# Conventional Workflow Technology for Scientific Simulation

**Chapter** · April 2011

DOI: 10.1007/978-0-85729-439-5_12

CITATIONS
63

READS
865

**5 authors**, including:

Katharina Görlach
Universität Stuttgart
**12** PUBLICATIONS **206** CITATIONS

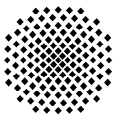
SEE PROFILE

Dimka Karastoyanova
University of Groningen
**170** PUBLICATIONS **1,982** CITATIONS

SEE PROFILE

Frank Leymann
Universität Stuttgart
**828** PUBLICATIONS **27,500** CITATIONS

SEE PROFILE



Universität Stuttgart

SimTech
Cluster of Excellence

K. Görlach M. Sonntag D. Karastoyanova F. Leymann M. Reiter

# Conventional Workflow Technology for Scientific Simulation

Stuttgart, March 2011

**Institute of Architecture of Application Systems (IAAS)**
University of Stuttgart,
Universitaetsstrasse 38
70569 Stuttgart, Germany
{goerlach, sonntag, karastoyanova, leymann, reiter}@iaas.uni-stuttgart.de
www.iaas.uni-stuttgart.de

**Abstract** Workflow technology is established in the business domain for several years. This fact suggests the need for detailed investigations in the qualification of conven-tional workflow technology for the evolving application domain of e-Science. This chapter discusses the requirements on scientific workflows, the state of the art of scientific workflow management systems as well as the ability of conven-tional workflow technology to fulfill requirements of scientists and scientific ap-plications. It becomes clear that the features of conventional workflows can be advantageous for scientists but also that thorough enhancements are needed. We therefore propose a conceptual architecture for scientific workflow management systems based on the business workflow technology as well as extensions of exist-ing workflow concepts in order to improve the ability of established workflow technology to an application in the scientific domain with focus on scientific simulations.

**Keywords** Business workflows, BPEL, Scientific workflows, Simulation, Workflow Management Systems.

**Reference** Görlach, K, Sonntag, M., Karastoyanova, D., Leymann, F., and Reiter, M. (2011) Conventional Workflow Technology for Scientific Simulation. In: Yang, Y. (ed), Wang, L. (ed), and Jie, W. (ed) Guide to e-Science, Springer-Verlag.



**Stuttgart Research Centre for Simulation Technology (SRC SimTech)**

SimTech – Cluster of Excellence
Pfaffenwaldring 7a
70569 Stuttgart
publications@simtech.uni-stuttgart.de
www.simtech.uni-stuttgart.de

# Conventional Workflow Technology for Scientific Simulation

Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova,
Frank Leymann, Michael Reiter

Institute of Architecture of Application Systems
University of Stuttgart
{goerlach, sonntag, karastoyanova, leymann, reiter}@iaas.uni-stuttgart.de

Workflow technology is established in the business domain for several years. This fact suggests the need for detailed investigations in the qualification of conventional workflow technology for the evolving application domain of e-Science. This chapter discusses the requirements on scientific workflows, the state of the art of scientific workflow management systems as well as the ability of conventional workflow technology to fulfill requirements of scientists and scientific applications. It becomes clear that the features of conventional workflows can be advantageous for scientists but also that thorough enhancements are needed. We therefore propose a conceptual architecture for scientific workflow management systems based on the business workflow technology as well as extensions of existing workflow concepts in order to improve the ability of established workflow technology to an application in the scientific domain with focus on scientific simulations.

**Keywords:** Business workflows, BPEL, Scientific workflows, Simulation, Workflow Management Systems

## 1. Introduction

Originally, workflows have been created to meet the IT support needs of the business world. In short, they are compositions of tasks (also referred to as activities) by means of causal or data dependencies that are carried out on a computer. They are executed on a workflow management system (WfMS) [1]. A workflow that utilizes Web services (WSs) as implementations of tasks is usually called *service composition.* Web services are the most prominent implementation of the service-oriented architecture (SOA). The Web Service technology is an approach to provide and request services in distributed environments independent of programming languages, platforms, and operating systems. It is applied in a very wide range of applications where integration of heterogeneous systems is a must.

In recent years the workflow technology has gained more and more attention in the scientific area [2] and the term *scientific workflows* has been coined. Workflows in science provide multiple benefits: (1) They contribute to sharing knowledge by being available as services for collaborating scientists; (2) with the help of workflows a community-based analysis of results is supported; (3) workflows are able to deal with huge amounts of data, e.g. collected by sensors or calculated by scientific algorithms; (4) workflows are capable of running in distributed and highly heterogeneous environments—a common scenario in scientific computations where a great variety of platforms and programming languages is usually employed; (5) the automation of steps during workflow design and execution allows scientists to concentrate on solving their main scientific problems; and (6) workflows can be utilized to conduct scientific simulations in a parallel and automated manner.

Since the WS and workflow technologies are established in the business area, it is reasonable to use them in the scientific domain, too, especially because the two areas exhibit many similar requirements on the IT support. For example, WSDL (Web Services Description Language) [4] can be used for the specification of service interfaces and BPEL (Web Services Business Process Execution Language) [5] for the specification of scientific workflows. As there are much more WS standards and a lot of additional requirements imposed by the scientific domain, there is a need for an advanced analysis of usability of the WS and Workflow technology in the field of scientific applications. This chapter discusses the requirements on workflows in the scientific domain, and the ability of conventional workflow technology regarding these requirements. Furthermore, we propose extensions of conventional workflow technology that significantly improve the ability to apply established workflow technology in the scientific domain.

