Coupling of Existing Simulations using Bottom-up Modeling of Choreographies

Andreas Weiß, Dimka Karastoyanova
Institute of Architecture of
Application Systems (IAAS)
University of Stuttgart
{andreas.weiss, dimka.karastoyanova}
@iaas.uni-stuttgart.de

David Molnar, Siegfried Schmauder
Institute for Materials Testing,
Materials Science and
Strength of Materials (IMWF)
University of Stuttgart
{david.molnar, siegfried.schmauder}
@imwf.uni-stuttgart.de

Abstract: As a contribution for eScience, we discuss the bottom-up derivation of scientific choreography models from existing simulation workflows interconnected as a multi-scale and multi-field simulation. Starting from a motivating scenario of only implicitly coupled simulation workflows for the studying of thermal aging of iron-copper alloys, we present a choreography life cycle supporting the bottom-up derivation of choreography models and the propagation of changes to the underlying simulation workflows in a round-trip manner. Furthermore, we discuss several distinct starting points for the derivation, namely explicitly and implicitly connected simulation workflow models and already running simulation workflow instances.

1 Introduction

In eScience, the main objective is to provide generic approaches and tools to support the whole eScience life cycle [HTT09] and different fields of natural and social sciences for the purpose of faster scientific exploration and discovery. In previous work in the scope of the Cluster of Excellence Simulation Techology (SimTech[1]), the Model-as-you-go approach was introduced [SK10], [SK13]. This approach is based on the workflow paradigm, i.e., simulation steps are modeled as workflow activities and implemented by existing software systems exposing their functionality as services [SHK+11]. Incomplete workflow models can be extended after they have already started in order to conduct the iterative, trial-and-error based modeling of eScience experiments [BG07]. Furthermore, workflow logic can be re-executed, i.e., already executed steps can be compensated and executed again with a different set of parameters as well as re-iterated for convergence of results. However, the existing Model-as-you-go approach and the corresponding implementing scientific workflow management system (sWfMS) only support single scale and single field simulations. Multi-scale and multi-field (so-called multi-*) simulations can only be supported if the orchestrated simulation software is already coupling the different scales/fields on the level of the mathematical formalization. Typically, descriptions of one or more scales/fields to another scale/field are used. Multi-scale simulations cover different scales within the same

[1] SimTech: `http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/`

computer experiment, where the scales can either refer to time scales, e.g., nanoseconds to days, or to length scales, e.g., nanometers to meters. Multi-field simulations use different scientific fields in the same experiment, e.g., physics, biology, or chemistry. An example is the nucleation, growth and coarsening of second phase particles [MMC+12], also referred to as precipitation. With this approach, multiple time scales of thermal aging as well as length scales in terms of sample volumes become accessible by coupling two simulation methods, the kinetic Monte Carlo (KMC) and the Phase-field Method (PFM), each describing the phenomena of precipitation from a different point of view.

We have observed two basic scenarios that need to be supported: (i) a bottom-up approach where existing software implements different mathematical models and/or scales that need to be orchestrated, very often across organizations. (ii) a top-down approach where one organization needs to realize a particular multi-scale/field simulation and starts modeling and implementation from scratch. In both scenarios the major open issue with respect to modeling is how the interactions among the participating simulations and the data exchange can be represented. In this paper, we focus on the bottom-up aspect of modeling and executing already existing simulation workflows and software. Parts of a multi-* simulation are modeled as workflows, but intermediate results may have to be copied between the simulation software components and subsequent simulation methods may have to be triggered manually. Modeling support of multi-* simulations and automation as a consequence of explicitly modeled control and data flow between simulation software components would decrease simulation duration and increase user friendliness. Furthermore, flexibility such as Model-as-you-go support for multi-* simulations is also needed to cope with the iterative, trial-and-error modeling approach of scientists.

Towards this goal, we contribute a bottom-up view on a life cycle realizing modeling and execution of simulations (Sec. 3) and discuss different cases that occur when deriving an overall model of existing and interacting simulations (Sec. 4). We focus on using proven methods and technologies of the business domain [SK10]. In this work, we make use of *choreographies* to describe the global model of a multi-scale and multi-field simulation. Choreographies are a concept known from the business domain that enables independent organizations to collaborate and reach a common business goal. Choreographies provide a global view on the interconnection of independent organizations communicating without a central coordinator [DKB08]. While choreographies show the public interfaces of the collaboration, these interfaces are implemented by orchestrations, i.e., the so-called enacting workflows, realizing the private business logic of a single organization. The distinct organizations (and their workflows) are called *choreography participants*. To complete this paper, Sec. 5 shows the current state of our implementation supporting the bottom-up life cycle and choreography derivation. We compare our approach with related ones in Sec. 6 and conclude the article with an outline of future research topics in Sec. 7.

