Institute of Architecture of Application Systems

A Life Cycle for Coupled Multi-Scale, Multi-Field Experiments Realized through Choreographies

Andreas Weiß, Dimka Karastoyanova

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

BibTeX:

```
@inproceedings   {INPROC-2014-39,
    author =     {Andreas Wei{\ss} and Dimka Karastoyanova},
    title =      {{A Life Cycle for Coupled Multi-Scale, Multi-Field
                 Experiments Realized through Choreographies}},
    booktitle =  {Proceedings of the 18th IEEE International EDOC
                 Conference (EDOC 2014)},
    publisher =  {IEEE Computer Society},
    pages =      {1--8},
    month =      {September},
    year =      {2014}
}
```


Universität Stuttgart
Germany

A Life Cycle for Coupled Multi-Scale, Multi-Field Experiments Realized through Choreographies

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

***Abstract*—Current systems for enacting scientific experiments, and in particular simulation workflows, do not support multi-scale and multi-field problems if they are not coupled on the level of the mathematical model. We present a life cycle that utilizes the notion of choreographies to enable the trial-and-error modeling and execution of multi-scale and/or multi-field simulations. The life cycle exhibits two views reflecting the characteristics of modeling and execution in a top-down and bottom-up manner. It defines techniques for composing data-intensive, scientific workflows in more complex simulations in a generic, domain-independent way, and thus provides scientists with means for collaborative and integrated data management based on the workflow paradigm.**

I. Introduction

eScience is a very active research field and its main objective is to provide generic approaches and tools to support its whole life cycle [1] and different fields of natural and social sciences for the purpose of faster scientific exploration and discovery. One approach for data processing and analysis, which is the third phase of eScience, is the workflow technology, also known as scientific workflows. It is used for enabling scientific simulations. Note that scientific workflows have different characteristics than 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 their requirements in one (or just a few) scientific domains. In most cases, scientific workflows support modeling of a workflow of data management or computing tasks and focus on how data is processed, in what sequence, and which data sources are to be used. Generally, these systems hide the complexity of distribution and parallelization of computational tasks from scientists, however, sometimes force them to use one specific technology. Usually such systems derive an optimized workflow model which can be executed multiple times. However, fault and exception handling are often 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 considered. Workflows are executed without interruptions and often do not allow human users to interact with it.

The second category of workflow systems is a more recent development and makes use of the conventional workflow technology known from BPM. The advantages of business workflows such as 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, 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 has to be considered.

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 physical model, i.e., a single field model describing the scientific phenomenon. The more complex multi-scale and/or multi-field[1] 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 [2]. 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 [3]. With this approach, multiple time scales of thermal aging as well as length scales in

[1]In the following we will use the term multi-* to abbreviate multi-scale and/or multi-field and explicitly state it when we refer to one particular characteristic.

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.

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 fields of the natural phenomena also need to be supported. Especially important is the case where collaboration among scientific groups or industry organizations, each of them having distinct expertise, is desired. This implies coupling of existing simulation software into complex simulations and the correlation of the interactions among them. Towards this goal, in this paper we present an approach that utilizes the notion of choreographies to enable the trial-and-error modeling and execution of multi-* simulations. We contribute a definition of the life cycle of such simulations and present in detail concepts and techniques that support all life cycle phases. The main objective has been to reuse as much of existing standards, techniques and mechanisms as possible, while providing a user-friendly system to scientists, who are both the developers of the simulations and their users. 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 structure the paper in the following way: Sec. II gives more details on our motivation for the presented approach and discusses the available system for scientific workflows, which serves as a basis for our choreographed, multi-* scientific simulations. We introduce our approach and supporting life cycle in Sec. III. We compare our approach with related ones in Sec. IV and conclude the article with an outline of future research topics in Sec. V.

II. Background and Motivation

In this section we present the requirements of eScience as introduced in our previous research work and will derive additional requirements for the approach 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[2]) 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 [4], [5], [6], [7], [1], 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 [8] 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 [9] 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 their precise conditional ordering, i.e., 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 [10]. 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 [11]. 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 [12]. They can also be used to capture both templates for data management and workflow 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

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

the workflow as a service, and the subsequent instantiation of a workflow instance – all of 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 scientist 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 [13], [10]. 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 allow for 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 major open issue with respect to the modeling is how the interactions among the participating simulations and the data exchange can be represented.

