Institute of Architecture of Application Systems

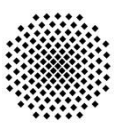
Enabling coupled multi-scale, multi-field experiments through choreographies of data-driven scientific simulations

Andreas Weiß, Dimka Karastoyanova

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{andreas.weiss, dimka.karastoyanova}@iaas.uni-stuttgart.de

BibTeX:

```
@article {ART-2014-10,
 author      = {Andreas Wei{\ss} and Dimka Karastoyanova},
 title       = {{Enabling coupled multi-scale, multi-field experiments through
                choreographies of data-driven scientific simulations}},
 journal     = {Computing},
 publisher   = {Springer Wien},
 pages       = {1--29},
 type        = {Article in Journal},
 month       = {October},
 year        = {2014},
 department  = {University of Stuttgart, Institute of Architecture
                of Application Systems},
}
```

University of Stuttgart
Germany

Computing manuscript No.
(will be inserted by the editor)

Enabling Coupled Multi-Scale, Multi-Field Experiments Through Choreographies of Data-Driven Scientific Simulations

Andreas Weiß · Dimka Karastoyanova

Received: date / Accepted: date

Abstract Current systems for enacting scientific experiments, and simulation workflows in particular, do not support multi-scale and multi-field problems if they are not coupled on the level of the mathematical model. To address this deficiency, we present an approach enabling the trial-and-error modeling and execution of multi-scale and/or multi-field simulations in a top-down and bottom-up manner which is based on the notion of choreographies. The approach defines techniques for composing data-intensive, scientific workflows in more complex simulations in a generic, domain-independent way and thus provides means for collaborative and integrated data management using the workflow/process-based paradigm. We contribute a life cycle definition of such simulations and present in detail concepts and techniques that support all life cycle phases. Furthermore, requirements on a respective software system and choreography language supporting multi-scale and/or multi-field simulations are identified, and an architecture and its realization are presented.

Keywords Simulation Workflow · Scientific Workflow · Choreography · Multi-Scale, Multi-Field Experiment · Data-driven Scientific Simulation · Life Cycle

Mathematics Subject Classification (2000) 68U20 · 68U35 · 68M14

Andreas Weiß
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
E-mail: andreas.weiss@iaas.uni-stuttgart.de

Dimka Karastoyanova
Institute of Architecture of Application Systems (IAAS)
University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
E-mail: dimka.karastoyanova@iaas.uni-stuttgart.de

1 Introduction

eScience is a very active field focusing on providing IT support for scientists from different natural and social sciences fields for the purpose of faster scientific exploration and discovery. In eScience, the objective is to provide generic approaches and tools to support its whole life cycle: from data collection and curation, through data processing and analysis to permanent archiving of discoveries [20]. One approach for data processing and analysis is the workflow technology, also known as scientific workflows, used for enabling scientific simulations. Note that scientific workflows have different characteristics in comparison to the workflow technology applied in business applications for Business Process Management (BPM). Existing scientific Workflow Management systems (sWfMS) can be grouped in two major categories: domain-specific and generic systems. The domain-specific systems are typically developed in cooperation with scientists and meet the requirements of one (or just a few) scientific domains. In most cases the sWfMSs support modeling of a workflow of data management or computing tasks and focus on how data is processed and in what sequence and which data sources are to be used. These systems hide the complexity of distribution and parallelization of computational tasks from scientists but sometimes require from scientists to use one specific technology. Usually, an optimized workflow model is derived which can be executed multiple times. However, fault and exception handling are not supported in an automated manner by means of modeling constructs and techniques and in addition the trial-and-error nature of scientific discovery is not always supported by these systems. Workflows are executed without interruptions and in most cases do not allow human users to interact with the workflow. Examples of domain-specific systems are Kepler [24], Taverna [26], and Triana [43].

The second category of workflow systems is a more recent development and makes use of the conventional workflow technology known from business applications. The advantages of business workflows like fault handling, forward and backward recovery and since recently workflow flexibility approaches for trial-and-error experimenting are drawn upon. The issues that are currently being dealt with in research are related to improving the support for data processing in a generic manner considering the huge amount of data available or produced, and the composition and integration of existing software into interoperable systems. Moreover, the huge complexity of eScience due to its interdisciplinary nature, software engineering aspects, and knowledge management supporting such systems have to be addressed in the same context.

Both types of systems support the modeling and execution of simulation workflows that can realize simulations on a single scale, i.e., metric or time scale, and/or on a single field model describing the scientific phenomenon, for example in plant molecular biology [19]. The more complex multi-scale and/or multi-field (also called multi-*) simulations can also be supported by these technologies and systems if the mathematical models implemented through the simulation software are 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 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 (or sub-fields) in the same experiment, e.g., physics, biology, or chemistry. An example for a multi-scale simulation is the remodeling of bones using the Theory of Porous Media [22]. Here, the cell and tissue model in the human body can be coupled with a bone model. Another example is the simulation of thermal aging of iron-copper alloys and emerging effects of existing precipitates on the mechanical behavior in the material science domain [29]. 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 each describing the phenomena of precipitation from a different point of view.

Based on our experience with scientists and experts from industry, where simulation is also a key enabler, simulations that are not based on models inherently coupling the different scales and physics of the natural phenomena also need to be supported. Especially important is the case where collaboration among distinct scientific groups or industry organizations is desired, which implies coupling of existing simulation software into complex simulations and the correlation of the interactions among them. Towards this goal, we present an approach that utilizes the notion of choreographies to enable the trial-and-error modeling and execution of multi-* simulations (Sec. 3). The approach considers top-down modeling starting from scratch with a multi-* domain problem as well as bottom-up modeling combining existing simulations. We contribute a life cycle definition of such simulations and present in detail concepts and techniques that support all life cycle phases, however, the main focus in this article lies on the modeling aspects. We also identify the requirements on a software system that can support the life cycle of multi-* simulations as well as the requirements on a choreography language (Sec. 4). One main objective has been to reuse as much as possible of existing standards and techniques as possible, while providing a user-friendly system to scientists, who are both the developers and users of the simulations. Additional contribution of this work is the definition of an architecture of the system for support of multi-* simulations and a corresponding realization (Sec. 5). The realization builds on top of a sWfMS that is based on the workflow technology from the business process application field and novel flexibility approaches introduced in previous work (Sec. 2). 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 Background and Motivation

2.1 Background

In this section we present the requirements of eScience as introduced in our previous research work and will derive additional requirements for the ap-

proach and life cycle for multi-* simulations. The approach presentation is based on our existing research publications. In our research in the scope of the Cluster of Excellence Simulation Technology (SimTech[1]) our goal is to enable IT support for scientists in their work on developing scientific simulations. Major starting requirements in this work have been to support the complete scientific simulations life cycle characterized by its trial-and-error nature and to maintain user-friendliness of the solution. After comparison with existing work in the field of scientific experiments and simulations, and the research in the BPM field mostly related to workflow management [40], [16], [15], [13], [20], we have designed a service-oriented, workflow-based approach for modeling, execution and monitoring of scientific simulations and a corresponding interactive, flexible, SOA-based scientific workflow system enabling this approach by means of concepts, architecture and implementation. The major components involved are: a modeling tool, a workflow engine, a service bus, and a monitoring component. In our work we make use of Web Services [45] to enable interoperability and the easier integration of simulation software.

For the modeling of scientific experiments we extended the existing workflow technology known from business applications [23] with features needed by scientists. Basically, using a workflow to model a simulation, or any other kind of experiment, involves specifying a number of steps (called activities) that have to be carried out as well as the involved control flow and data flow. The steps in such a workflow, which we also call simulation workflow, are typically data processing steps such as copying data from one location to another, solving a set of mathematical models (differential equations solvers, sequencing algorithms etc.), visualization steps, preprocessing of data, and many others. A workflow modeling tool is the infrastructure component supporting the modeling step, typically with a graphical notation. In addition to constructs for modeling the control flow of interactions of simulation modules and the data exchanged among them and with the user, we explicitly support data management activities and domain-specific activities, which are also part of the construct catalog in our modeling tool [37]. The data management activities are abstract and can be used to model storing, retrieving, and manipulating scientific data from different data source types. Depending on the context in which such an activity is used, we define mappings of such an activity to a predefined template realizing complex operations on data, the necessary interactions with data sources and performing the required format transformations [33]. The domain-specific activities stand for complex sequences of domain-specific tasks orchestrating several simulation modules/services that we provide to the users hidden behind individual activities. Using a domain-specific activity leads to a subsequent code generation that adds the actual workflow code into the model. For the purpose of reusability we also defined and utilized the concept of workflow fragments capturing predefined workflow logic. Fragments are stored in the fragment library Fragmento [36]. They can also be used to capture both templates for data management and work-

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

flow logic realizing a domain-specific activity. Scientists can start simulations from our modeling tool just by a simple click of a button, without performing any additional steps. This contributes to the user-friendliness of the tool. The realization of the tool incorporates, on the level of both architecture and implementation, support for deployment of the workflow model on a workflow engine, the provision of the workflow as a service, and the subsequent instantiation of a workflow instance – all these steps are hidden from the user.

The workflow engine navigates through the workflow model and delegates the invocation of the individual activities to the service bus. Since the individual activities are implemented by services hiding simulation software, the execution of the simulation software is done by the concrete execution environment. The resulting data or data reference is brought back to the engine by the service bus. The monitoring component is typically responsible for collecting execution data about process instances and providing them to a monitoring tool for visualization. For scientists this is in contradiction to the trial-and-error nature of their work. For this purpose, we enabled the simultaneous modeling and execution of workflows. The scientists can define only a part of the workflow they would like to carry out, start its execution (on the engine) and add additional steps in the workflow while it is being executed. We denoted this concept *Model-as-you-go* approach [39], [37]. 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. The approach allows for the interactive modeling and execution of scientific workflows and contributes also to the field of flexible workflows and service compositions. The corresponding infrastructure uses the monitoring component to transfer monitoring/status information to the modeling tool and back, so that both execution engine and modeling tool have all the information about the state of the executed workflow instance and the newly modeled activities. We enable the visualization of the workflow execution state directly in the modeling tool and incorporated a stop/resume functionality to enable the interactive completion of the workflow. These functionalities have the corresponding counterpart components at the workflow engine.