2 Motivation

In this section, we introduce a motivating scenario from the domain of material science. Molnar et al. have studied the thermal aging of iron-copper alloys [MMC+12] and emerging effects of existing precipitates on the mechanical behavior [MBHS12], [MBM+14]

of the single crystalline structures by coupling kinetic Monte Carlo (KMC), Molecular Dynamics (MD) and Phase-field Method simulations (PFM) sequentially. Each of these methods is working on a different time scale while, in addition, KMC and PFM are typically applied on different length scales. In this paper, we focus on the sequential coupling of KMC and MD simulations for the case of nano tensile tests of iron-copper alloys at different states of thermal aging [MBHS12], [MBM+14].

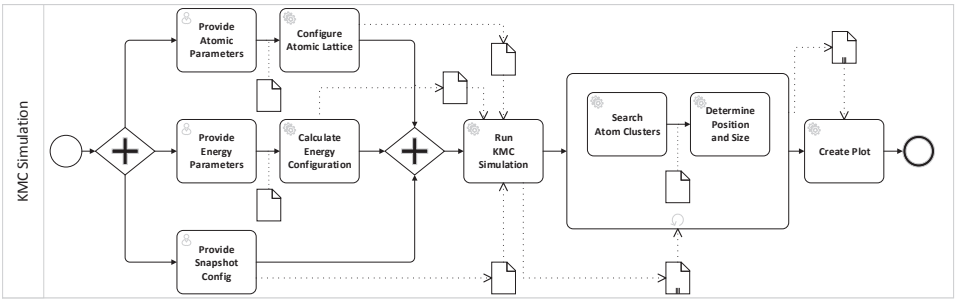

Figure 1: Simplified kinetic Monte Carlo (KMC) simulation workflow. Adapted from [SK13]

Fig. 1 shows a simplified workflow model of a KMC simulation using the custom-made simulation software **O**stwald ripening of **P**recipitates on an **A**tomic **L**attice (OPAL) [Bin06]. The modeling of OPAL as scientific workflow has been documented in [SHK+11]. OPAL simulates the formation of copper precipitates, i.e., atom clusters, due to thermal aging. The workflow receives a set of parameters such as atom concentration, energy values, and the number of intermediate snapshots to be taken from the scientist, configures the atomic lattice and calculates the energy configuration as input for the KMC simulation. According to the desired number of snapshots, the OPAL software saves the current state of the atom lattice at a particular point in time in a snapshot file. Every snapshot is then searched for atom clusters and their position and size. The result is visualized using an external visualization software.

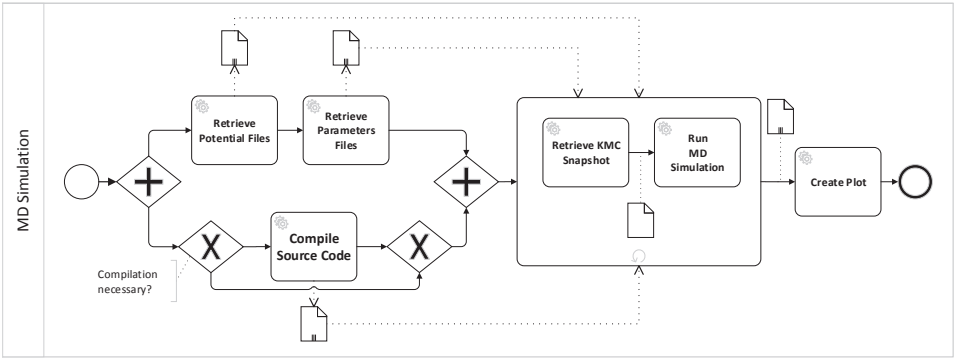

Figure 2: Simplified Molecular Dynamics (MD) simulation workflow

Fig. 2 depicts a simplified workflow of a Molecular Dynamics simulation. The MD simulation, which is implemented by the **I**TAP **M**olecular **D**ynamics (IMD) software pack-

age [SMT97], is used to study the tensile deformation of the snapshots generated in the kinetic Monte Carlo simulation. This simulation is computationally very costly and cannot be done in the KMC simulation due to the rigid lattice. The simulation services encapsulating the IMD software package and used as implementations for the activities in Fig. 2 have been developed in the scope of [Nem14].