A special kind of scientific workflows we focus on are simulation workflows. Simulations are typically *complex calculations* which predestinate them for a realization with workflow technology. Examples are partial differential equations (PDE) that must be solved to determine temporal or spatial changes of simulated objects. *Remote access* to data outside the actual scientific workflow management system (sWfMS) is another characteristic of simulation workflows.

In this chapter, after introduction we will discuss the application of conventional workflow technology in the scientific domain in detail. Therefore, in Section 2 we will point out the most important requirements on scientific workflows and compare them with requirements on conventional, i.e. business workflows. Furthermore, we will present a workflow system architecture meeting the requirements on scientific workflows and a prototype implementing this architecture. In Section 3 we will show an example scenario for the simulation of “ink diffusion in water” implemented with the help of our prototype. Afterwards in Section 4 we will present extensions of conventional workflow technology in order to bridge the gap between conventional workflow technology and unfulfilled requirements arising by the new application domain e-Science. Finally, in Section 5 we will conclude and look out on future work.

### 1.1. Background and Related Work

Several approaches were already created for using workflows in scientific applications to meet some of the requirements imposed by that domain. Existing scientific WfMSs are usually built from scratch and do not rely on the workflow technology as it is established in the business area. Here we give a short overview of the systems Triana, Taverna, Kepler, and Pegasus that focus on a specific scientific domain and/or on supporting specific functionality. Additionally we briefly introduce Trident which is one of very few systems that applies conventional workflow technology for scientific applications.

*Triana* [6] focuses on supporting services and workflow execution in distributed environments. It provides predefined services, for example signal or image processing, static analysis and visualization, useful in diverse domains. Such local or remote services as well as local applications can be composed in workflows. Triana workflows are data-driven and specified with a WSFL[1]-like notation. For workflow and service execution Triana basically supports peer-to-peer systems and Grid environments that enable dynamic resource allocation. Recently, efforts are made towards supporting WSs and integrating WSRF [7].

*Taverna* [8] is a workbench for biologists that allows a data-centric specification and execution of workflows. In contrast to Triana, it supports only services with known locations. Taverna aims at supporting long-running, data intensive workflows and the interaction with users. It provides semantic service discovery, fault handling mechanisms and provenance information.

Similarly to Taverna, *Kepler* [9] is mainly used in the bioinformatics domain. It allows the integration of different types of resources (e.g. databases or Grid resources) and of different tools (e.g. MatLab[2]). Kepler provides a service library that includes services with capabilities for advanced data exchange, e.g. mediation or data hub functionality. These services can be used to build compositions. So-called directors enable flexible control strategies for service compositions. For execution a Kepler workflow is translated into a Java program and the utilized services are mostly local Java applications, too.

*Pegasus* [10] is rather a workflow compiler than a WfMS. It cooperates with Condor DAGMan[3] and supports input of DAGMan workflows. Such workflows represent a template without resource bindings. These templates are the basis for Pegasus users to describe their demands for the resource mapping. Based on this template Pegasus determines an execution strategy. For optimized strategies intelligent planning algorithms are used to cope with data movements and application runs on heterogeneous and dynamic execution environments. Additionally, Pegasus aims at reducing the accumulated amount of data during workflow execution.

---

[1] Web Services Flow Language: http://www.ibm.com/developerworks/library/ws-ref4/

[2] MathWorks Website: http://www.mathworks.com/

[3] http://www.cs.wisc.edu/condor/dagman/

More precisely, it identifies existing data that does not need to be produced once more and it performs garbage collection.

In contrast to previous approaches *Microsoft Trident* [11] applies conventional workflow technology. It uses the Microsoft Workflow Foundation as technology platform and a control flow-oriented modeling language based on the Extensible Orchestration Markup Language (XOML). Trident also supports some modeling extensions like data-flow edges. The Microsoft system consists of independent components for workflow modeling and execution. For some components multiple implementations exist. For example scientists can choose between different workflow composers (i.e. a modeling tool) like a text editor or a graphical modeler. A workflow composer offers basic and composite activities that can be chosen from a workflow catalog. Additional activities can be created by scientists using Visual Studio. Furthermore the user can customize the workflow composer by including domain-specific workflow packages that exist for astronomy, biology, meteorology or oceanography. For the execution of workflows two different ways are provided. First, it is possible to execute the workflow in the Trident WfMS. Second, a workflow can be executed by compiling it into a usual application and run it on Microsoft .Net platforms.

In the remainder of this chapter we will use the terms process model, workflow model, process (instance) and workflow (instance) in the following meaning: In conventional workflow technology a *process model* depicts the structure of a process in the reality. A real word process model describes in many cases human interactions. A part of a process model that is executed on a computer is called a *workflow model* [1]. A workflow model specifies actions that need to be performed and control or data flow dependencies between these actions. This workflow model can be seen as a template from which each *workflow* is instantiated, i.e. a *workflow instance* created from a workflow model. After all an individual workflow instance can be executed by a workflow engine.

## 2. Applying Conventional Workflows for Scientific Computations

As shown in the previous section only few approaches rely on the conventional workflow technology as described in [1] or in the Workflow Reference Model[4]. Since it is an established technology there already exist standards and tools that can be used as basis for further development. Therefore, we make efforts to use the conventional workflow technology for scientific computations.

At first life cycles of conventional and scientific workflows are presented and compared. The outlined differences in these life cycles reveal the need for extensions of the conventional workflow technology when being applied for scientific simulations. Afterwards main requirements on WfMSs in the scientific domain are

[4] http.//www.wfmc.org/reference-model.html

discussed and a conceptual architecture for sWfMSs is proposed that fulfills these demands. Finally, a prototypical implementation of this architecture is presented.

### 2.1. Life Cycle of Workflows

The workflow life cycle, is important to understand the actions needed to set up and run workflows as well as the user roles that carry out these actions. In conventional workflow technology, the life cycle is well-known and accepted by the community [1]. It consists of different repeatable management phases that are dealt with by different user groups or roles (see Figure 1a).