With our original approach we can support user friendly modeling, execution and monitoring of scientific workflows for cases in which existing simulation and data processing software has to be orchestrated automatically in a specific order. Such simulation workflows can realize simulations on a single scale and/or scientific model or multi-scale/field simulations if the mathematical models implemented through the simulation software are already coupling the different scales/fields. The latter case typically uses approximation of one or more scales/fields to another scale/field. Based on requirements provided by scientists and industry experts we identify two basic scenarios that need to be supported: (i) there is existing software implementing different mathematical models and/or scales that need to be orchestrated, very often across organizations, i.e., a bottom-up approach, or (ii) one or more organizations need to support a particular multi-scale/field simulation and starts its modeling and realization from scratch, i.e., a top-down approach. In both scenarios the ma-

jor open issue with respect to the modeling is how the interactions among the participating simulations and the data exchange can be represented.

2.2 Choreographies for Multi-* Simulations

In this work, we use choreographies to couple scientific experiments from distinct scientific fields into combined multi-* experiments. Choreographies are a concept known from the business domain that enables independent organizations to collaborate and reach a common business goal. They provide a global view on the interconnection of independent organizations communicating without a central coordinator [9], [49]. Therefore, choreographies are coordinated peer-to-peer-like interactions between services or orchestrations of services (i.e., workflows). 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 individual organizations (and their workflows/orchestrations) are called *choreography participants*. In the context of our approach, every scientific model based on a separate scientific field or using a different scale is implemented as an orchestration of scientific services and the overall experiment is represented by a choreography without centralized control. The benefits of using choreographies for modeling and executing multi-* experiments are the following: (i) different scientists or groups of scientists can model the global view on a multi-* experiment, i.e., the public interfaces of each participant, while the implementation of actual experiment workflow logic can be conducted separately by the group having the proper expertise. (ii) Choreography participants and their enacting workflows are loosely coupled in comparison to sub-workflows. Each enacting workflow has its own life cycle and is not tightly coupled to a parent workflow. This allows specifying appropriate fault handling and compensation logic for each potentially long-running workflow individually. No parent workflow might fail and thereby affect subordinated workflows. (iii) The communication patterns between participants of a choreography can be more complex than simply request/response patterns [4]. Instead, the enacting workflows can engage in complex communication patterns, i.e., conversations that allow modeling a scientific experiment where intermediate feedback between multiple scales and/or fields can be incorporated into the computation.

2.3 Motivating Scenario

In order to further illustrate the properties multi-* experiments, we present a motivating scenario from the domain of material science [47]. Molnar et al. have studied the thermal aging of iron-copper alloys [29] and emerging effects of existing precipitates on the mechanical behavior [27], [28] 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. To motivate our approach, 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 [27], [28].

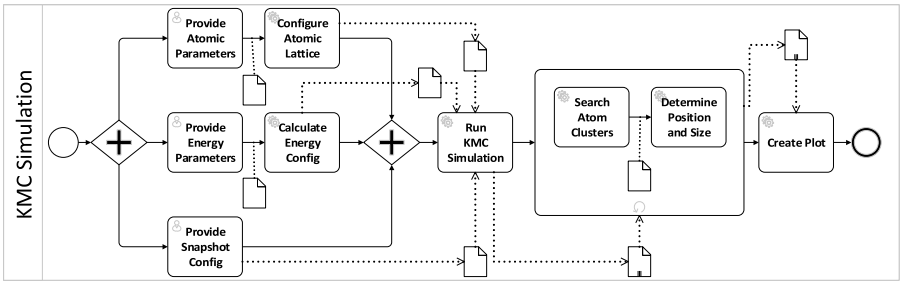


Fig. 1 Simplified kinetic Monte Carlo (KMC) simulation workflow [47]

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) [5]. The modeling of OPAL as scientific workflow has been documented in [38]. 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. The snapshots are then searched for atom clusters and their position and size. The result is visualized using an external visualization software.

Fig. 2 depicts a simplified workflow of a Molecular Dynamics simulation. The MD simulation, which is implemented by a different research group in the **I**TAP **M**olecular **D**ynamics (IMD) software package [41], 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. 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 ten-

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

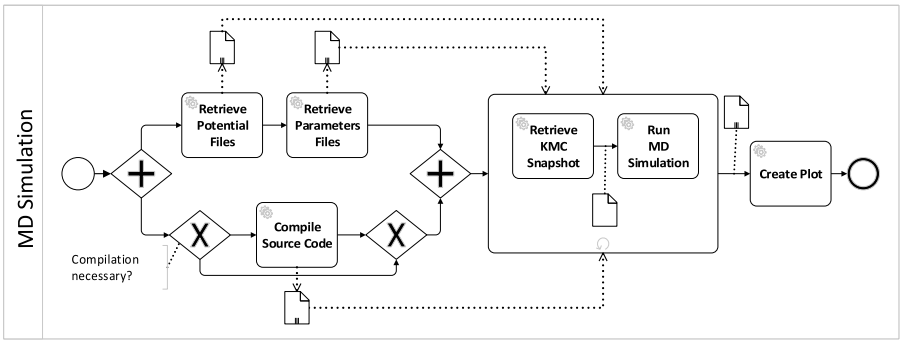


Fig. 2 Simplified Molecular Dynamics (MD) simulation workflow [47]

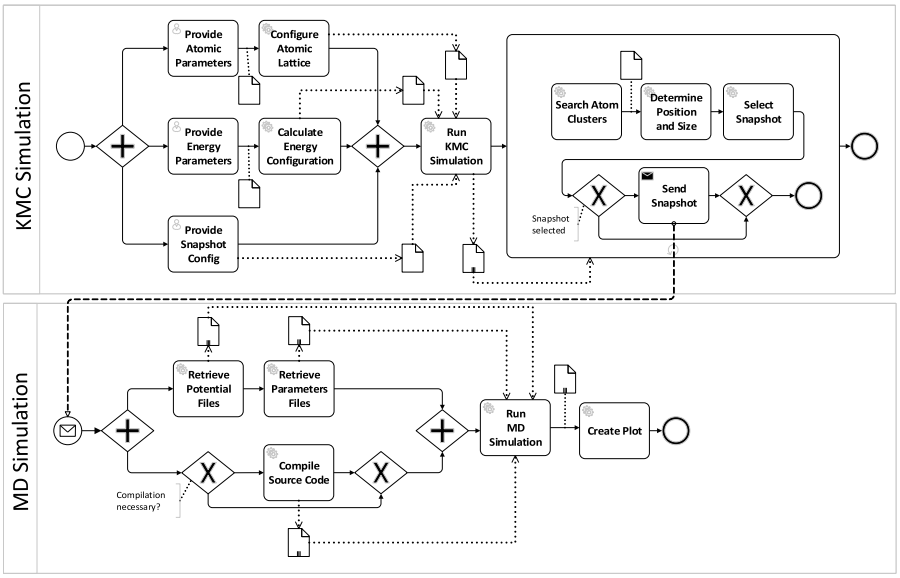


Fig. 3 KMC and MD simulation forming a coupled multi-scale simulation [47]

sile test is executed. The result is also visualized using an external visualization software.

Fig. 3 shows both the KMC and the MD simulation in a coupled manner. Both simulations form a choreography without a centralized coordinator. Note that the choreography is different from the previously presented individual simulation workflows. The visualization step after the KMC simulation has been removed in order to visualize the results after the combined multi-scale simulation. Moreover, the activity *Select Snapshot* uses some specific criteria to evaluate if a snapshot should be sent 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 has to be conducted manually. This is especially cumbersome and error-prone if the number of generated snapshots is high or if the software for the different simulations belong to geographically distant scientific groups. An automation of the coupling of both workflows would decrease the overall simulation time and manual errors due to file copying. Furthermore, parameter sweeps with different alloys can be automated. Note that although the motivating example only needs a sequential coupling, our approach also provides support for interleaved communication between scientific workflows.

3 Approach

As shown in Sec. 2, previous work has enabled scientists to model and execute scientific workflows that orchestrate scientific services coping with either one single scientific field and one single scale, or with services combining distinct scientific fields and different scales into one merged scientific model. As a logical continuation, it is our goal to enable scientist to model and execute multi-scale and multi-field scientific experiments in an easy and user-friendly manner for the cases not supported by existing work. It should be possible to couple scientific workflows without distracting scientists from their core tasks, i.e., formulating scientific models, and performing a corresponding experiment. Furthermore, it should be possible that different scientist model different parts of a scientific choreography according to their scientific expertise in a collaborative manner. As in previous work, we focus on using proven, standardized methods and technologies of the business domain [25]. Whenever necessary, we extended these technologies without transgressing their standard specification. As pointed out in Sec. 2.2, we use the concept of choreographies to flexibly model and execute multi-* simulations. With reference to previous work [37], [25], and [39] we propose the notion of *Model-as-you-go for choreographies*. For the scientist the technical complexities and the difference between workflow models and instances must be hidden. Instead, the phases for modeling on choreography and workflow level, the deployment, execution, and monitoring are merged to realize the appropriate experimentation process for scientists.

Our approach for modeling and executing scientific experiments as choreographed orchestrations of scientific services is one example of Collaborative, Dynamic & Complex systems (CDC) as described in [1]. CDC systems exhibit three different life cycle phases, namely Modeling, Provision, and Execution (Sec. 3.3). In this article we emphasize the Modeling phase and do not discuss the other two phases exhaustively. Details on the Provision and Execution phases have been reported in [1].