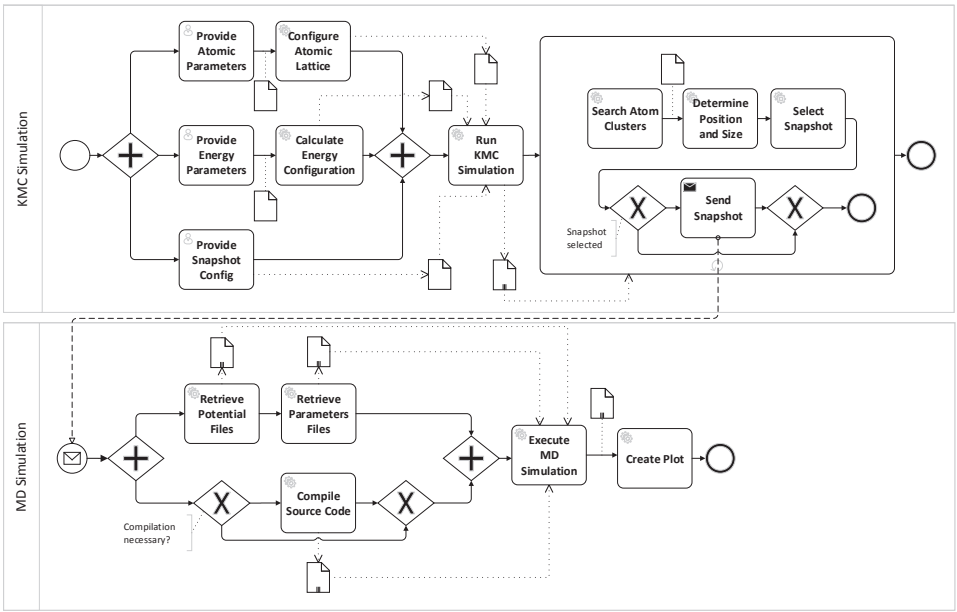

Figure 3: Bottom-up derived choreography forming a multi-scale simulation

In order to provide an adjusted simulation tool, the workflow compiles the source code for a particular computation platform if this has not already been done in a previous run. IMD runs on single cores as well as on computing clusters using the MPI interface[2] standard between computing instances. In parallel, the necessary inter-atomic potential and parameter files containing the simulation's boundary conditions are retrieved for the simulation run. Subsequently, for each selected KMC snapshot, an MD simulation instance is created and the computer-based tensile test is executed. The result is also visualized using an external visualization software.

Fig. 3 shows the target state after coupling both simulations. The coupled workflows form a choreography. Note that the visualization step after the KMC simulation has been removed in order to visualize after the combined multi-scale simulation. Moreover, the activity *Select Snapshot* uses some specific criteria to evaluate if a snapshot should be send to the MD simulation. The *Send Snapshot* activity sends every selected snapshot file or a reference to it to the molecular dynamics simulation. Without the workflow-based coupling the data transfer, the selection of the appropriate snapshot, and the subsequent triggering of the MD simulation for every selected KMC snapshot has to be conducted manually. This

[2] http://www.mcs.anl.gov/research/projects/mpi/

is especially cumbersome and error-prone if the number of generated snapshots is high. An automation of the coupling of both workflows would decrease the overall simulation time and manual errors due to file copying. However, automating data transfer using a implicit choreography is not sufficient to enable a flexible multi-* simulation modeling. We want to derive an explicit choreography model from the interacting workflows which can itself be adapted, e.g., by adding new participants, i.e., simulation methods to the overall multi-* simulation. These changes can then be propagated to the existing workflows to update them. In order to realize Model-as-you-go for choreographies, this has to be possible both with workflow models and already running instances without losing the instance state. Therefore, in Sec. 3 we provide a life-cycle supporting a round-trip modeling beginning bottom-up and in Sec. 4 we discuss different bottom-up derivation cases from existing workflows and to what degree this can be conducted automatically. Note that we do not limit our approach to sequential interaction of simulation methods/participants.