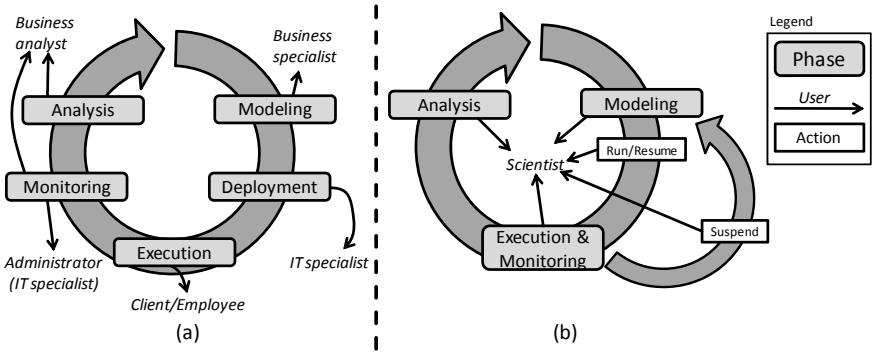



**Figure 1: Life cycle of business (a) and scientific workflows (b).**

A workflow is modeled by business specialists that know the concrete steps that are performed to achieve a certain business goal. A specialist with IT knowledge explicitly deploys the workflow on an engine, i.e. makes it executable. Execution of a workflow is triggered by a client or employee, often late after deployment. A workflow model can be instantiated multiple times. Workflow monitoring can deal with individual running workflow instances or aggregated information over several instances. It can also present statistics on the overall state of a system including running and completed workflow instances. Hence, monitoring can be valuable for both administrators/IT specialists and business analysts. Finally, a business analyst analyses one or more workflow executions and may deduce a need for business process reengineering.

The life cycle for scientific workflows heavily distinguishes from its business counterpart (see Figure 1b). We inferred the life cycle from observations about the way scientists create and conduct experiments and from known properties of scientific simulations and computations [12], [13], [14]. Typically, there is only one user group, the scientists, playing the roles of a modeler, user, administrator, and analyst. The focus of their work is usually on single workflow instances. To be precise, scientists do not distinguish between workflow models and instances or are not aware of a difference between models and instances, respectively. They set up a simulation and execute it once at a time. Because scientists typically develop

their workflow in a trial-and-error manner, modeling and execution phases are not arranged in a strict sequence. In fact, they can be carried out alternated. An additional cycle therefore leads from execution back to modeling phase with the help of a "suspend" operation on the workflow instance, which remains hidden from the scientists. Technical details are transparent for scientists altogether. For instance, conventional workflow adaptation is experienced as scientific workflow modeling; the deployment (of parts of a workflow) is part of the "run/resume" operation [15], which is also hidden for the scientist. Workflow execution starts immediately after modeling. The traditional execution and monitoring phases are merged into a single phase in the scientific workflow life cycle because from a scientist's point of view monitoring only visualizes a running workflow (instance). After execution a scientist can analyze the computed results and may re-model and re-execute the workflow possibly with different parameters.

### 2.2. Main Requirements on Scientific Workflow Management Systems

Workflows in the scientific area make use of a data-centric approach. Typically, huge amounts of data have to be processed, e.g. 5 GB data per day is transmitted by the Hubble telescope, and probably need to be processed by a workflow. The modeling language for scientific workflows is primarily *data-driven*, sometimes with support of a few control structures (e.g. in Taverna, Triana). That means, for accommodating the scientists' needs the main focus of the modeling language should be on data flow while the control flow is considered secondary. Additionally, the language has to provide modeling constructs for *advanced data handling*, e.g. data references, and pipeline mechanisms since typically huge amounts of data are processed [34].

One of the most important requirements on sWfMSs is *usability* since the majority of users of sWfMSs are no computer scientists. A scientist needs the support of easy-to-use tools, automation as far as possible and maximal flexibility in the usage of the sWfMS. First, this means that scientists want to model their workflows in a convenient way. Second, they want to run their workflows and store their data on resources that are specified by the user himself or automatically chosen by the sWfMS. Scientists need the same support for services used in traditional workflows, i.e. scientists should be able to specify the services to be used in a scientific workflow themselves, or to delegate the service discovery to the sWfMS. The need for automation in sWfMSs basically includes the deployment of workflows, the provisioning of workflows, services and data, the instantiation and execution of workflows, and the service discovery. Additionally, the automation of data transformation is desirable as well as support of data vs. function shipping decisions.

A sWfMS should be *flexible*. With flexibility we denote the ability of a system to react to changes in its environment. Approaches to flexibility of workflows can be divided into two groups. First, a workflow can be modified—be it automatical-

ly or manually—according to the changed situation (known as *adaptation*). Such modifications can address different workflow dimensions (logical, functional, organizational dimension [1]) and may be applied to workflow models or single workflow instances. Second, a workflow can be modeled in a way that avoids its modification even in the presence of a changing environment (known as *avoid change*). Avoid change can be achieved by different mechanisms (e.g. role-oriented staff queries on organizational databases, alternative paths in the workflow model, automatic re-execution of a task).

The need for flexibility mechanisms in scientific applications and simulations is manifold. Setting up a scientific experiment is often not a chain of actions that lead to the expected result right from the beginning, but rather a trial-and-error process (cf. Section 2.1) [33]. That means a scientist examines an experiment while conducting it. If it deviates from the expectations, the scientist modifies the workflow to get the desired course. These modifications comprise changing the workflow structure, adding or removing activities, correcting erroneous data, and others. A sWfMS is therefore required to support these kinds of process logic adaptation, ideally especially for running workflows.