3.1 Modeling

The goal of the Modeling phase is to enable scientists to model multi-* experiments in a user-friendly manner. Modeling can be started top-down or

bottom-up. Both modeling approaches must enable the typical trial-and-error style of scientists when modeling scientific experiments [3], [25]. The overall choreography may not be complete in the sense that it models the whole problem the scientist wants to cover. Scientists want to react to intermediate results during execution without modeling the experiment completely beforehand. While in the classical, i.e., business-driven BPM context there exist several distinct roles, such as a business analyst modeling the workflow and an IT specialist responsible for deployment and monitoring, in the domain of scientific experiments this is typically done by one role, the scientist. However, the role can be taken by several individuals at the same time when scientists work together.

Top-down vs. bottom-up modeling: In the *top-down* approach (cf. Fig. 4a), scientists start from a scientific multi-scale and/or multi-field problem (1) and model the experiments from the distinct scientific fields as participants of a choreography. The modeled choreography (2) provides a global view on the communication between distinct single-scale and single-field experiments. The modeling is done manually using a choreography editor. The editor should provide a graphical notation and abstract away from choreography languages. In each participant only the activities and control and data flow constructs necessary for the communication with other choreography participants are modeled. That means, each participant exposes only its public communication interface. The orchestration logic of each single-field and single-scale experiment part is not explicitly modeled in this step. Since choreography modeling languages are often not executable [9], the scientific choreography is transformed into an abstract representation of an executable workflow language (3). For the transformation three cases can be distinguished: (i) For each participant a separate workflow model is newly generated, which only contains the communication constructs such as send and receive activities. (ii) A scientist or a group of scientists has already created a partial (incomplete) choreography model before. The transformation step has to be aware of this and already existing communication constructs must be updated in the workflow models accordingly. (iii) A long-running experiment has already been started before and has been paused to conduct Model-as-you-go operations in the choreography editor. This includes the introduction of new modeling elements (i.e., new scientific fields and/or scales) in form of choreography participants, or the iteration/re-execution of parts of the coupled multi-* experiment. In this case, the existing instance state of the enacting workflows must be considered.

An orchestration editor is used to conduct a manual refinement of the generated/updated workflow definitions. In the refinement step, one or several scientists add the internal orchestration logic for every distinct single-scale and single-field experiment participating in the overall choreography (4). Different scientist may refine different workflows according to their scientific expertise in a collaborative manner. For example, a physicist may refine a workflow simulating forces on a bone model, whereas a biologist covers a workflow operating on the biological cell level. They add activities and activity implementations

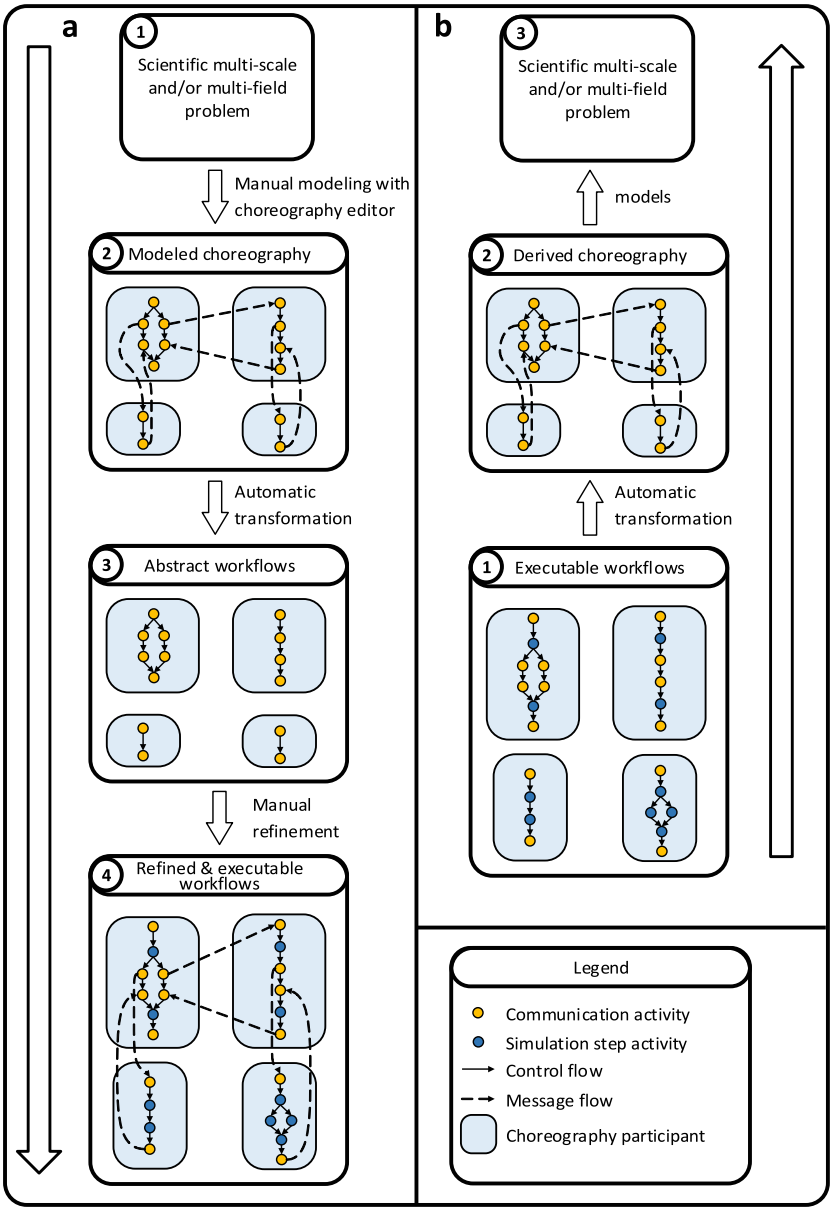


Fig. 4 Top-down (a) vs. bottom-up (b) modeling approach

to the workflow to make it executable. It must be possible for the scientists to model a particular workflow only partially with regard to the scientific domain problem, i.e, the only partially modeled workflows must be executable. However, this may lead to situations where the overall choreography can not yet be executed as the activities producing the results of one workflow that

are necessary for another workflow are not yet completely modeled. Changes made on the choreography must be checked for consistency and correctness, since they may influence the communication activities among choreography participants. For integration purposes of legacy experiment services it is possible to slightly deviate from the presented top-down approach. After modeling a participant, scientists can indicate that the implementation will not be a workflow but an application with a specified interface. In the transformation step this has to be considered. However, while this enables interoperability, Model-as-you-go for choreographies operations cannot be conducted on this particular participant as the composition of simulation steps is realized as a black box. The same is true if one scientist/research group wants to hide the internal logic of their scientific workflow from other groups.

In Fig. 4b the *bottom-up* modeling approach is depicted. Following this approach, multiple workflows representing distinct scientific fields on different scales are already existing (1). However, a corresponding choreography model explicitly capturing the interconnection is missing. A transformation step is necessary to derive a meaningful multi-scale and/or multi-field choreography (2) from the interconnected workflows. 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 the choreography editor, i.e., participants can be added or deleted and communication constructs and links can be altered. This should also be possible after the underlying workflows have already been started. In this case, the underlying workflows have to be paused and the instance state must be considered in the derivation and adaptation step. 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 again in a *round-trip* fashion. The update action must modify the underlying workflow models and their instances if they have already been started. At the same time the correctness of the changes has to be enforced. Participants in the bottom-up modeling approach can also be derived from black-box applications without exposed control flow for interoperability. However, this also reduces flexibility and restricts Model-as-you-go operations.

3.2 Provision

The following subsection shortly discusses the provisioning aspects of our approach for scientific choreographies. With regard to CDC systems the *Provision* phase is the second phase after modeling a choreography as global view on a multi-* experiment, transforming it into a set of abstract workflows, and refining them into executable ones. In the Provision phase the refined scientific workflows are deployed onto execution engines and exposed as services that typically requires the involvement of a service middleware, too. Similarly, the services that realize the experiment steps are also deployed on their corresponding execution environments. Therefore, appropriate deployment descrip-

tors have to be generated and configured. In order to automate the deployment steps and the management during run time, the service and workflow topology has to be captured and deployable packages containing services and workflows must be built [44]. The necessary monitoring infrastructure is configured using the requirements defined in the modeled choreography. Context and correlation data identifying workflow instances participating in a particular choreography may have to be initialized. As the scientific choreographies may be used by several scientists in parallel, the underlying infrastructure must be capable of mapping interactions with the system to distinct tenants and users. The technical complexity of the Provision phase must be invisible to the scientist. Instead, pressing one Run/Resume button in the modeling environment must trigger the provisioning and execution or resume it after adaptation actions, respectively.

3.3 Execution

The third phase of our approach is the *Execution* phase of scientific choreographies. This is achieved by executing the refined enacting workflows and scientific services participating in the choreography. Both the enacting workflows and the services are potentially distributed on several execution engines and service execution environments. The overall execution environment, i.e., the workflow execution engines and the service execution environments, have to support context-awareness and adaptation mechanisms to enable the flexibility needed for performing *Model-as-you-go for choreographies* operations. Some of the adaptations may be predefined in the choreography model, such as abstract activities/placeholders that have to be refined at runtime [7], reactions to context changes, and binding strategies for the simulation services [44]. Other adaptations are inherent to the execution phase and have to be addressed in an ad hoc manner. Examples are the manual adaptations of the choreography model during execution conducted by a scientist when performing Model-as-you-go operations both on the workflow and choreography level, the forced termination of the choreography, the substitution of scientific service endpoints, the substitution of choreography participant, or insertion of additional ones, and others. The overall execution environment must also be able to cope with an increasing number of scientists, simulations participants, and interactions between the participants, i.e., the system must scale with its load both in terms of computation power needed and data used. Observations during the Execution and Monitoring phase can be incorporated into new versions of the choreography model and trigger further executions of the scientific experiment. *Execution and Monitoring* of the running choreography must be indistinguishable for the scientist as in previous work [25]. This is motivated by the fact that for the scientist the monitoring of a running workflow instance is the visual representation of it. After the execution all provisioned systems and services have to be de-provisioned [44].