3 Bottom-up Life Cycle

In this section, we show a bottom-up view on a life cycle for modeling and executing multi-* experiments that are realized by choreographies introduced in [WK14]. A scientific Workflow Management system (sWfMS) implementing the life cycle enables the typical trial-and-error style of scientists when modeling scientific experiments [BG07], [SK10]. Since scientists want to be able to react to intermediate results during execution without modeling the experiment completely beforehand, the enactment of choreographies may be started even before the choreography model is completely specified. We call this *Model-as-you-go for choreographies*. The bottom-up life cycle is an extension of the scientific workflow life cycle introduced in [SK10], which itself has been implemented by an extended Business Process Management (BPM) life cycle. While in the traditional BPM life cycle there are several distinct roles, such as business analyst and IT specialist, in the domain of scientific experiments this is typically done by one role, the scientist. In order to ease the handling of the sWfMS, the technical complexities and the difference between workflow models and instances must be hidden, so that the actual phases for modeling on choreography and workflow level, the deployment, the execution, and monitoring are perceived as one experimentation phase by scientists.

Fig. 4 shows the bottom-up modeling approach view on the life cycle as the scientist experiences it. The life cycle starts with the *Workflow Modeling* phase and the modeling of executable workflows (1), which are interconnected. However, the interconnection is not explicitly captured by a choreography model. A graphical workflow editor is used for workflow modeling. The scientist is able to run the workflows and proceed to the *Execution and Monitoring* phase, the deployment is hidden behind the *Run/Resume* action. In this phase, scientists can suspend one or all running workflows using the *Suspend Workflow* action and thus returning to the Workflow Modeling phase in order to adapt them. Furthermore, the *Derive/Update* action can be used to derive a meaningful multi-* choreography model (2) from the interconnected workflows. If the choreography model has already been derived once, it is updated. Note that the derivation can be triggered either in the Execution and Monitoring phase under consideration of the already running workflow

instances and their state or directly from the Workflow Modeling phase only considering the existing workflow models. The derived choreography reflects the error handling, monitoring, and adaptation capabilities of the underlying executable workflows and services. It can be examined and adapted in a graphical choreography editor.

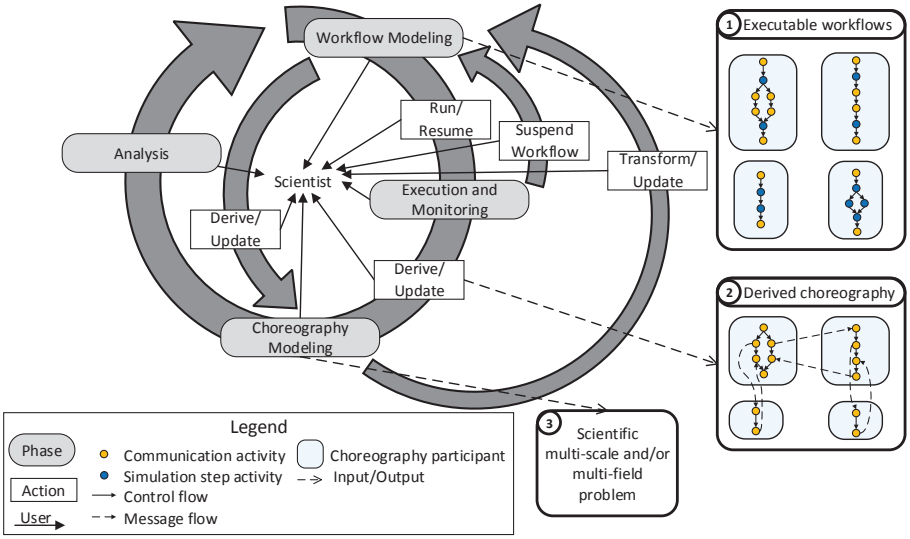

Figure 4: Bottom-up life cycle as perceived by the scientist

Finally, the derived choreography represents a specific scientific multi-scale and/or multi-field problem (3). The adaptation on the choreography level can be used to update the existing executable workflows in a *round-trip* fashion. Via the *Transform/Update* action the underlying workflow models can be modified and at the same time correctness of the changes is enforced. Thus, the scientist returns to the Workflow Modeling phase. The life cycle is concluded by the *Analysis* phase where the experiment results are evaluated.