Usually, scientific computations are dealing with huge amounts of data or allocate enormous computing resources. Thus, they are typically long-running although often being conducted in powerful computer clusters or Grid environments. The reliability of the underlying infrastructure cannot be guaranteed: networks may fail; servers may be temporarily unavailable due to administration; servers may move to another location. Hence, failures during execution cannot be avoided especially in scenarios with long-running computations. If an application is not explicitly programmed to cope with such unforeseen failures, its unsuccessful termination would mean a loss of data, time and money. Avoid change mechanisms are required: if an activity implementation (i.e. a computation) is not available, e.g. because of a server crash, another implementation with identical semantics may be invoked at run time without interrupting the workflow execution.

During workflow execution scientists want to *monitor* the process run. Therefore, a sWfMS should support the ability to monitor the status of the workflow execution. For example, the scientist is interested in knowledge about running, finished or faulted activities, allocated resources, and the dynamical choice of a service implementation. Inspecting the produced data is also a need.

After carrying out a scientific experiment its *reproducibility* is of utmost importance for different reasons. Scientists who are not involved in the execution of the workflow need to be able to retrace the simulation, have to review the findings, or need to use the results. The operating scientist may want to repeat a workflow or parts of it to draw conclusions, and to prove statements and assumptions. Additionally, the scientist may share the results with people he collaborates with. While following and reproducing workflow runs and their (intermediate) results scientists use provenance information [16]. Data provenance enables scientists to study new results and determine data derivation paths. Workflow provenance enables

scientists to study process execution and covers the flow of a workflow instance including the nature and the course of decisions in the execution path, used resources and services, meta data (e.g. timestamps) for service invocation, and error logs. All relevant data has to be stored by a sWfMS. Finally, provenance information should be displayed in a way that (non-computer) scientists understand the execution of workflows including the derivation of data.

The *robustness* of scientific workflows is an important issue since scientific workflows are long-running. The term robustness denotes the ability of being error-resistant. The needed flexibility mechanisms mentioned above are a way to improve the robustness of a system. But additional approaches are needed to protect the execution progress from being lost in case of unforeseeable failures, to reach a consistent system state even in the presence of failures, and to proceed a simulation/experiment after a failure.

*Scalability* of sWfMSs enables acceptable performance for workflow execution. Scientific workflows should scale with the number of users, number of utilized services, data or calculation resources and involved participants. Today, typical scientific workflows are mostly executed in a central manner on a single machine. A decentralized workflow enactment can help to scale via distributed process execution. It can be achieved, for example, by parallel execution in distributed and heterogeneous execution environments using provisioning techniques.

The requirements presented above are valid for the entire scientific domain. Certainly, there are *specific requirements* for each specific scientific domain, e.g. life science, medical science, chemistry or mechanical engineering. Specific domain related requirements and the fact that it is hard to cover all scientific domains in one WfMS create the need to extend sWfMSs and the workflow models and possibly adapt these models and their instances. This includes, for example, domain specific services, result displays or meta-models for semantic annotations.

The special type of scientific workflows we focus on, the simulation workflows, imposes additional requirements on sWfMSs. Like scientific workflows simulations are characterized by data-intensive and compute-intensive load. Naturally, simulations represent long-running calculations since scientists frequently study complex systems. Therefore, simulation workflows require the support of long-running workflow instances. Additionally, simulation workflows demand the possibility to integrate different tools in one simulation workflow as well as a heterogeneous execution environment with integration of databases, computation nodes and sensor nets. Finally, simulations represent a special domain that requires domain-specific modeling language constructs. For example, complex simulations can consist of multiple cohering workflows. Therefore there is the need to share context data between these workflows in a simulation. Furthermore, the user should be supported in typical simulation tasks like parameter searching, for example.

### 2.3. Comparing Conventional and Scientific Workflow Technology

The main intention of (conventional) workflows is the *automation* of processes. In conventional workflow technology this comprises the instantiation and execution of workflows. Additionally, humans have to be integrated in the course of actions which results in an automated "invocation" of human beings during execution of workflows. This requirement is not among the most important requirements for scientific workflows. Nevertheless, scientific workflows can benefit in this field since setting up and conducting scientific experiments and simulations often includes manual tasks that are currently not part of scientific workflows. However, in general scientists make higher demands on automation while using workflow technology. As non-computer scientists they want to use the whole power of workflow technology without the need for further education, i.e. they require an automated (i.e. hidden) deployment of workflows and services since the location of execution is not predefined in all cases in e-Science. Rather in the workflow scientists want to specify the location where services have to be automatically deployed and executed. Supplementary, this creates the need of automated service and workflow provisioning. Finally, an automated data transformation as well as an automated service discovery is desirable for scientific users. Triana [6] provides the specification of the location for service execution. With the help of this mechanisms it even allows to specify the way sub-workflows have to be executed, i.e. how many instances of a sub-workflow have to be created and the distribution of these instances in the execution environment. A special control unit is responsible for data transfer realizing data dependencies between the instances. For the distribution of sub-workflow instances Triana also provides an automatic specification.

Allmost all sWfMSs provide a *service catalog* with predefined services. When starting the particular sWfMS it automatically searches for these predefined services and if they are still available the user can choose them in the service catalog for integration into workflow models. Although these service catalogs are extendable an automatic service discovery based on semantic data is not sufficiently considered by sWfMSs in most cases until now. The sWfMS Triana provides advanced mechanisms for workflow deployment and execution since it is specialized on distributed execution of workflows. However, most sWfMSs are specialized by provided services in the service catalog. For example, Taverna[8] and Kepler[9] provide mainly services needed for biological algorithms or services providing access to biological databases. Moreover, a service catalog is an important factor of the usability of a sWfMS.