4 Requirements on a System for Choreographies of Scientific Simulations

In our previous work, we have identified requirements on scientific Workflow Management systems in general and in particular on monitoring and adaptability features of such systems [25]. In this work these requirements still apply and in this section we extend them to account for the requirements arising from our approach for multi-* experiments we presented in the previous section. We group the identified requirements in two classes.

Requirements on the supporting software system: The following list provides a number of requirements that have to be met from a software system supporting our approach.

SR1. Similarly to the customizable monitoring component integrated into our existing workflow modeling tool, the choreography editor must be enhanced to capture all running workflow instances belonging to a particular choreography to enable monitoring of the choreography itself. This would provide for the desired amalgamation of the choreography modeling and monitoring phases.
SR2. The system should provide a facility to inspect already executed choreographed workflow instances for analysis.
SR3. Facilities for aggregated statistical data over choreographed workflow instances and resource consumption are necessary for analysis purposes.
SR4. Scientists must be allowed to steer the choreography, i.e., to run, suspend, and resume individual workflow instances and the choreography as a whole. This must be possible without exposing the underlying complexity such as deployment steps to scientists. This facility enables the blending of modeling and execution phases.
SR5. The system should provide means to prohibit the alteration of communication activities on the workflow level in order to ensure compliance with the choreography model.
SR6. Straightforward manual adaptation facilities on the choreography level must be available. This includes the joining and leaving of participants in the choreography, i.e., the adaptation of the communication activities after starting the enacting workflows. For this, the Model-as-you-go operations from previous work [39], i.e., iteration and re-execution of workflow logic in single workflow instances must be extended to allow this behavior for collaborating choreographed workflow instances.
SR7. Adaptations on the choreography level must be checked for correctness to ensure that a choreography is still enactable.
SR8. It must be possible to execute the choreography, i.e., the enacting workflows, even if they are not completely refined/modeled to enable the trial-and-error manner of working in scientific explorations.
SR9. Already defined experiment activities and possible execution state must be preserved when transforming the updated choreography to the en-

acting scientific workflows for a second time. The same is true for the bottom-up derivation.

SR10. In order to react to infrastructure changes and exceptions, automatic adaptation facilities at run time must available. The scientist should not be aware of adaptations for failure handling purposes.

SR11. The adaptations on workflow and choreography level must be tracked to ensure the completeness of provenance information. The scientist must have facilities to inspect provenance information [8].

SR12. The workflow and the choreography editor should use similar graphical concepts to help scientists understand the two complementary editors more quickly.

SR13. A mechanism is needed for correlation of all choreographed workflow instances forming one virtual choreography instance. This is necessary as one choreography model can be instantiated more than once.

SR14. The scientific choreography system needs facilities for managing experimental context. This can for example be shared input parameters, shared intermediate experiment results that are produced by one enacting workflow and used by another enacting workflow as input, or shared data objects, simulation clocks, and others. The context either directly contains the needed objects or stores references. Synchronization mechanisms must be made available to coordinate concurrent access of different workflow instances belonging to one choreographed experiment. The manipulation of context information of one particular choreography must also be tracked in order to have complete provenance information about the conducted experiments.

Requirements on the choreography language: The following list provides a set of choreography language requirements, which have to be met by any choreography language that would be used for realizing our approach. Our approach is not tied to a particular language though.

CR1. For both modeling approaches described in Sec. 3.1 an appropriate choreography language has to be used. The language should preferably cover all service interaction patterns as identified in [4]. This facilitates the modeling of not only standard communication patterns such as send/receive or one-way messages but also more advanced patterns such as one-to-many send/receive or contingent requests. The latter ones are requests that are sent from a sender S to a participant Y if a previous request to participant X does not lead to a response in a predefined time frame.

CR2. The choreography language should avoid issues with locally unenforceable behavior [49]. For example, it must not be possible to model the sending of messages between the distinct simulations that depend on the sending of other messages but it is left unspecified how the current sender can learn of the previous messages between the choreography participants. The choreography language must ensure that the control flow

of every participant is explicitly specified. However, deadlocks could still occur if messages between participants never arrive, but can be avoided if the language provides appropriate timeout constructs [11].

CR3. The choreography language should provide explicit modeling constructs for handling faults occurring when executing the choreographed workflows.

CR4. As previous work in the simulation workflow domain has successfully used conventional workflow technology, it is desirable also to have a choreography language that is compatible with this approach.

CR5. The choreography modeling language must provide elements that facilitate an easy adaptation such as abstract communication activities that can be refined at run time [7].

5 Realization

In this section we discuss the conceptual architecture and the current state of the realization of our system.

5.1 Architecture

Fig. 5 shows the architecture we propose for our scientific choreography support system. The software system requirements from Sec. 4 are mapped to the components addressing them. The gray shaded figures have already been implemented, or are reused from previous work or other research group members. As an instance of a Collaborative, Dynamic & Complex system, our architecture has three distinct levels. The first level addresses all modeling aspects for scientific experiments.

The *ChorDesigner* is responsible for modeling, displaying, and adapting choreographies. It comprises several components each implementing a particular functionality. The *Choreography Modeling* component contains the actual drawing area where scientists can insert graphical shapes for participants, message links, and communication behavior such as send and receive from a modeling palette (SR6). With the help of this component scientists can model new scientific choreographies as well as adapt existing ones. The graphical notation is similar to the one used in the Workflow Modeling component to ease the understanding for scientists (SR12). Information about the running choreographed workflow instances will be interwoven with the graphical constructs (SR1). That means, different colors will display the current state of the specified communication behavior. This approach has already been used in [37] for workflow monitoring. The Choreography Modeling component is therefore the graphical user interface presented to the scientist that incorporates all other aspects of the system to be displayed. It is able to retrieve and reuse choreography fragments from the extended *Fragmento Library*, a library for workflow and choreography fragments. The *Choreography Monitor* is the component of the ChorDesigner that manages and prepares the information about

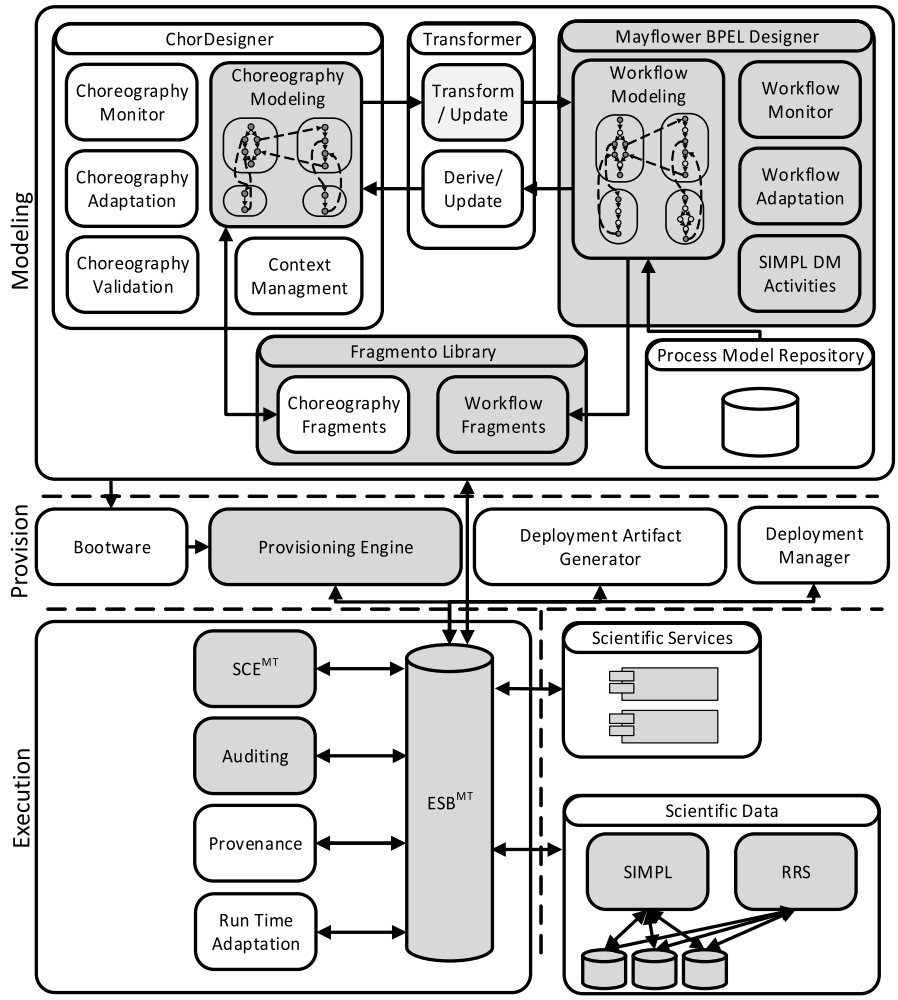


Fig. 5 Architecture of the scientific choreography system

the running choreographed instances for display in the Choreography Modeling component. The *Choreography Adaptation* component is responsible for providing all necessary choreography steering and adaptation facilities (SR4, SR6). The *Choreography Validation* component provides functions for checking the correctness of the adapted choreography (SR7). The responsibility of the *Context Management* component is the management of scientific choreography context such as shared input parameters for an experiment (SR14) as well as providing a grouping mechanism for the enacting workflow instances (SR13), which will have access to the shared choreography context. In [16], the concept of *Simulation Workflow Containers* holding context information for simulation workflows has been proposed. We plan to refine this concept

and provide an implementation through the Context Management component for scientific choreographies.