A scientific Workflow Management system implementing this bottom-up view on the life cycle must have two additional technical phases (Fig. 5). The previously hidden *Deployment* phase realizes the Run/Resume action. In this phase, the executable scientific workflows are deployed onto execution engines and exposed as services that typically require the involvement of a service middleware, too. Similarly, the services that realize the experiment steps are also deployed on their execution environments. The Derive/Update action is technically realized by the *Derivation* phase. It comprises the automatic steps necessary to derive a meaningful choreography from the executable workflows or update the choreography model, respectively. Four adaptation cycles can be observed: In the *Execution and Monitoring* phase, the system is adapted along the functions and the logic dimension. Functions dimensions adaptation comprise the replacement of service implementations whereas the logic dimension can be adapted by refining abstract placeholders in the enacting workflows during run time. The other three adaptation cycles depict the capabilities of adapting the enacting workflows and the choreography along the logic dimension, i.e., the adaptation of the workflow and choreography constructs.

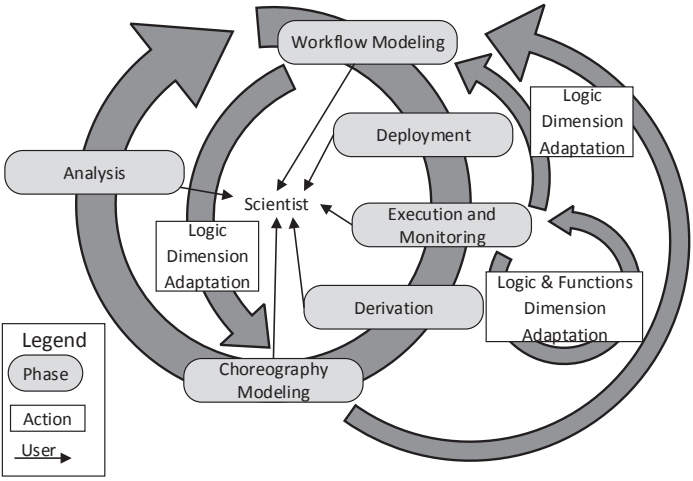

Figure 5: Bottom-up life cycle from the sWfMS perspective

4 Discussion of the Bottom-up Derivation

In this section we discuss the different starting points for a bottom-up derivation of a choreography model from existing simulation workflows. We assume that simulations using a single scale and single field have been modeled as workflows. When these workflows form a multi-* simulation, we distinguish between explicitly and implicitly connected workflows. *Explicitly connected* workflows contain explicitly defined links described in a workflow language. *Implicitly connected* workflows do not contain any explicit specification of a communication link between them. Nevertheless, the workflow models are implicitly connected by some manual or script-based action such as copying of files or invocation of a subsequent workflow. Note that these manual steps could be modeled as manual tasks in the used modeling language, however, we aim for automation when coupling workflows. The benefit of having a derived choreography model from an existing multi-* simulation is that it can be used as a starting point for iterative optimization procedures. Changes to the choreography model can be propagated to the underlying workflows. This helps to explicitly model the connection between implicitly connected workflows or to iteratively incorporate new simulation methods while the multi-* simulation is already running. A key question is whether the choreography derivation can be conducted automatically or needs manual steps and how we can support these manual steps.

Choreography derivation of explicitly connected workflows models: Fig. 6a shows a simplified illustration of explicitly connected workflow models picking up the motivating example from Sec. 2. A selected snapshot (or a reference to it) produced in the KMC simulation is directly sent to an MD simulation instance A possible derivation approach can use explicitly specified communication links between the simulation workflow models to discover the overall choreography. After the derivation the choreography model can be altered using a choreography editor, for example to incorporate a Phase-field Method simulation workflow into the overall simulation.