Next to the service catalog the *usability* of a sWfMS strongly depends on supported tools needed for the realization of the whole workflow lifecycle. Hence, Taverna, Triana and Kepler provide the modeling of workflows (including the support of a service catalog), starting workflow runs "by mouse click", integrated *monitoring* and result visualization tools. In summary easy-to-use tools in combination with wide-ranging automation are key factors of usability. In conventional

workflow technology required tools often exists but their usability leave a lot to be desired for scientific users. Additionally, service registries allowing service catalogs in WfMS are not sufficiently supported. As suggested before the scope of automation in conventional workflow technology needs to be extended in order to meet requirements on usability for scientific workflows.

Most conventional *modeling languages* for business workflows are control flow-driven since they are designed for the implementation of control flow-oriented processes. However, scientists think data-oriented which creates the need for new modeling constructs in order to allow for example an explicit specification of data flow in scientific workflows. The modeling languages of sWfMSs (e.g. in Taverna and Triana) are mostly hybrid languages that are mainly data-driven with support of a few control structures. Intentionally the additional control structures allow an enhanced support of controlling mechanisms for the data flow.

Futhermore, modeling languages for scientific workflows have to support advanced data handling mechanisms. In conventional workflow technology the handling of huge amounts of data is not a primary requirement. Therefore, new mechanisms should upgrade the workflow technology. For example Pegasus [10] optimizes the execution of scientific workflows by reusing data that was already generated and is still up to date. Hence, the workflows' execution time can be reduced because the data does not have to be generated again.

In conventional workflow technology the *reproducibility* of workflow runs particularly comprises executed activities, invoked services and corresponding resources during workflow execution. For scientists research results are only accepted and usable if they are repeatable which constitutes the need for the collection of huge information sets in order to enable the reproducibility of almost the whole workflow run. Especially, existing monitoring and auditing mechanisms of conventional workflow technology are not sufficient. Origin of the data and their manipulation is not tracked but this type of information is of high importance for scientists. Therefore in sWfMSs a special provenance component should support the traceability and reproducibility of workflow runs. That component combines information about a workflow instance including the nature and the course of decisions determining the path through the workflow with information about the origin and the manipulation of data.

Requirements on *robustness* are quite similar for conventional and scientific workflows. Recoverability of faulty workflow runs is an important issue already considered by conventional workflow technology. Therefore, scientific computations seriously benefit from conventional workflow technology. However, an increased flexibility of workflows would further improve their robustness [25]. Imagine a workflow that automatically reacts to a changed environment: if a server becomes unavailable, an alternative server can be chosen at runtime; if a network connection error occurred, a simple retry operation could solve the problem. Although flexibility is a key point in scientific workflow management, it is insufficiently unaddressed in currently existing scientific workflow systems. For example, Triana and Kepler just allow the modification of simulation/experiment

parameters during workflow execution as adaptation mechanism; Pegasus and Taverna implement avoid change concepts such as retry of service invocation [33]. In conventional workflow technology, there are already many approaches addressing the mentioned kinds of flexibility. We believe that it is of high value to harness the results of existing concepts.

*Scalability* is already addressed by conventional workflow technology regarding tasks in workflows, workflows themselves and humans integrated in workflow execution. Since scientific workflows require a dynamic infrastructure for workflow and service execution, scalability regarding resources strongly suggests the execution of scientific workflows in Grid or Cloud environments. Of course the execution of conventional workflows in Grids and Clouds is already an ongoing issue since it seems natural to use workflow and service based execution mechanisms in such kind of distributed execution environments. In the scientific domain Pegasus is especially designed for the execution of workflows in Grid environments. It plans the execution of scientific workflows in Grids, i.e. Pegasus maps Grid resources to workflow tasks. Triana also enables the execution of workflows in Grids and additionally introduces an abstraction level (in the form of an API) that allows scientific users to integrate different types of infrastructures in the same way.

For scientific of *simulation* workflows conventional workflow technology can deal with the demand on heterogeneous execution environments and their long-running character. The mechanism for the integration of hardware like databases in conventional workflow technology constitutes a beginning for the integration of sensor nets as well. Nevertheless, further studies are needed in order to integrate sensor nets in an efficient way also utilizing valuable characteristics of sensor nets. Conventional workflow technology also easily enables the integration of different tools in a workflow implementing one simulation. Nevertheless, the data exchange between different tools is still difficult since they rarely share the same data format.

The interaction between workflow models assembled in one simulation is modeled in a so called choreography [3] in conventional workflow technology. However, in order to assemble simulation workflows the support of shared context data is missing. In the scientific domain only a small number of WfMS especially promote the application for simulations. All together do not consider choreographies of workflows. Amongst other things Kepler is used for the implementation of simulations (cf. [21]). The support of tools like MatLab emphasizes the competence of Kepler for the modeling and execution of simulation workflows. A language that specifically addresses requirements of scientists on a workflow language for simulations is GriCoL [24]. For example it supports special mechanisms for parameter searching. In order to prove the concepts a use case “molecular dynamics simulation of proteins” was successfully implemented by biology scientists using GriCoL [24].

Finally, it has to be noticed that discussed sWfMSs except Trident [11] do not follow the architecture given by conventional management systems. Nevertheless,

they are representatives of sWfMSs established in e-Science. Typically, early sWfMSs set a high value on frameworks integrating tools for the support of the whole life cycle and emphasize less on a sophisticated runtime environment for workflows. Upcoming sWfMSs, e.g. Trident or Sedna [19] confirm the approach to use conventional workflow technology in e-Science since they successfully integrate its mechanisms and benefit from its attributes.