The *Transformer* component is an integral part of our modeling level architecture [46]. It is responsible for the transformation from choreography models to enacting abstract workflow models and vice versa. The *Transform/Update* sub-component provides functions for realizing the top-down modeling approach. The *Transform* function translates a scientific choreography into abstract workflows if the choreography model is transformed for the first time. Otherwise, the enacting workflow models (or instances if they are being executed) are updated without losing information (SR9). The second sub-component of the Transformer is the *Derive* component. It is responsible for providing functions for the bottom-up modeling approach and uses already existing workflow models or instances as input. The existing process models can be retrieved from the *Process Model Repository*. By analyzing the communication interconnection a meaningful choreography model is constructed. It mirrors the capabilities of the underlying workflows. With the help of the Transformer component a seamless blending between modeling and adaptation can be achieved without the scientist noticing it.

Workflow models and workflow instances are handled inside the *Mayflower BPEL Designer* already introduced in [37] and [39]. The *Workflow Modeling* component incorporates the functions for specifying workflow models on a drawing area and for monitoring and adapting a particular workflow instance during run time. However, the component must be extended to prevent the alteration of the communication constructs on the workflow level (SR5). Scientists can retrieve from and store workflow fragments in the *Fragmento Library* in order to ease modeling and foster reuse. The *Workflow Monitor* component is responsible for managing the information about each refined, running workflow instance that is then displayed in the Workflow Modeling component. The *Workflow Adaptation* component contains functions for workflow instance adaptation during run time as well as for versioning and instance migration of adapted workflow models. The *SIMPL DM Activities* component provides data management activities encapsulating complex data management functionality that can be inserted into the workflows. The SIMPL DM Activities are the part of the data-centric SIMPL framework [33] responsible for the graphical modeling of such activities. It supports uniform access to scientific data sources and provides an abstraction for scientists that hides low-level details of data management tasks. The framework provides workflow modeling constructs that can be used during modeling/adaptation time to easily integrate and configure data access.

The Provision phase is supported by several components. The *Deployment Artifact Generator* is responsible for packaging and grouping/distributing the choreographed workflow models. For this, it generates the necessary deployment descriptors. Furthermore, it also provides deployment information for the scientific services. The workflow models are deployed on our multi-tenant aware Service Composition Engine SCE^{MT} by the *Deployment Manager* component using the generated deployment information. The *Provisioning Engine*, e.g.,

OpenTOSCA [6], is responsible for provisioning middleware components such as the ESB^{MT} [42] and the SCE^{MT} [17] as well as the scientific services and data in an on-premise or off-premise environment and potentially on Cloud infrastructures. This can be done in an on-demand fashion when the scientist needs the execution environment [44]. The *Bootware* [44] is a component responsible for bootstrapping the Provisioning Engine.

The Execution phase is enabled by the components described next. *Scientific Services* implement the functions necessary for scientific experiments. Scientific Services can be complex software systems that expose their functionality via specified interfaces. We support interfaces specified with the Web Service Description Language (WSDL), REST-based interfaces [18], and command-line access via the SIMPL framework. The scientific services can be hosted on an application server, or on grid or cloud infrastructures. In our architecture the composite component *Scientific Data* contains all means to reach the data resources necessary for scientific experiments. The *SIMPL* component is part of the already mentioned SIMPL framework that deals with the actual access to data sources described by the data management activities on the modeling level. Data can be passed by reference directly between scientific services using the Reference Resolution System (RRS) [48] in order to avoid the transfer of huge amounts of data through the SCE^{MT} workflow engine. The communication backbone of our system is the *ESB^{MT}* [42]. The ESB^{MT} is a multi-tenant aware Enterprise Service Bus routing messages between all components. The *SCE^{MT}* [17] is our multi-tenant aware Service Composition Engine that is responsible for executing the enacting workflows. Execution may be started, even if the workflow models do not model the complete multi-* experiment (SR8). The *Auditing* component collects and stores all information generated during run time when navigating through the choreographed workflows as well as the information about the choreography model, workflow model, and instance adaptations (SR2, SR3). The *Provenance* component explicitly captures all information necessary to reproduce the results of a scientific experiment (SR11). Run time adaptations such as replacement of faulted services are the responsibility of the *Run Time Adaptation* component (SR10).

5.2 Implementation

Since some of the components of the system have been described in other publications of ours (cf. also Sec. 2), here we present details about the ones implemented most recently: the *ChorDesigner* and the *Transformer* component, which responsible for translating a BPEL4Chor choreography model to abstract BPEL workflow models in a top-down manner.

ChorDesigner: Although our approach to scientific choreographies does not rely on a particular choreography language, we have chosen to use BPEL4Chor because it fits most to our needs. BPEL4Chor [10] is a non-executable choreography language forming an additional layer on top of the BPEL standard [30].

Due to space constraints it is not possible to discuss the language artifacts in depth. The interested reader is referred to [10] and [11] for detailed information. In the *Participant Topology* the structural aspects of the choreography are specified. The Participant Topology contains the *Participant Declaration* which enumerates the involved choreography participants or participant sets. *Participant Sets* in BPEL4Chor are a grouping mechanism for choreography participants having the same type and whose number may be unknown during modeling time. *Message Links*, also part of the Participant Topology, indicate the connection between participants/participant sets on the level of the involved communication activities. The *Participant Behavior Descriptions* (PBD) contain the communication logic of each participant/participant set. The PBDs are realized by abstract BPEL processes following the abstract Process Profile for Participant Behavior Descriptions [10], i.e., BPEL Partner Links, Port Types and Operations are omitted. Technical information is decoupled from the logical definition of the choreography by the so-called *Participant Grounding*. This artifact specifies the concrete port type and operation of a logical message link.

The reason for choosing BPEL4Chor is that it fulfills the choreography language requirements CR1 - CR4 (Sec. 4). Regarding CR1, Decker et al. [11] evaluate BPEl4Chor and show that it covers most of the service interaction pattern in comparison with other choreography languages such as WS-CDL. Language requirement CR2 is fulfilled because BPEL4Chor follows the so-called *interconnected interface behavior model* approach. This approach avoids issues of *locally unenforceable* behavior that languages following the *interaction model* approach have. This is accomplished by explicitly specifying the control flow of every choreography participant. Deadlocks can be prevented by using the timeout mechanism of BPEL4Chor. Choreography language requirements CR3 and CR4 are fulfilled through the inherent concepts of BPEL4Chor. Fault Handling (CR3) can be accomplished by specifying fault handlers in the Participant Behavior Descriptions, i.e., in the abstract BPEL processes. However, to cope with choreography-wide fault handling, additional mechanisms have to be introduced to BPEL4Chor in future. Requirement CR4 – the compatibility to conventional workflow technologies from the business domain – is ensured by transforming the BPEL4Chor description to standardized, abstract BPEL. Adaptation mechanisms that allow a refinement of communication activities during run time of an multi-* experiment (CR5) are not yet part of BPEL4Chor. Our future work will concentrate on this.

The ChorDesigner is an Eclipse-based choreography editor with its own metamodel, but allowing the import and export of BPEL4Chor artifacts. Fig. 6 uses the coupled multi-scale simulation example from Sec. 2.3 and shows the modeling with our ChorDesigner. On the left hand side on the canvas, the kinetic Monte Carlo simulation participant has been modeled. The participant also includes the actual simulation steps modeled as opaque (unspecified) activities to show the complete simulation, i.e, it is indicated that these steps have to be conducted in a particular order but it is not specified how. Scientists can either omit these activities in the ChorDesigner and add them after trans-