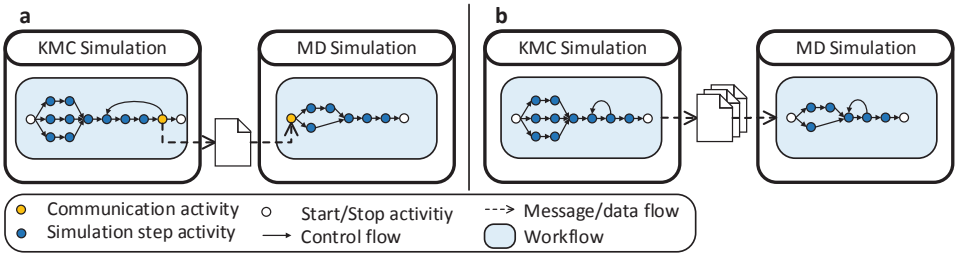

Figure 6: (a) Explicitly connected workflows, (b) implicitly connected workflows

Choreography derivation of implicitly connected workflow models: A more interesting use case for the bottom-up derivation are implicitly connected workflow models. In Fig. 6b, the two simulation workflows are only coupled implicitly. The modeled KMC simulation finishes the calculation for every given time step writing a snapshot to a file. Only when all time steps have been executed, the MD simulation can be started. The transfer of the selected snapshot files and the triggering of the subsequent simulation has to be conducted manually. An automatic derivation of a choreography model is not possible with a derivation approach relying on communication links specified in the workflow modeling language because there are no communication links available in this case. Mining the audit trails of the workflow execution engines is also not an option because the implicit connection will not occur in any execution log. However, modeling support for the definition of explicit interconnections between simulation workflows is still possible. A derivation algorithm has to use the involved workflow models and generate a new choreography participant for each workflow model. A scientist is then able to connect the generated choreography participants by manually drawing a message link in a choreography editor. The changed choreography model can be used to update the existing simulation workflows according to the scientific choreography life cycle presented in Sec. 3.

Choreography derivation of explicitly connected and running simulation workflows: The previous two cases only consider non-running multi-* simulations and the corresponding simulation workflow models. If the simulation workflows have already been started and are long-running because a scientist wants to simulate with a high level of detail, it might not be desirable to abort the simulation workflows in order to derive a choreography model. Rather, the derivation of a choreography model from explicitly connected and running simulation workflows must consider the execution state of the workflow instances. Before the derivation starts, all involved simulation workflows have to be suspended, i.e., the execution has to be paused. Subsequently, a derivation algorithm that recognizes explicitly described communication links between the workflow models can be used to derive a choreography model. Every workflow model is represented by a choreography participant in the choreography model. In contrast to the non-running workflow models before, the state of the paused workflow instance has to be attached to the corresponding choreography participant. If the scientist wants to re-execute, iterate or change the suspended workflow instances and, thus, the derived choreography, the attached instance state information has to be considered (see next paragraph).

Choreography derivation of implicitly connected and running simulations workflows: In the case of only implicitly connected, but already running simulation workflows collaborating as a multi-* simulation, the following approach can be used. The simulation workflows belonging to the same multi-* simulation are suspended. A choreography model is derived from the collaborating workflow models without relying on explicitly modeled communication links in the used workflow language. For every workflow model a choreography participant is generated. The execution state of the corresponding workflow instances is attached to the choreography participant – if several instances of one particular workflow model take part in the multi-* simulation, this also has to be reflected in the derived choreography model. In order to explicitly connect the choreography participants, communication activities have to be inserted on the sending and the receiving participant and a communication link has to be drawn using a graphical choreography editor.

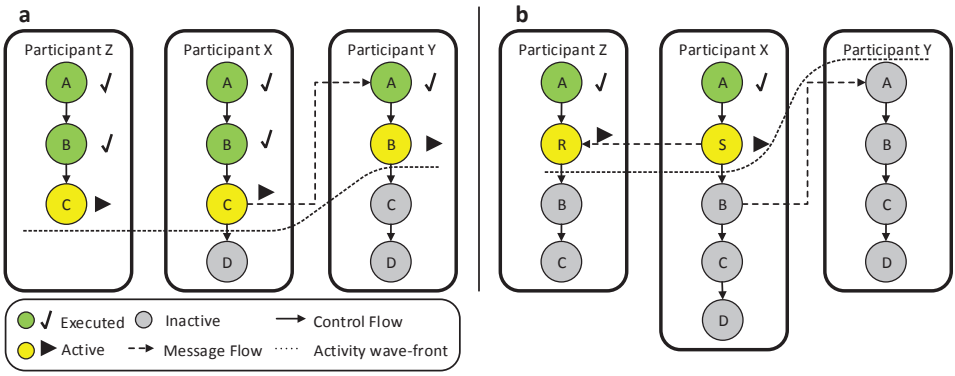

Figure 7: (a) Before the explicit connection of Participant A with Participant C , (b) after the explicit connection of Participant A with Participant C

This is straightforward if the communication link must be inserted ahead of the so-called activity wave-front [SK13], i.e., ahead of the set of already executed or currently active activities. If the insertion of the communication link has to be conducted behind the activity wave-front, i.e., between already executed or currently active activities, it is more complex. The activities after the new communication activity to be inserted may have to be compensated, i.e., semantically undone and re-executed to also consider the behavior of the newly inserted communication activity. The compensation is also necessary for further participants that have been already been invoked. This becomes clear when considering the example in Fig. 7. Fig. 7a shows a simplified scientific choreography with three Participants (X, Y, Z). While Participant X and Participant Y are explicitly connected via a message link from activity C in Participant X to activity A in Participant Y, Participant Z contributes to the overall multi-* simulation but is not explicitly connected to any other choreography participant. If the decision is made to explicitly incorporate Participant Z into the scientific choreography without aborting the already started multi-* simulation, this leads to the situation depicted in Fig. 7b. All activities following the new sending activity S and receiving activity R have to be compensated and old data snapshots [SK13] have to be loaded. Then, a message link can be inserted between the activities S and R and the changes can be propagated from the choreography model to the enacting workflow

instances. A detailed specification of the different compensation cases and the propagation to the workflow instances is out of scope in this paper but will be investigated in future.