III. Life Cycle for Multi-Scale and Multi-Field Experiments

As shown in Sec. II, previous work has enabled scientist to model and execute scientific workflows that orchestrate scientific services coping with either one single field and one single scale or with services combining separate scientific fields and different scales into one coherent scientific model. As a logical continuation, it is our goal to enable scientist to model and execute scientific multi-* experiments in an easy and user-friendly manner for the cases not supported by existing work. Again, our approach focuses on using proven methods and standard-based technologies of the business domain [14]. In this work, we use the concept of 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. Choreographies provide a global view on the interconnection of independent organizations communicating without a central coordinator [15], [16]. 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 distinct organizations (and their workflows) 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. This enables the modeling and execution in a distributed manner utilizing the expertise of different scientist form different domains. Note that for integration purposes we assume that experiments are available as services, as discussed for example in [17].

In the following, we introduce a life cycle for multi-* experiments that are realized by choreographies and address the identified need for both top-down and bottom-up modeling and execution. Both modeling approaches must enable the typical trial-and-error modeling style of scientists [18], [14]. Since scientists want to be able react to intermediate results during execution without modeling the experiment completely beforehand, the enactment of choreographies, i.e., the execution of the collaborating workflows, may be started even before the choreography model is completely specified. An adaptation follows afterwards, for example to add new experiment methods on a different scale. We introduce the notion of *Model-as-you-go for choreographies* to reflect this fact. The proposed life cycle is an extension of the scientific workflow life cycle introduced in [14], which itself has been

implemented by an extended BPM life cycle. While in the traditional BPM life cycle there are 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. 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, execution, and monitoring are perceived as one experimentation process by scientists. Both the top-down and the bottom-up approaches enable modeling of multi-* experiments in a round-trip fashion.

A. Top-down life cycle

Fig. 1 shows the extended life cycle for the top-down modeling and execution approach as it is experienced by the scientist. The dashed arrows point to the artifacts visible to the scientist as input or output of a particular life cycle phase. Scientists start from a scientific multi-scale and/or multi-field problem (1) and model the experiments in the distinct scientific fields as participants of a choreography in the *Choreography Modeling* phase. 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 from choreography languages. In each participant, only the activities, control and data flow constructs necessary for the communication with other choreography participants are modeled (i.e., its public communication interface). The orchestration logic of each single-field and single-scale experiment is not explicitly modeled in this phase. Typically, choreography modeling languages are not executable [15]. Therefore, the scientific choreography is transformed into an abstract representation of an executable workflow language. For this, separate abstract workflow models are generated for each participant containing only the communication constructs such as send and receive activities (3). If the scientist has already built a (incomplete) choreography model previously, the transformation action has to be aware of this and existing communication constructs must be updated accordingly. Furthermore, in this case refined workflow logic and already running experiment state must not be lost. The transformation is kept transparent for scientists and is hidden behind the by *Transform/Update* action. Scientists can then use an orchestration editor to conduct a manual refinement of the generated workflow definitions. In the *Workflow Modeling* phase, the internal orchestration logic for every distinct single-scale and single-field experiment participating in the overall choreography is added. Scientists add activities and activity implementations to the workflow thus making it executable (4). It must be possible for the scientists to model a particular workflow only partially or with abstract placeholders, i.e, the only partially modeled workflows must still be executable. However, this may lead to situations where the overall choreography can not yet be executed since 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. It is possible that different scientists 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. Scientists may switch back to the Choreography Modeling phase using the *Return* action if necessary. The *Run/Resume* action hides all technical details of deployment and instantiation of the updated workflows. The next phase is the *Execution and Monitoring* phase where the scientist monitors the running choreographed workflow instances. It is possible to suspend a running workflow instance and return to the Workflow Modeling phase in order to introduce/remove orchestration logic or conduct Model-as-you-go operations such as iteration and re-execution. Additionally all choreographed workflow instances can be suspended in order to return to the Choreography Modeling phase and change the choreography model. The *Suspend* action also updates the choreography model if changes to the interconnected workflows affect the choreography model. Finally, after the execution of the workflows the scientist can analyze the final results in the *Analysis* phase and start the whole experiment life cycle again.

