research Center Nancy-Grand Est, 2013.

[17] C.-A. Ellis, Workflow Technology, Chapter No. 2 in Computer Supported Cooperative Work, Edited by Beaudouin-Lafon, John Wiley and Sons Ltd, 1999.

[18] B. Dashtban, Scientific Workflow Patterns, Msc Dissertation, School of Advanced Computer Science, Manchester University, 2012.

[19] X.-R. Xiang, andG. Madey, Improving the reuse of scientific workflows and their by-products, in *Proc.IEEE International Conference on Web Services*, pp. 792-799, 2007.

[20] A. WeiB, andD. Karastoyanova, A Life Cycle for Coupled Multi-Scale, Multi-field Experiments Realized through Choreographies, in *Proc. Enterprise Distributed Object Computing Conference (EDOC)*, 2014.

[21] N. Cerezo, J. Montagnat, and M. Baly-Fornarino, Computer-Assisted Scientific Workflow Design, *Journal of Grid Computing*, Vol. 11, No. 3, pp. 585-610, Springer Verlag, 2013.

[22] M. Maouche, M. Bettaz, Towards a Software Engineering Approach to Multi-Scale Modelling and Simulation, *IJSEIA Journal*, Vol. 10, 2016.

[23] J. Qin, T. Fahringer, andS. Pllana, UML Based Grid Workflow Modeling Under ASKALON, Chapter in Distributed and Parallel Systems Book, pp. 191-200, Springer, 2007.

[24] Garijo, D., Alper, P., Belhajjame, K., Corcho, O., Gi, Y., Common Motifs in Scientific Workflows: An Empirical Analysis, Future Generation Computer Systems, Elsevier, (2013).

[25] R. Ferreira Da Silva, R., S. Camarasu-Pop, B. Grenier, V. Hamar, D. Manset, J. Montagnat, J. Revillard, J,-R. Balderrama, A. Tsaregorodtsev, and T. Glatad, Multi-Infrastructure Workflow Execution for Medical Simulation in the Virtual Imaging Platform, in *Proc. HealthGrid Conference*, pp.1-10, 2011.

[26] D. Groen, S.-J. Zasade, and P.-V. Coveney, Survey of Multiscale and Multiphysics Applications and Communities, *Computing in Science & Engineering*, Vol. 16, Issue. 22, pp. 34-43, 2014. https://doi.org/10.1109/MCSE.2013.47

[27] S. Turki, E. Senn, and D. Blouin, Mapping the MARTE UML profile to AADL, in MoDELS 2010 ACES-MB Workshop Proceedings, pp. 11-20, 2010.