### 2.4. Architecture for Scientific Workflow Management Systems

In order to meet the requirements on WfMSs in the scientific domain we propose a conceptual architecture for sWfMSs that is shown in Figure 2. Individual components in this architecture are discussed in the following.

Since the user interface offers the functionality provided by the sWfMS to a (non-computer) scientist the usability is the main requirement to be met by the user interface. We propose a *Graphical User Interface (GUI)* with four main components: a service catalog, a workflow modeler, a monitor and a result display. The GUI is connected to the runtime environment to realize workflow deployment, execution, monitoring, and other functionalities.

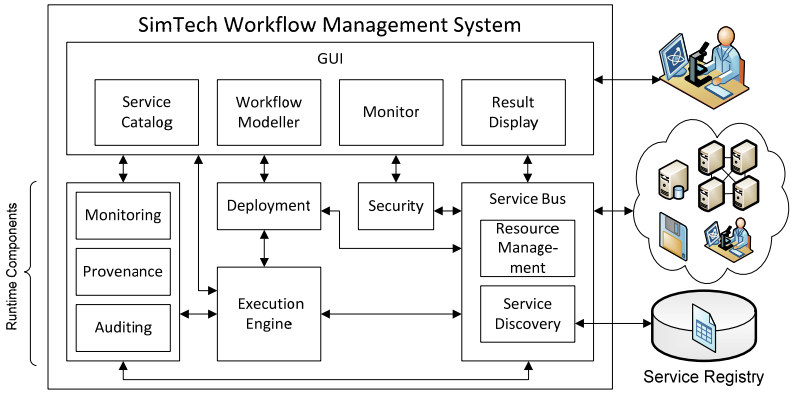



**Figure 2: Architecture of a simulation workflow management system.**

The *service catalog* provides the user with a list of available services that can be used in the workflow modeler. The discovery of such services includes among other things the search for new services in the user domain and the identification of services that are no longer available. The underlying mechanisms should be initiated either explicitly by the user or automatically by the sWfMS.

The *workflow modeler* supports the scientific user during workflow specification/modeling. Graphical modeling as well as program like “modeling” has to be provided since many scientists prefer to code their workflows. The modeling language is mainly data-driven with support of control structures. Additionally, the language has to provide mechanisms for the handling huge amounts of data.

The GUI enables the user to select his favored services from the service catalog and compose them using the workflow modeler. The deployment information of such services can be specified by the service provider itself, by the workflow modeler, or automatically by the sWFMS. For example, lots of parallelism in scientific workflows is achieved by parallel workflow instances. The number of these instances can be huge if multi-scale simulations or for parameter search in simulations are conducted (where one instance for each element in the parameter range is created). In such cases it is reasonable to distribute the execution of instances which in turn only can be decided in the workflow context. These and other scenarios impose the necessity for specification of service deployment information by the workflow modeler. Since only advanced modelers use this feature default deployment information is needed for the cases where this information is specified neither in the service catalog nor by the user.

Scientists should be able to execute workflows using the GUI. In order to improve the usability of the GUI workflow and data provisioning, workflow and service deployment as well as workflow instantiation and execution should be automated and thus rendered transparent to the user.

The *monitor* is the third main part of the GUI. It allows the user to inspect the workflow execution and to identify unexpected occurrences or faults in the workflow execution. In order to afford a user reaction the user should have interactive control on workflow execution and possibility to adapt particular workflows.

Using the *result display* the final outcome as well as intermediate results of a workflow are presented in an appropriate way. The kind of presentation depends very much on the domain and on the particular data with its specific structure. Support of simulations additionally requires the availability of result visualization. For a customized result visualization adapted to the particular simulation it is possible to compose special services in a visualization workflow that is part of the simulation workflow. Overall, the result display has to be adaptable/configurable in order to meet the different user requirements.

The main purpose of the *runtime environment* is to carry out the workflows. This comprises navigation through a workflow, maintaining its state, and invoking service implementations, storing workflow specific run time data and dealing with experiment dates and simulation results. The architecture's run time components (see Figure 2) are similar to those of existing conventional WfMSs but extended by additional functionality to meet the requirements related to the scientific area, e.g. provenance tracking or scientific data handling. The runtime components for the simulation WfMS are the execution engine, the service bus, the security and deployment components as well as monitoring, auditing and provenance. All mentioned modules may have access to persistent storage mechanisms to durably save process data. These process storage components differ from the scientific data handling components and are not shown in Figure 2 for better readability.

The *execution engine* runs the scientific workflow instances, which means that it instantiates workflow models with provided input data, navigates through the

graphs of activities, and triggers the execution of activity implementations and handles faults and events from the outside. The engine thereby maintains instance data such as the state of a workflow instance and its activities and variables. Furthermore, the execution engine should support transaction concepts in order to support fault handling, e.g. by compensation of completed work. The engine should provide mechanisms to cope with workflow model and instance adaptations at run time.

The *service bus* primarily deals with the task of invoking services that implement workflow activities. Therefore it discovers and selects services, routes messages, and transforms data. Two major components can be distinguished: the resource management and the service discovery.

Since resources have different properties, the *resource management component* is used, for example, to identify servers or Grids that have enough storage capacity and calculation power to carry out a computationally intensive task. Furthermore, the resource management is responsible for data vs. code shipping decisions at run time.

The *service discovery* queries service registries (such as UDDI) to find services by means of descriptive information (e.g. interfaces, and semantic annotations). The service discovery component delivers a list of candidate services. On the one hand, this capability is used by the service catalog component of the GUI. On the other hand, it is the basis for flexibility mechanisms regarding activity implementations, i.e. late binding [26] and rebinding of failing activities. The late binding strategy denotes the process of selecting a concrete service as late as possible, i.e. during workflow execution at the latest.