Fig. 2 shows the details of the top-down life cycle from the perspective of the realization by means of the BPM approach, i.e., showing all phases supported by a software system based on workflows that implements the life cycle perceived by scientist. For brevity we discuss only the phases that are not visible to scientists. To enable the interleaving of modeling and execution on the levels of both choreography and orchestration, we introduce four different adaptation cycles. The adaptation cycle at the bottom denotes adaptations 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 right hand side adaptation cycles indicate adaptations on the logic dimension, i.e. changes, to the orchestration and choreography logic. Additionally, Fig. 2 depicts the *Transformation* and *Deployment* phases. The Transformation phase realizes the Transform/Update action introduced in Fig. 1. It comprises the automatic steps necessary to transform a choreography model into the enacting workflows or update existing workflows, respectively. This also includes the generation of interface descriptions.

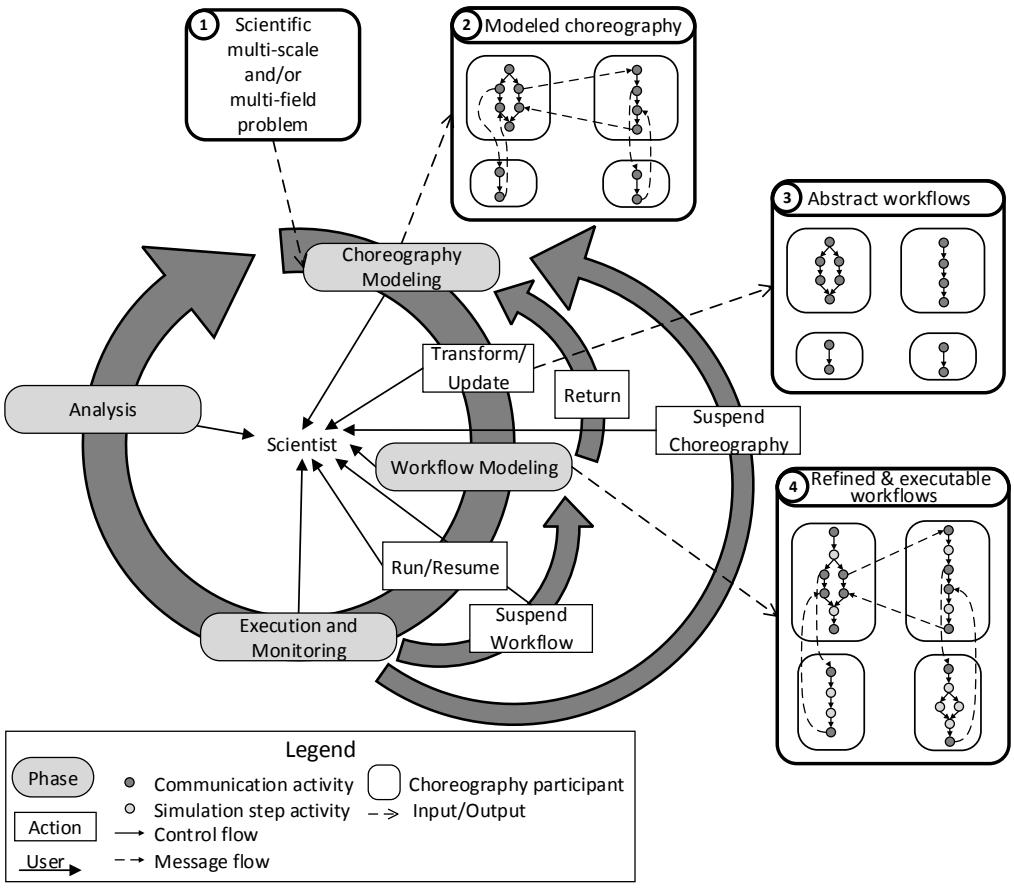

Figure 1: Top-down life cycle as perceived by the scientist

The *Deployment* phase realizes the Run/Resume action in Fig. 1. In this phase the refined 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. Therefore, appropriate deployment descriptors 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 created [19]. 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. Since 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.