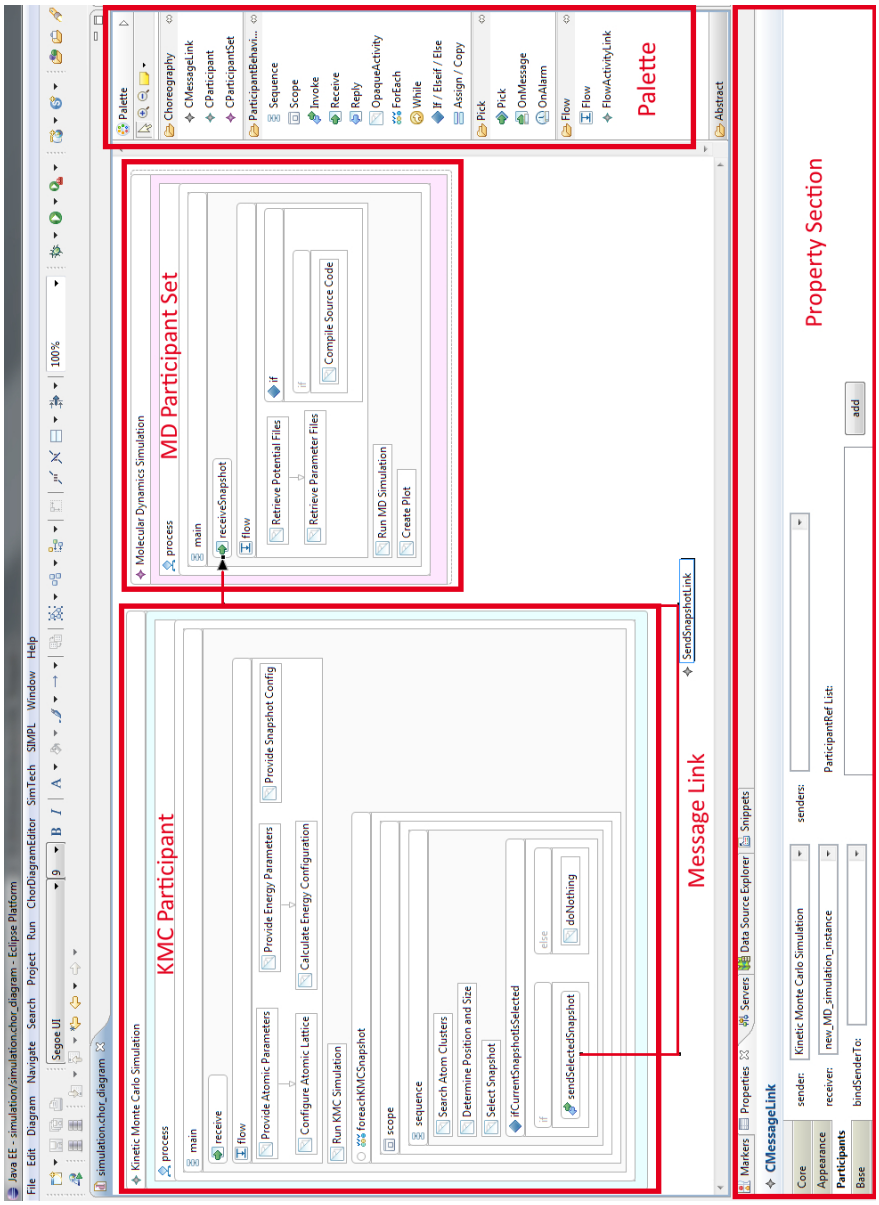


Fig. 6 Motivating scenario modeled with the ChorDesigner

forming the choreography model into the enacting workflows or turn them into concrete activities after the transformation. The right hand side figure represents the Molecular Dynamics simulation participant. It forms a *Participant Set* because it gets instantiated more than once during execution depending on the number of snapshots that is selected in the KMC simulation. The con-

nection between both simulations is modeled with the *SendSnapshotLink*. A snapshot file is either transfered directly or a reference to the file is sent. The modeling palette is visible on the right side containing shapes for the choreography participants and the activities to be modeled. They can be placed on the canvas via drag and drop. The ChorDesigner validates the elements placement directly and prohibits for example the placement of communication activities outside of a participant figure. On the bottom of Fig. 6, the property section for the message link is displayed. Here the sender and the receiver of the link are configured. Each modeling element has such a property section allowing to configure mandatory and optional properties.

Transformer: The Transformer component has originally been conceptually specified in [32]. Our implementation picks up the general idea from this concept, which is collecting all necessary information from the choreography artifacts and then using the collected information in the transformation. However, we specified and implemented our own algorithm to overcome the deficiencies of previous implementations [46]. A grounded BPEL4Chor choreography model contains the main input artifacts: a Participant Topology, a set of Participant Behavior Descriptions, and a Participant Grounding. Optionally, existing WSDL definitions can also be used as additional input for the transformation. These WSDL files describe already existing port types, operations, and messages of particular participants. If none exist, the WSDL files will be generated using the information specified in the participant grounding. During the transformation, for each participant/participant set and the corresponding Participant Behavior Description, an abstract BPEL process is generated. The process model contains exactly the communication constructs and their conditional ordering specified on the drawing area inside a participant. The Message Links stored in the Participant Topology are transformed into BPEL Partner Link Types and Partner Links associated to the corresponding BPEL process models. The technical information of the Participant Grounding is incorporated into newly generated or existing WSDL files. The output of the transformation is a set of self-contained *Process Bundles*. Each Process Bundle consists of an abstract BPEL Process and a corresponding WDSL definition denoted by *Process WSDL* specifying the interface of the process. Furthermore, the bundle contains a set of *Partner WSDL* definitions representing the choreography participants the abstract BPEL process communicates with.

A high-level view on the transformation can be seen in Algorithm 1 and Algorithm 2. Due to space limitations only the most noteworthy steps are explicitly explained in the following. Both algorithms omit some details such as the handling of BPEL scopes, BPEL4Chor Participant Sets and the checking if elements already exist in a set before adding it and show only the most important steps for brevity. Furthermore, it does not show the update of existing models or instances. The central concept of the transformation is the creation of a data structure denoted by *CommunicatingParticipantsData* that contains all information about one Message Link between two participants. Algorithm 1 shows a *Preparation* algorithm for building a set of Communicat-

Algorithm 1: Preparation algorithm

```
1  input  : Topology T = (PL, ML), a set of PBD, Grounding G
2  output: A set of CommunicatingParticipantsData CP where
           cp ∈ CP = (p_s, a_s, p_r, a_r, ml, ml_grounded, syncResponse)
3  begin
4      Set of abstract BPEL Processes P ⟵ ∅
5      Set of CommunicatingParticipantsData CP ⟵ ∅
6      Set of maps of Participants to abstract BPEL processes M ⟵ ∅
7      foreach Participant Behavior Description pbd ∈ PBD do
8          Process p ⟵ changeToAbstractProcess(pbd)     // Turn PBD to process
9          P ⟵ P ∪ p
10         Map m ⟵ mapParticipantToProcess(PL,p)
11         M ⟵ M ∪ m
12     end
13     foreach MessageLink ml ∈ ML do
14         ml ↦ grounded Message Link ml_grounded
15         Process p_s ⟵ p ∈ M such that participant_ml ∈ ml = participant_m ∈ M
           ∧participant_ml is sending participant          // Assign send. process
16         Process p_r ⟵ p ∈ M such that participant_ml ∈ ml = participant_m ∈ M
           ∧participant_ml is receiving participant        // Assign rec. process
17         String activityId_s ⟵ sender attribute s ∈ ml   // Set send. activity id
18         String activityId_r ⟵ receiver attribute r ∈ ml  // Set rec. activity id
19         CommunicationActivity a_s ⟵ findActivity(p_s,activityId_s)
20         CommunicationActivity a_r ⟵ findActivity(p_r,activityId_r)
21         boolean syncResponse ⟵ isSychronousResponse(ML,ml)
22         CommunicatingParticipantsData
           cp ⟵ createCP(p_s,a_s,p_r,a_r,ml,ml_grounded,syncResponse)
23         CP ⟵ CP ∪ cp
24     end
25     return CP
26 end
```

ingParticipantsData taking a grounded BPEL4Chor choreography as input. The input topology $T = (PL, ML)$ contains a set of participants PL and a set of Message Links ML. A CommunicatingParticipantsData cp is a tuple $cp = (p_\mathrm{s}, a_\mathrm{s}, p_\mathrm{r}, a_\mathrm{r}, ml, ml_\mathrm{grounded}, syncResponse)$ where p_s is the sending and p_r the receiving abstract BPEL process of the communication relationship represented by cp. $\mathrm{a_s}$ and $\mathrm{a_r}$ are the sending and receiving communication activities connected by the Message Link ml. ml_grounded provides the technical information for each Message Link ml. *syncResponse* is a boolean value indicating if a particular CommunicatingParticipantsData realizes a synchronous response to a previous request.

In the lines 7-11 of Algorithm 1, the Participant Behavior Descriptions are prepared for the transformation. First, every PBD is turned into an abstract BPEL process by simply removing the attribute *abstractProcessProfile* referencing the profile *Abstract Process Profile for Participant Behavior Descriptions* of BPEL4Chor in the function *changeToAbstractProcess*. This allows to add elements such as Partner Links which were previously forbidden accord-

Algorithm 2: High level transformation algorithm

```
1  input  : Topology T = (PL, ML), a set of Part. Behav. Descr. PBD, Grounding G
2  output: A set of process bundles B where b ∈ B = (p, wsdl_process, WSDL_partners)
3  begin
4      Set of Process Bundles B ⟵ ∅
5      Set of CommunicatingParticipantsData CP ⟵ prepare(T,PBD,G)
6      foreach CommunicatingParticipantsData cp ∈ CP do
7          WSDL wsdl^s_process, wsdl^r_process          // Init. sending/receiving WSDLs
8          Set of WSDL WSDL^s_partners, WSDL^r_partners ⟵ ∅  // Init. partner sets
9          Process p_s ⟵ p_s ∈ cp            // Assign to sending process variable
10         Process p_r ⟵ p_r ∈ cp          // Assign to receiving process variable
11         if B ∋ Process Bundle b for p_s then
12             Process Bundle b_s ⟵ b ∈ B              // Assign existing bundle
13         else
14             Process Bundle b_s ⟵ new Process Bundle      // Create new bundle
15             p_s ↦ b_s                    // Map sending process to sending bundle
16             wsdl^s_process ↦ b_s     // Map sending process WSDL to sending bundle
17             WSDL^s_partners ↦ b_s         // Map partner WSDLs to sending bundle
18             B ⟵ B ∪ b_s                // Add sending bundle to set of bundles
19         end
20         if B ∋ Process Bundle b for p_r then
21             ...         // Handling of the receiving process bundle is similar
22         end
23         Partner Link Pair PLP ⟵ buildPartnerLinks(cp)
24         Partner Link pl_s ∈ PLP ↦ p_s      // Map partner link to send. process
25         Partner Link pl_r ∈ PLP ↦ p_r       // Map partner link to rec. process
26         WSDL Pair WP ⟵ buildWSDLs(cp,PLP)
27         WSDL wsdl_s ∈ WP ↦ wsdl^s_process        // Map to sending process WSDL
28         WSDL wsdl_r ∈ WP ↦ wsdl^r_process     // Map to receiving process WSDL
29         if cp is synchronous response then
30             importWSDL(p_s,wsdl^s_process)   // import send. WSDL in send. process
31             importWSDL(p_r,wsdl^r_process)   // import recei. WSDL in rec. process
32             WSDL^r_partners ⟵ WSDL^r_partners ∪ wsdl^s_process
33         else
34             importWSDL(p_s,wsdl^s_process)   // import send. WSDL in send. process
35             importWSDL(p_s,wsdl^r_process)    // import send. WSDL in rec. process
36             importWSDL(p_r,wsdl^r_process)   // import recei. WSDL in rec. process
37             WSDL^s_partners ⟵ WSDL^s_partners ∪ wsdl^r_process
38             WSDL^r_partners ⟵ WSDL^r_partners ∪ wsdl^s_process
39         end
40         Variable Pair VP ⟵ buildVariables(cp)        // Build variable pair
41         modifyCommunicationActivity(pl_s,v_s ∈ VP,a_s ∈cp)
42         modifyCommunicationActivity(pl_r,v_r ∈ VP,a_r ∈cp)
43     end
44     return B
45 end
```

ing to the profile. The corresponding participant from the set of Participants $PL \in T$ is mapped to the current process p using the function *mapParticipantToProcess*. The resulting map m is added to the set of maps M. The mapping simplifies the identification of processes by using their corresponding participants in the following. Subsequently, we iterate over each Message Link $ml \in ML$. In line 14 the Message Link ml is mapped to a grounded Message

Link ml_{grounded}. Line 15 contains the assignment of the sending process to the variable p_{s} by retrieving the process in the set of maps M which has the same participant as the Message Link ml in its sender attribute. Line 16 conducts the assignment for the receiving process. Using the function *isSychronousResponse*, it is determined if a Message Link ml is a synchronous response to a request (line 21).