5 Realization

In the following, we will discuss the realization options of a system supporting the life cycle and the derivation cases. As workflow language for simulation workflows, we use BPEL [OAS07], which has also been used in previous work in the scope of single scale and single field simulation workflows [SK13]. BPEL4Chor [DKLW07] is the choreography language, we have chosen to represent our bottom-up derived choreography models. BPEL4Chor is a non-executable choreography language forming an additional layer on top of the BPEL standard. In BPEL4Chor, the communication behavior is specified by the so-called Participant Behavior Descriptions (PBD), which are abstract, but more restricted BPEL processes. In the Participant Topology artifact, the structural aspects of a choreography such as the participants and the message links between them are specified. Simulation workflows can be modeled using the *Mayflower BPEL Designer* [SHK12], for choreography models we use our graphical choreography editor, the *ChorDesigner* [Sch14]. The derivation algorithms will be implemented in a *Transformer* component connecting both editors. The Transformer contains functions for both deriving choreography models and updating the underlying workflow models to support the modeling round-trip. The approach of Steinmetz [Ste07] can be applied in order to derive a BPEL4Chor choreography from explicitly connected simulation workflow models. The approach generates a participant for every workflow model and follows the declared BPEL Partner Links to derive the communication links between the participants. However, the approach is dependent on explicit communication links between workflows in order to derive a choreography model, thus, it is not directly suitable in the case of implicitly connected workflow models. In this case, we propose to generate choreography participants from the given BPEL workflows without relying on BPEL Partner Links. The BPEL workflows can be easily transformed into BPEL4Chor PBDs by turning them into abstract BPEL processes and removing Partner Links, Port Types and Operations as demanded by the Abstract Process Profile for Participant Behavior Descriptions [DKLW07]. For the derivation of a BPEL4Chor choreography model from running implicitly and explicitly connected simulation workflows, the derivation approach of Steinmetz has to be extended to attach the workflow instance state to the derived choreography participant. The Model-as-you-go operations for choreographies such as re-execution, iteration and iterative modeling of already started multi-* simulation workflow instances will be implemented in future.

6 Related Work

In [TKOAM08], a bottom-up approach for the interconnection of existing workflows to form choreographies spanning short-term virtual enterprises is presented. In order to preserve privacy for the participants, the approach only exposes the parts of existing workflows that are necessary for communicating with other participants using formal methods. Furthermore, the abstracted workflows are advertised and matched with potential partner workflows. In contrast, in our approach we do not conduct a semantic matching between

simulation workflows to find an complementary participant, since the simulation workflow may not be explicitly connectible when deriving a choreography model. Additionally, only exposing the external visible communication behavior is too strict for the scenario of simulation workflows when scientists want to perform iterative modeling in the choreography modeling phase. In [ea08], the authors introduce MUSE, a software for multi-scale and multi-physics simulations in the scope of astrophysics. Software modules implementing distinct physical methods on distinct scales are orchestrated using a Python layer. While this is a significant improvement compared to completely monolithic approaches, our approach is usable for different scientific fields and is based on simulation workflows, which also specify distinct steps inside the scope of one scale or field. Furthermore, we support the derivation of a global model in order to support the iterative addition of further simulation methods with respect to scales and/or scientific fields. In [DKB08], a generic choreography life cycle is presented. While the proposed life cycle supports choreography modeling beginning with a domain specific problem, i.e., top-down modeling, the life cycle presented in our work starts bottom-up with executable workflows from which a choreography model is derived.