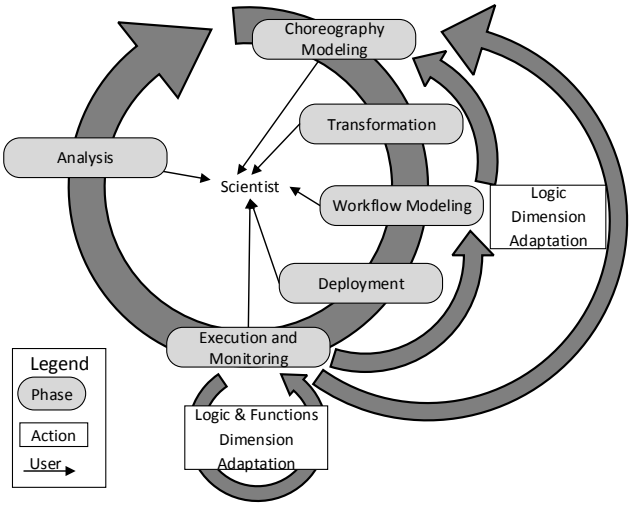

Figure 2: Top-down life cycle from the sWfMS perspective

As a result of this phase, the workflows that enact the choreography collaboratively are available for execution. The execution takes place in the *Execution and Monitoring* phase. Multi-* experiments are conducted by executing the refined enacting workflows and scientific services, which are participating in the choreography and are potentially distributed on several execution engines and service execution environments. The overall execution environment for the

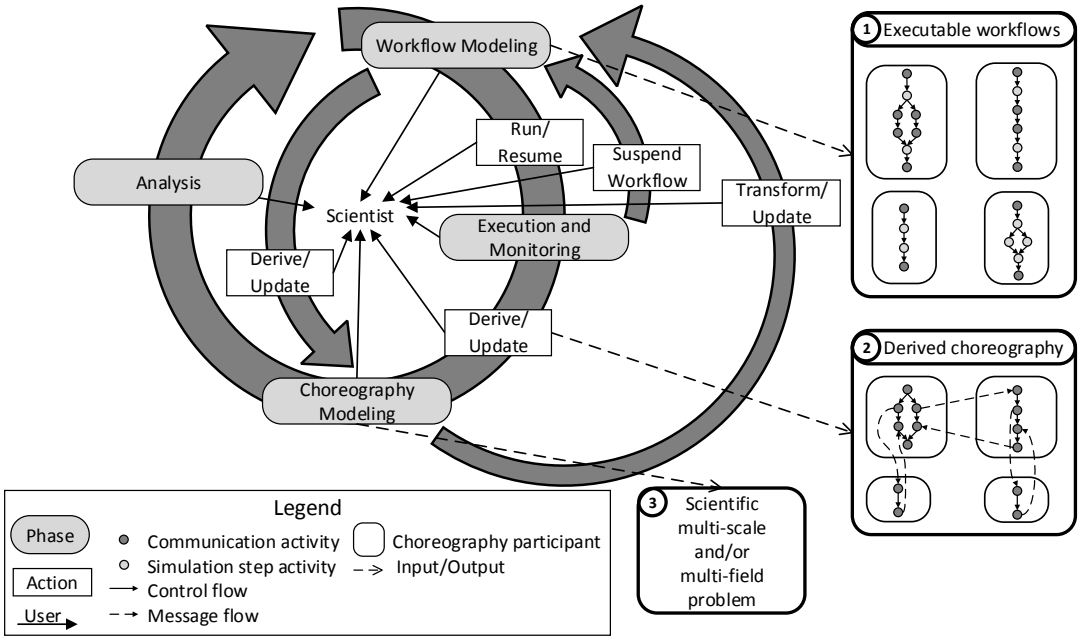

Figure 3: Bottom-up life cycle as perceived by the scientist

enacting workflows and scientific services has to support context-awareness and adaptation mechanisms to enable the flexibility needed for executing *Model-as-you-go for choreographies* operations. Some of the adaptations may be predefined in the choreography model, such as abstract activities that have to be refined at run time [20], reactions to context changes, and binding strategies for the simulation services [19]. 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 conducted by a scientist by pausing execution, performing Model-as-you-go operations both on the workflow and choreography level, and resuming execution. Furthermore, the forced termination of the choreography, the substitution of scientific service endpoints, the substitution of choreography participant types, or insertion of additional ones, and others. The overall execution environment must also be able to cope with an increasing number of scientists, simulation 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 re-executions of the scientific experiment. Execution and Monitoring of the running choreography must be indistinguishable for the scientist as in previous work [14]. 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 [19].