The *deployment component* transforms workflow models into an engine-internal representation, installs them on the engine, and publishes the workflows as services. A so-called deployment descriptor prescribes how to make the workflow runnable on a specific engine. The deployment descriptor may, for instance, statically bind services, specify selection rules, or influence resource management decisions. Although the deployment is triggered by the GUI its complexity should be hidden especially to scientific users.

The *security component* has two main functions, namely to ensure both local and remote security policies. The local security protects the GUI from unauthorized access. The remote security protects services provided by the sWfMS from remote access. Furthermore, it enables the service bus to access secured resources such as scientific databases that request a user login.

Finally, there is a group of components that are rather passive with respect to workflow execution: the auditing, provenance and monitoring component. The *auditing* component is responsible for recording workflow or activity related events, e.g. the start time of a workflow or the duration of an activity execution. The *monitoring* component uses the events related to a single workflow run and indicates the status of this workflow run. The *provenance* component records data that goes beyond simple auditing information. All together these components enable the examination of workflow runs and their reproducibility.

## 2.5. Prototype

In the context of the DFG Cluster of Excellence Simulation Technology (SimTech)[5] we developed a first prototype that implements the conceptual architecture for sWfMSs mentioned above. SimTech concentrates on simulation technology, ranging from molecular dynamics and modern mechanics over numerical mathematics and systems analysis to data management und interactive visualization as well as high performance computing. In our field we make efforts in creating a sWfMS especially tailored to requirements for simulation workflows. The underlying technical complexity ought to be hidden for scientists to enable them to concentrate on their core competencies. The prototype presented in this chapter focuses on the modeling of scientific and simulation workflows. We decided to rely on BPEL [5] as workflow language for various reasons. It is widely accepted in industry and research, supports integration of legacy applications by using WSs as activity implementations, and there are a number of open source BPEL tools that can be used as basis for a sWfMS. However, in order to satisfy requirements of scientists and simulation workflows, extensions to standard BPEL are needed.

We use the Eclipse BPEL Designer[6] as starting point for a tool to model simulation workflows but the tool needs extensions in order to support newly introduced modeling constructs. On the basis of Eclipse we implemented different perspectives corresponding to the different phases in the lifecycle of simulation process management. A “SimTech modeling perspective” (see Figure 3) provides for the design of workflows with the help of the modeler and the service catalog. The “SimTech runtime perspective” supports the user during workflow execution to follow the simulation progress. The result display shows intermediate results in this perspective. At last the “SimTech analysis perspective” allows analyzing the outcome of workflow runs and therefore contains the result display and the monitoring component. In contrast to the runtime perspective, the result display is highlighted here.

The service catalog was recently implemented and allows the use of services as building blocks in DUNE[7] (Distributed and Unified Numerics Environment) simulations represented as BPEL workflows. Currently, efforts are being made to support other simulation frameworks and visualization services. With the visualization services users are able to specify result visualization in a way similar to the actual simulation. The monitor is a proprietary implementation which is fitted for the special needs in simulations. This means that for example in DUNE simulations, special DUNE events can be monitored instead of ordinary BPEL events. In our further work we will improve the monitor component with additional features. The implementation of the result display is currently in progress. It will enable a

[5] http://www.simtech.uni-stuttgart.de/

[6] http://www.eclipse.org/bpel/

[7] DUNE, a C++ template library for solving partial differential equations with grid-based methods: http://www.dune-project.org/

user defined result display. Calculation steps needed to display data will be provided as services and workflows respectively.

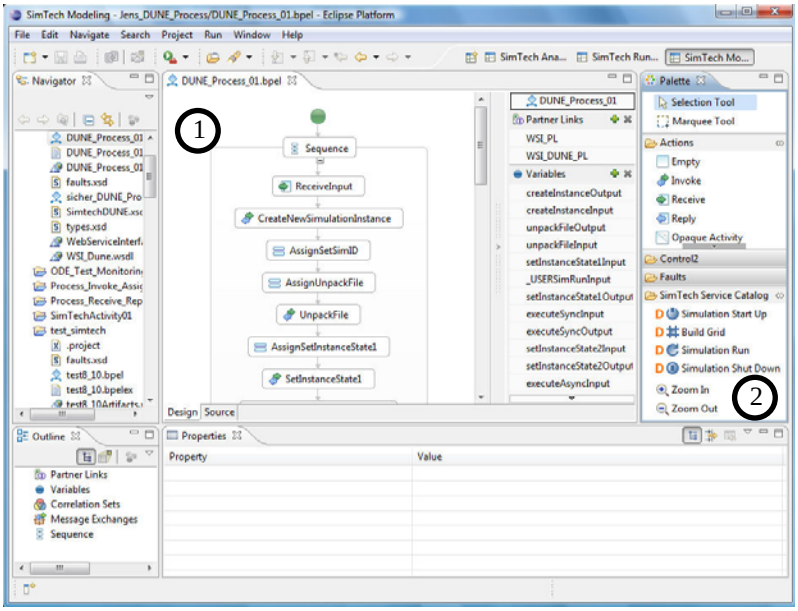



**Figure 3: The SimTech Modeling Perspective in the prototype including a workflow modeler (1) and the service catalog (2).**

The runtime environment of our prototype uses open source software for all components. The execution engine is based on the Apache Orchestration Director Engine[8] (ODE). In order to provide the basic functionality of a service bus we use Apache Axis2[9]. In our prototype Axis2 invokes services that implement workflow activities but extensions are needed for two main objectives: service discovery and resource management. Currently, we provide a service registry for data services based on Apache jUDDI[10] that can be used to choose data sources at runtime.