Algorithm 2 depicts the high-level *Transformation* algorithm. The input consists of the artifacts of a grounded BPEL4Chor choreography. Output is a set of Process Bundles $B = (p, wsdl_{\text{process}}, WSDL_{\text{partners}})$ where p is an abstract BPEL process, $wsdl_{\text{process}}$ is the process WSDL definition of the abstract process and W_{partners} are the set of WSDL definitions of the partner processes, which communicate with the abstract process p. The main idea is the pairwise generation of BPEL process elements, e.g., Partner Link and WSDL pairs. The handling of the WSDL definitions in Algorithm 2 only considers newly generated ones and not existing ones. The generation (*buildWSDLs* function) internally uses the current Partner Link Pair PLP for the building of BPEL Partner Link Types and results in a *WSDL Pair* output (lines 26-28). The generation of the WSDL imports in the processes is conducted differently depending on whether the current CommunicatingParticipantsData data structure represents a synchronous response or not. If it is a synchronous response (lines 29-32), the *importWSDL* function is executed only once for the sending process p_{s} and once for the receiving process p_{r} with the corresponding process WSDL definitions. Moreover, the sending process WSDL definition $wsdl^{\text{s}}_{\text{process}}$ is only added to the set of partner WSDL definitions $WSDL^{\text{r}}_{\text{partners}}$ of the receiving side. If the current CommunicatingParticipantsData does not represent a synchronous response (lines 34-38), the WSDL definition of the receiving Participant $wsdl^{\text{r}}_{\text{process}}$ is also added to the sending process p_{s}. In case of the synchronous response this is not necessary because the corresponding WSDL file will be imported in any case when the CommunicatingParticipantsData is processed, which represents the synchronous request.

6 Related Work

In this section we compare our approach to existing ones documented in literature. Scherp and Hasselbring [35] propose an model-driven approach with two levels of abstractions for scientific workflows. On the domain-specific level scientists model data and control flow with BPMN in order to hide the technical complexity of executable workflow languages. The resulting model is transformed into an executable control flow oriented workflow language. While our work also uses conventional workflow technology from the business domain, we aim at enabling scientists to create and conduct coupled multi-* experiments using choreographed workflows. Moreover, we also consider the blending of modeling and adaptation phases of choreographies and workflows.

In [19] a scientific workflow system for molecular biology MoBiFlow is presented. The system's functionality has been derived from a set of biology-

specific and technical requirements and uses a Web 2.0 based graphical user interface that abstracts from the underlying BPEL workflow language. Although the collaborative aspect of scientific workflow modeling is the main focus, the interconnection of software systems from different scientific fields in one experiment are not considered. Furthermore, the system does not incorporate flexibility aspects that allow adapting workflow instances during runtime to facilitate a trial-and-error approach for the scientists.

Fleuren et al. [12] propose an approach and an implementation for integrating control and data flow by combining orchestration and choreography. The main workflow is modeled as an orchestration using a control flow perspective also considering fault handling and compensation. Data flow is integrated into the main workflow as sub-workflows denoted by workflow skeletons. Inside the workflow skeletons choreographed proxies represent workflow tasks such as service calls or job executions and are responsible for parallel data handling. Data is exchanged directly between proxies to avoid the central orchestration engine becoming a bottleneck. In contrast, we do not only model data flow as choreographies but control flow as well and also consider bottom-up derivation.

In Taverna, scientific workflows model the data flow between a set of activities which represent processing nodes [26]. The system uses a recursive model for activities, i.e., activities can also represent sub-workflows. Some control flow constructs such as loops have become first class citizens in the mean time and must not be simulated with the help of sub-workflows any more. The nesting capability of Taverna would also allow modeling of multi-scale problems in the biology domain and even multi-field problems if groups of scientists from different domains collaboratively model a parent Taverna workflow and corresponding sub-workflows. Similarly to Taverna, in the Kepler system [24] activities, denoted by Actors, represent either atomic tasks or nested sub-workflows. In comparison our approach does not aim for the nesting of sub-workflows to model multi-* problems where one parent workflow forms the central orchestration. Instead, we propose the use of choreographies of scientific workflows without a centralized coordinator. This has the advantage that the scientific workflows for particular scales and fields are more decoupled and have no transactional dependencies on a parent workflow. Fault handling and compensation logic is defined in each enacting workflow separately. The communication patterns between the workflows are not restricted to request/response as in the case of sub-workflows and we support the adaptation of the choreography and the enacting workflows during run time.

Freire et al. present the features of the VisTrails sWfMS in [14]. The Python-based VisTrails system is inherently built around the concept of provenance. Workflow executions and produced data are tracked even if file names change. Intermediate results are cached in order to provide check-points for long-running computations and for the reuse of results when exploratively designing a scientific experiment. Further Python-compatible packages can be easily integrated into VisTrails, e.g, the ALPS package for simulating strongly correlated quantum lattice models [2]. Furthermore, VisTrails supports provenance tracking over implicitly coupled workflows [21]. These workflows are tied

together via their output and inputs but without being specified by a global model. In contrast, we use the notion of choreographies to provide a global model of a multi-* simulation as starting point for explorative modeling and adaptation. Furthermore, we support the derivation of a choreography model from existing and implicitly coupled workflows.

Plankensteiner et al. propose the Interoperable Workflow Intermediate Representation (IWIR) to integrate different scientific Workflow Management systems [31]. The systems ASKALON, MOTEUR, WS-PGRADE, and Triana and their corresponding scientific workflow languages are used as case studies. The distinct languages are first translated to the IWIR as intermediate language and then to the desired target language. The authors discuss collaboration scenarios where modeling can be started in one language and continued in another after translation. Our approach also aims for the collaboration of scientists, however, we do not use an intermediate language for coupling different workflow languages and enable the translation of workflow models. Instead, we model a multi-* experiment as a choreography of scientific workflows and explicitly consider adaptation of the coupled workflows during run time.

In [34], the authors present an approach and implementation for a scalable, distributed execution of so-called meta-workflows consisting of multi-language sub-workflows. A bundling format (SHIWA) for representing workflows and corresponding input/output data is used for enabling the communication between scientific workflow engines during run time. Workflow engines, in this case Triana and Monteur, subscribe to a centralized execution pool to retrieve self-contained SHIWA bundles with workflow models in their respective language. After execution of the retrieved bundle, the workflow engine inserts a new bundle with the results to the pool. Again, the main difference to our work is that we do not treat the coupling of (multi-*) workflows as the invocation of several sub-workflows from a meta-workflow but as collaborating workflows forming a decentralized choreography. Moreover, we aim for trial-and-error modeling support for coupled experiments.

In summary, the comparison with related work shows several differences to our work: (i) we use a standard-based approach which aims for a generic applicability in all scientific domains whereas existing work is often triggered by individual requirements by one or a few scientific fields. Furthermore, we extend control-flow oriented technologies with capabilities to better handle data flow while related approaches start from data flow orientation which they enrich with control flow constructs. (ii) We use choreographies, which provide loosely coupled collaboration between scientific fields and scales plus complex interaction patterns between the enacting workflows instead of more tightly coupled sub-workflows dependent on the life cycle of a parent workflow. (iii) We consider not only the top-down modeling of a scientific experiment from scratch but also bottom-up derivation of a global choreography model from existing scientific workflows. (iv) Our approach enables adaptation on both the workflow and choreography level after an experiment has already been started to support modeling in an trial-and-error manner, while existing approaches are often static and do not support run time adaptation.

7 Conclusions and Future Work

In answer to the need for support of multi-* simulations in this paper we introduced an approach that provides scientists with the means to model such simulations in a user-friendly and trial-and-error manner and execute them automatically. The approach is based on existing technologies known from business applications and extensions in terms of concepts and implementation that we created in order to address issues typical for scientific simulations, like long duration and data and computational intensive nature. We improve the user friendliness of the system by providing scientists with means to easily model, instead of program, their simulations. Moreover, unlike typical business workflows, our approach allows modeling of the simulation even after its execution has started (i.e., in support of the trial-and-error way simulations are typically created). Existing systems do not provide comprehensive support for collaborative development and execution of complex, multi-scale and/or multi-field simulations involving software from different research or industry organizations. To address this need, we borrow the notion of choreographies and extend and apply it for modeling and execution of this kind of simulations.

A comprehensive evaluation of the approach and the supporting system with other scientists participating in SimTech and beyond as well as the application of the top-down and bottom-up modeling to other scenarios will be done in future. Currently, our results are being evaluated in the scope of the motivating material science application scenario by the involved natural scientists whose requirements directly influenced our approach.