B. Bottom up life cycle

Fig. 3 shows the bottom-up modeling approach view on life cycle as the scientist experiences it. The life cycle starts with the *Workflow Modeling* phase and the modeling of workflows (1), which are interconnected but the interconnection is not explicitly captured by a choreography model. The scientist is able to run the workflows and proceed to the *Execution and Monitoring* phase, again 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-scale and/or multi-field choreography model (2) from the interconnected workflows. If the 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 the choreography editor. Finally, the derived choreography represents a specific scientific multi-scale and multi-field problem (3). The adaptation on the choreography level can be used to update the existing executable workflows in a *round-trip* fashion. The *Update/Transform* action modifies the underlying workflow models and at the same time enforces correctness of the changes. The life cycle is concluded by

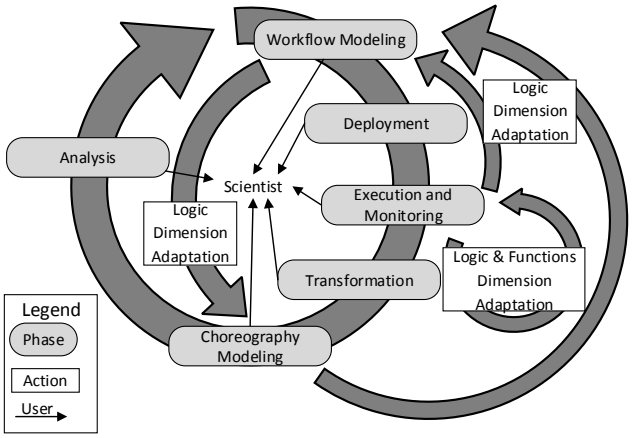

Figure 4: Bottom-up life cycle from the sWfMS perspective

the *Analysis* phase where the experiment result are evaluated.

Fig. 4 depicts the bottom-up modeling approach life cycle from the perspective of the supporting sWfMS, i.e., the life cycle that has to be implemented by a sWfMS to enable the perceived life cycle. As its top-down counterpart, the technical deployment phase has been hidden in Fig. 3 behind the Run/Resume action. Four adaptation cycles can be observed: In the *Execution and Monitoring* phase, the system is adapted along the functions and the logic dimension. The other three adaptation cycle 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. The Derive/Update action is technically realized by the *Derivation* phase. The Deployment and the Monitoring and Execution phase exhibit the same properties as discussed for the top-down life cycle in Sec. III-A.

IV. Related Work

In this section we compare our approach to existing ones documented in literature. Scherp and Hasselbring [21] 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 from scientists. 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 [22] 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 run time to facilitate a trial-and-error approach for the scientists.

Fleuren et al. [23] 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.

An architecture with centralized control flow and distributed data flow is introduced in [24]. A central concept in the proposed Circulate approach is the notion of a proxy. A proxy in this work is a middleware component that encapsulates web service invocation in order to avoid the necessity of reconfiguration of individual services. The service invocations are delegated to the proxies by a centralized engine. Data references are exchanged through the engine, whereas the actual data is directly transfered from proxy to proxy. While the centralized orchestrations can be modeled using BPEL or any other executable workflow language, it is not explicitly discussed how the interconnection of the choreographed proxies can be modeled or adapted.

Ludäscher et al. present a scientific workflow life cycle in [25]. In contrast to our work, the life cycle does not explicitly distinguish between top-down and bottom-up modeling views. Furthermore, it does not consider modeling multi-* experiments as choreographies to realize a distributed, collaborative experiments. In [15], a generic choreography life cycle is presented. However, it only supports choreography modeling beginning with a domain specific problem, i.e., top-down modeling. The life cycle presented in our work also considers starting bottom-up with executable workflows from which a choreography model is derived.