7 Conclusions and Future Work

In this paper, we discussed the bottom-up derivation of scientific choreography models from existing simulation workflows interconnected as a multi-scale and multi-field simulation. Starting from a motivating scenario of only implicitly coupled simulation workflows for the studying of thermal aging of iron-copper alloys, we presented a choreography life cycle supporting the bottom-up derivation of choreography models and the propagation of changes to the underlying simulation workflows in a round-trip manner. Furthermore, we discussed several distinct starting points for the derivation, namely explicitly and implicitly connected simulation workflow models and already running simulation workflow instances. While we already have an choreography editor for adapting choreography models, in our future research we plan to implement the discussed derivation cases based on the choreography language BPEL4Chor as well as to develop concepts for the Model-as-you-go operations for choreographies.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

[BG07] R. Barga and D. Gannon. Scientific versus Business Workflows. In *Workflows for e-Science*, pages 9–16. Springer, 2007.

[Bin06] P. Binkele. *Atomistische Modellierung und Computersimulation der Ostwald-Reifung von Ausscheidungen beim Einsatz von kupferhaltigen Stählen*. PhD thesis, University

of Stuttgart, Stuttgart, 2006.

[DKB08] G. Decker, O. Kopp, and A. Barros. An Introduction to Service Choreographies. *Information Technology*, 50(2):122–127, 2008.

[DKLW07] G. Decker, O. Kopp, F. Leymann, and M. Weske. BPEL4Chor: Extending BPEL for Modeling Choreographies. In *ICWS'07*. IEEE, 2007.

[ea08] S. Portegies Zwart et al. A Multiphysics and Multiscale Software Environment for Modeling Astrophysical Systems. In *ICCS'08*, pages 207–216. Springer, 2008.

[HTT09] T. Hey, S. Tansley, and K. Tolle, editors. *The fourth paradigm: data-intensive scientific discovery*. Microsoft Research, 2009.

[MBHS12] D. Molnar, P. Binkele, S. Hocker, and S. Schmauder. Atomistic multiscale simulations on the anisotropic tensile behaviour of copper-alloyed α-iron at different states of thermal ageing. *Philosophical Magazine*, 92(5):586–607, 2012.

[MBM+14] D. Molnar, P. Binkele, A Mora, R. Mukherjee, B. Nestler, and S. Schmauder. Molecular Dynamics virtual testing of thermally aged Fe-Cu microstructures obtained from multiscale simulations. *Computational Materials Science*, 81(0):466 – 470, 2014.

[MMC+12] D. Molnar, R. Mukherjee, A. Choudhury, A. Mora, P. Binkele, M. Selzer, B. Nestler, and S. Schmauder. Multiscale simulations on the coarsening of Cu-rich precipitates in a-Fe using kinetic Monte Carlo, molecular dynamics and phase-field simulations. *Acta Materialia*, 60(20):6961–6971, 2012.

[Nem14] Markus Nemet. Kapselung von bestehenden Simulationsanwendungen mit Hilfe von Web Services. Bachelor thesis, 2014.

[OAS07] OASIS. Web Services Business Process Execution Language Version 2.0, April 2007.

[Sch14] J. Schilling. Analyse und Erweiterung eines bestehenden Choreographiewerkzeugs. Student Thesis No. 2433, University of Stuttgart, 2014.

[SHK+11] M. Sonntag, S. Hotta, D. Karastoyanova, D. Molnar, and S. Schmauder. Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications. In *ServiceWave'11*, pages 1–12. Springer, 2011.

[SHK12] M. Sonntag, M. Hahn, and D. Karastoyanova. Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In *CEUR Workshop'12*, pages 1–5. Springer, 2012.

[SK10] M. Sonntag and D. Karastoyanova. Next Generation Interactive Scientific Experimenting Based On The Workflow Technology. In *MS'10*. ACTA Press, 2010.

[SK13] M. Sonntag and D. Karastoyanova. Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. *Grid Computing*, 11(3):553–583, 2013.

[SMT97] J. Stadler, R. Mikulla, and H. R. Trebin. IMD: A software package for molecular dynamics studies on parallel computers. *International Journal of Modern Physics C*, 8(5):1131–1140, 1997.

[Ste07] T. Steinmetz. Generierung einer BPEL4Chor-Beschreibung aus BPEL-Prozessen. Student Thesis No. 2101, University of Stuttgart, 2007.

[TKOAM08] S. Tata, K. Klai, and N. Ould Ahmed M'Bareck. CoopFlow: A Bottom-Up Approach to Workflow Cooperation for Short-Term Virtual Enterprises. *Services Computing*, 1(4):214–228, 2008.

[WK14] A. Weiß and D. Karastoyanova. A Life Cycle for Coupled Multi-Scale, Multi-Field Experiments Realized through Choreographies. In *EDOC'14*, pages 1–8. IEEE, 2014.