In our future research we plan to enhance our work with techniques for modeling complex correlation dependencies among the participant simulations in a choreography, enable the definition of a common context, improve reusability of choreographies by means of choreography fragments or templates, and by using generic configurable connectors between choreography participants as well as configurable workflows. With respect to the dynamic provisioning and de-provisioning of choreographed simulations we will focus on the optimization of the distribution of the simulations or parts of them as well as the placement of individual simulation (Web or REST) services across a distributed infrastructure. Provenance of scientific workflows and simulations is of major importance in eScience and this is also part of our future research plans.

Acknowledgements 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

1. Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., Weiß, A.: Collaborative, Dynamic & Complex Systems: Modeling, Provision & Execution. In: CLOSER'14, pp. 0–10 (2014)

2. B. Bauer et al.: The ALPS project release 2.0: open source software for strongly correlated systems. J. of Statistical Mechanics: Theory and Experiment **2011**(05), P05,001 (2011)
3. Barga, R., Gannon, D.: Scientific versus Business Workflows. In: Workflows for e-Science, pp. 9–16. Springer (2007)
4. Barros, A., Dumas, M., Hofstede, A.H.M.T.: Service interaction patterns. In: BPM'05, pp. 302–318. Springer (2005)
5. Binkele, P.: Atomistische Modellierung und Computersimulation der Ostwald-Reifung von Ausscheidungen beim Einsatz von kupferhaltigen Stählen. Ph.D. thesis, University of Stuttgart, Stuttgart (2006)
6. Binz, T., Breitenbücher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A., Wagner, S.: OpenTOSCA - A Runtime for TOSCA-based Cloud Applications. In: ICSOC'13, pp. 694–697. Springer (2013)
7. Bucchiarone, A., Marconi, A., Pistore, M., Raik, H.: Dynamic Adaptation of Fragment-Based and Context-Aware Business Processes. In: ICWS'12, pp. 33–41 (2012)
8. Davidson, S.B., Freire, J.: Provenance and Scientific Workflows: Challenges and Opportunities. In: ACM SIGMOD Management of Data'08, pp. 1345–1350 (2008)
9. Decker, G., Kopp, O., Barros, A.: An Introduction to Service Choreographies. Information Technology **50**(2), 122–127 (2008). DOI 10.1524/itit.2008.0473
10. Decker, G., Kopp, O., Leymann, F., Weske, M.: BPEL4Chor: Extending BPEL for Modeling Choreographies. In: ICWS'07. IEEE (2007)
11. Decker, G., Kopp, O., Leymann, F., Weske, M.: Interacting services: from specification to execution. Data & Knowledge Engineering **68**(10), 946–972 (2009)
12. Fleuren, T., Götze, J., Müller, P.: Workflow Skeletons: Increasing Scalability of Scientific Workflows by Combining Orchestration and Choreography. In: ECOWS'11, pp. 99–106. IEEE (2011)
13. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a new computing infrastructure. Elsevier (2003)
14. Freire, J., Koop, D., Santos, E., Carlos Scheidegger, C.S., Vo, H.T.: VisTrails. The Architecture of Open Source Applications. Lulu (2011). URL http://aosabook.org/en/vistrails.html
15. Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. Computer **40**(12), 24–32 (2007)
16. Görlach, K., Sonntag, M., Karastoyanova, D., Leymann, F., Reiter, M.: Conventional Workflow Technology for Scientific Simulation, pp. 323–352. Springer (2011)
17. Hahn, M.: Approach and Realization of a Multi-tenant Service Composition Engine. Diploma Thesis No. 3546, University of Stuttgart, Germany (2013)
18. Haupt, F., Fischer, M., Karastoyanova, D., Leymann, F., Vukojevic-Haupt, K.: Service Composition for REST. In: EDOC'14. IEEE (2014)
19. Held, M., Küchlin, W., Blochinger, W.: Mobiflow: Principles and design of a workflow system for molecular biology. IJSSMET (4), 67–78 (2011)
20. Hey, T., Tansley, S., Tolle, K. (eds.): The fourth paradigm: data-intensive scientific discovery. Microsoft Research (2009)
21. Koop, D., Santos, E., Bauer, B., Troyer, M., Freire, J., Silva, C.T.: Bridging Workflow and Data Provenance Using Strong Links. In: SSDBM'10, pp. 397–415. Springer (2010)
22. Krause, R., Markert, B., Ehlers, W.: A Porous Media Model for the Description of Adaptive Bone Remodelling. PAMM **10**(1), 79–80 (2010)
23. Leymann, F., Roller, D.: Production Workflows. Prentice Hall (1999)
24. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E.A., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System: Research Articles. Concurr. Comput. : Pract. Exper. **18**(10), 1039–1065 (2006)
25. M. Sonntag and D. Karastoyanova: Next Generation Interactive Scientific Experimenting Based On The Workflow Technology. In: MS'10. ACTA Press (2010)
26. Missier, P., Soiland-Reyes, S., Owen, S., Tan, W., Nenadic, A., Dunlop, I., Williams, A., Oinn, T., Goble, C.: Taverna, Reloaded. In: SSDBM'10, pp. 471–481. Springer (2010)
27. Molnar, D., Binkele, P., Hocker, S., Schmauder, S.: 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)

28. Molnar, D., Binkele, P., Mora, A., Mukherjee, R., Nestler, B., Schmauder, S.: Molecular dynamics virtual testing of thermally aged Fe-Cu microstructures obtained from multiscale simulations. Computational Materials Science **81**(0), 466 – 470 (2014)
29. Molnar, D., Mukherjee, R., Choudhury, A., Mora, A., Binkele, P., Selzer, M., Nestler, B., Schmauder, S.: 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)
30. OASIS: Web Services Business Process Execution Language Version 2.0 (2007). URL http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html
31. Plankensteiner, K., Prodan, R., Janetschek, M., Fahringer, T., Montagnat, J., Rogers, D., Harvey, I., Taylor, I., Balask, A., Kacsuk, P.: Fine-Grain Interoperability of Scientific Workflows in Distributed Computing Infrastructures. Grid Comp. **11**(3), 429–455 (2013)
32. Reimann, P.: Generating BPEL Processes from a BPEL4Chor Description. Student Thesis No. 2100, University of Stuttgart (2007)
33. Reimann, P., Reiter, M., Schwarz, H., Karastoyanova, D., Leymann, F.: SIMPL - A Framework for Accessing External Data in Simulation Workflows. In: BTW'11, pp. 534–553 (2011)
34. Rogers, D., Harvey, I., Huu, T.T., Evans, K., Glatard, T., Kallel, I., Taylor, I., Montagnat, J., Jones, A., Harrison, A.: Bundle and Pool Architecture for Multi-Language, Robust, Scalable Workflow Executions. J. of Grid Computing **11**(3), 457–480 (2013)
35. Scherp, G., Hasselbring, W.: Towards a model-driven transformation framework for scientific workflows. Procedia Computer Science **1**(1), 1519 – 1526 (2010)
36. Schumm, D., Karastoyanova, D., Leymann, F., Strauch, S.: Fragmento: Advanced Process Fragment Library. In: ISD'10. Springer (2010)
37. Sonntag, M., Hahn, M., Karastoyanova, D.: Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In: CEUR Workshop'12, pp. 1–5. Springer (2012)
38. Sonntag, M., Hotta, S., Karastoyanova, D., Molnar, D., Schmauder, S.: Using Services and Service Compositions to Enable the Distributed Execution of Legacy Simulation Applications. In: ServiceWave'11, pp. 1–12. Springer (2011)
39. Sonntag, M., Karastoyanova, D.: Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. J. of Grid Computing **11**(3), 553–583 (2013)
40. Sonntag, M., Karastoyanova, D., Leymann, F.: The Missing Features of Workflow Systems for Scientific Computations. In: GWW'10, pp. 209–216. GI (2010)
41. Stadler, J., Mikulla, R., Trebin, H.R.: IMD: A software package for molecular dynamics studies on parallel computers. Int. J. of Modern Physics C **8**(5), 1131–1140 (1997)
42. Strauch, S., Andrikopoulos, V., Leymann, F., Muhler, D.: ESB^{MT}: Enabling Multi-Tenancy in Enterprise Service Buses. In: CloudCom'12, pp. 456–463. IEEE (2012)
43. Taylor, I., Shields, M., Wang, I., Harrison, A.: The Triana Workflow Environment: Architecture and Applications. In: I. Taylor, E. Deelman, D. Gannon, M. Shields (eds.) Workflows for e-Science, pp. 320–339. Springer (2007)
44. Vukojevic-Haupt, K., Karastoyanova, D., Leymann, F.: On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows. In: SOCA'13, pp. 1–8 (2013)
45. Weerawarana, S., Curbera, F., Leymann, F., Storey, T., Ferguson, D.F.: Web Services Platform Architecture. Prentice Hall (2005)
46. Weiß, A., Andrikopoulos, V., Gómez Sáez, S., Karastoyanova, D., Vukojevic-Haupt, K.: Modeling Choreographies using the BPEL4Chor Designer: an Evaluation Based on Case Studies. Technical Report 2013/03, University of Stuttgart, Germany (2013)
47. Weiß, A., Karastoyanova, D., Molnar, D., Schmauder, S.: Coupling of Existing Simulations using Bottom-up Modeling of Choreographies. In: Workshop on Simulation Technology: Systems for Data Intensive Simulations (SimTech@GI) in Conjunction with INFORMATIK 2014, pp. 101–112. Gesellschaft für Informatik e.V. (GI) (2014)
48. Wieland, M., Görlach, K., Schumm, D., Leymann, F.: Towards Reference Passing in Web Service and Workflow-based Applications. In: EDOC'09, pp. 109–118. IEEE (2009)
49. Zaha, J., Dumas, M., ter Hofstede, A., Barros, A., Decker, G.: Bridging global and local models of service-oriented systems. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on **38**(3), 302–318 (2008)

All links were last followed on October 11, 2014.