V. Conclusions and Future Work

In this paper, we introduced a life cycle that utilizes the notion of choreographies to enable the trial-and-error modeling and execution of multi-* simulations. The life cycle exhibits two views reflecting the characteristics of modeling and execution in a top-down and bottom-up manner. It supports the collaborative development and execution

of complex multi-* simulations involving software from different research or industry organizations. Furthermore, the life cycle is geared towards existing, standard-based technologies known from business applications that will be extended for composing data-intensive, scientific workflows in a generic, domain-independent way as choreographies and thus provide scientists with means for collaborative and integrated data management. 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 means of generic configurable connectors. With respect to the dynamic provisioning and de-provisioning of choreographed simulations we need to focus more 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 is also part of our future research plans.

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

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

[2] R. Krause, B. Markert, and W. Ehlers, "A porous media model for the description of adaptive bone remodelling," *PAMM*, vol. 10, no. 1, pp. 79–80, 2010.

[3] 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*, vol. 60, no. 20, pp. 6961–6971, 2012.

[4] M. Sonntag, D. Karastoyanova, and F. Leymann, "The Missing Features of Workflow Systems for Scientific Computations," in *GWW'10*. Gesellschaft für Informatik e.V. (GI), 2010, pp. 209–216.

[5] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, and M. Reiter, *Conventional Workflow Technology for Scientific Simulation*. Springer, 2011, pp. 323–352.

[6] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers, "Examining the challenges of scientific workflows," *Computer*, vol. 40, no. 12, pp. 24–32, 2007.

[7] I. Foster and C. Kesselman, *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier, 2003.

[8] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson, *Web Services Platform Architecture*. Prentice Hall, 2005.

[9] F. Leymann and D. Roller, *Production Workflows*. Prentice Hall, 1999.

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

[11] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, and F. Leymann, "SIMPL - A Framework for Accessing External Data in Simulation Workflows," in *BTW'11*, vol. 180, 2011, pp. 534–553.

[12] D. Schumm, D. Karastoyanova, F. Leymann, and S. Strauch, "Fragmento: Advanced Process Fragment Library," in *ISD'10*. Springer, 2010.

[13] M. Sonntag and D. Karastoyanova, "Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows," *Grid Computing*, vol. 11, no. 3, pp. 553–583, 2013.

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

[15] G. Decker, O. Kopp, and A. Barros, "An Introduction to Service Choreographies," *Information Technology*, vol. 50, no. 2, pp. 122–127, 2008.

[16] J. Zaha, M. Dumas, A. ter Hofstede, A. Barros, and G. Decker, "Bridging global and local models of service-oriented systems," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 38, no. 3, pp. 302–318, 2008.

[17] 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*. Springer, 2011, pp. 1–12.

[18] R. Barga and D. Gannon, "Scientific versus Business Workflows," in *Workflows for e-Science*. Springer, 2007, pp. 9–16.

[19] K. Vukojevic-Haupt, D. Karastoyanova, and F. Leymann, "On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows," in *SOCA'13*, 2013, pp. 1–8.

[20] A. Bucchiarone, A. Marconi, M. Pistore, and H. Raik, "Dynamic Adaptation of Fragment-Based and Context-Aware Business Processes," in *ICWS'12*, 2012, pp. 33–41.

[21] G. Scherp and W. Hasselbring, "Towards a model-driven transformation framework for scientific workflows," *Procedia Computer Science*, vol. 1, no. 1, pp. 1519 – 1526, 2010.

[22] M. Held, W. Küchlin, and W. Blochinger, "Mobiflow: Principles and design of a workflow system for molecular biology," *IJSSMET*, no. 4, pp. 67–78, 2011.

[23] T. Fleuren, J. Götze, and P. Müller, "Workflow Skeletons: Increasing Scalability of Scientific Workflows by Combining Orchestration and Choreography," in *ECOWS'11*. IEEE, 2011, pp. 99–106.

[24] A. Barker, J. B. Weissman, and J. I. van Hemert, "Reducing Data Transfer in Service-Oriented Architectures: The Circulate Approach," *IEEE Transactions on Services Computing*, vol. 5, no. 3, pp. 437–449, 2012.

[25] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers, "Scientific workflows: Business as usual?" in *BPM'09*. Springer, 2009, pp. 31–47.