In order to support resource management functionalities in the service bus we developed a set of generic Web Service interfaces to deal with external data sources, scientific applications, or computing environments. The WS interface to invoke scientific applications consists of a generic adapter that provides fundamental functionality: The adapter manage all instances of the simulation applications and assists basic operations like generate directories, files access, execute supporting services like configuring, compile source code, or start a scientific application without user interaction. To run applications with user interaction or in

[8] http://ode.apache.org/
[9] http://ws.apache.org/axis2/
[10] http://ws.apache.org/juddi/

an asynchronous manner a plug-in that fulfills the special needs of the application is necessary.

The ingredients of the generic adapter are a basic WS, an instance pool, a program manager, and a callback WS. The basic WS is the main interface for a client to communicate with the generic Web Service interface. It offers basic operations to interact with the instances of simulation applications synchronously or asynchronously. The instance pool manages every instance of simulation applications individually. The program manager implements all operations needed to execute simulation applications like create directory structures, install compiler, compile source code, or start executable programs. It provides functionality for applications with or without user interaction. Beyond that, the program manager supports an interface that can handle the program output, e.g. for troubleshooting. The callback WS serves the notification by an asynchronously running simulation application. For example, an application can notify its state (e.g. runnable, busy) to the generic WS interface. To do this, the simulation application must be enriched with a platform specific callback stub.

The WS interface for data sources provides a framework for data access from a BPEL workflow to handle huge amounts of data stored in different sources. The database domain has inspired generic activities like Select, Update, Insert, and Delete. Our prototype currently supports comma separated value files (CSV) and relational databases (SQL).

For the auditing, monitoring, and provenance components we developed an integrated database environment. To store the audit information the environment can use different database systems. The ODE execution engine uses an Apache Derby[11] database management system (DBMS) to store all audit information. As a result of an evaluation we have replaced Derby with the more suitable open source DBMS PostgreSQL[12]. It is also feasible to utilize the commercial IBM DB2[13] DBMS. Using the audit information and based on the management tools of the DBMSs or an external tool like SQuirreL[14] it is possible to extract monitoring information like the status of a workflow or the number of running instances.

## 3. A DUNE-based Simulation – An Example

We implemented several workflows that perform simulations with different complexity to prove the viability of the architecture and prototype of the SimTech WfMS. For complexity reason we demonstrate a simple example: a fluid dynamics problem that simulates the diffusion of an ink drop into a box of water with the help of the finite elements method (FEM). In the scientific experiment, the ink dif-

[11] http://db.apache.org/derby/
[12] http://www.postgresql.org/
[13] http://www-01.ibm.com/software/data/db2/
[14] http://squirrel-sql.sourceforge.net/

fusion in water can be expressed as partial differential equation (PDE) in three spaces and one time dimension with particular conditions:

$$\frac{\partial c}{\partial t} + \nabla \cdot (uc) = 0 \quad in\ \Omega \times T$$

$\Omega \subset \mathbb{R}^3$ is a domain

$T = (0, t_{end})$ is a time interval

$c: \Omega \times T \rightarrow \mathbb{R}$ is the unknown concentration

$u: \Omega \times T \rightarrow \mathbb{R}^3$ is a given velocity field

intitial condition: $c(x, 0) = c_0(x), \qquad x \in \Omega$

boundary condition: $c(x, t) = b(x, t), \qquad t > 0, \qquad x \in \Gamma_{in}(t) = \{y \in \partial\Omega \mid u(y, t) \cdot v(y) < 0\}$

Here $v(x)$ is the unit outer normal at a point y $\partial\Omega$
and $\Gamma_{in}(t)$ is the inflow boundary at time $t$

To solve this PDE we used the FEM framework of DUNE (Distributed and Unified Numerics Environment). DUNE fits a wide range of simulation problems and is therefore frequently used SimTech. For example it contains software libraries that provide modules for solving PDEs. These C++ source code libraries include for example implementations of grid-based techniques such as finite elements. Furthermore, DUNE contains different solver libraries, e.g. for linear equations, for vector and matrix calculations, or iterative solvers. Other libraries support data import and export for various file formats.

In order for the scientists to execute a DUNE-based simulation first the source code for the simulation program must be created for the target runtime platform (e.g., operating system, multi-core support). After compilation the source code of the executable simulation application must be copied into a specific directory structure. In detail, the creation of a typical DUNE-based simulation consists of the following twelve steps:

1. Create a new DUNE module.
2. Define dependencies to other existing DUNE modules.
3. Implement custom modules if needed.
4. Construct the main routine which uses the modules. (Simulation parameters like space dimension must be defined in the main routine or in the modules.)
5. Integrate file importers and exporters for the data.
6. Generate a configuration file specifying the build sequence.
7. Compile required modules and the main routine using the GNU build system[15].
8. Create files with input data for the simulation, e.g. initial grid description.

[15] http://www.gnu.org/software/libtool/manual/automake/

9. Copy the executable code and all needed files in a suitable directory structure.
10. Run the simulation.
11. Store the results.
12. Delete out-dated data, e.g. compiler output.

A DUNE-based application created this way is monolithic and cannot be executed stepwise or with user interaction. To achieve this we enrich the DUNE library at step 4. Hence, step 10 can be divided in subtasks:

a. Initialize MPIHelper[16] if multi-core support is needed
b. Create a basic FEM grid
c. Calculate the FEM grid refinement steps with respect to the boundary values
d. Refine the FEM grid
e. Set initial values and the FEM grid
f. Solve the simulation for one time step
g. Write the (intermediate) result data
h. Repeat subtask f. and g. for all required time steps

Until now a C++ program is created in order to execute this simulation. Since we want to use workflow technology for the execution of this simulation, we modeled two different BPEL processes for two use cases. The first use case executes the DUNE-based application without user interaction. The second use case supports user interaction and therefore requires the integration of a gSOAP[17] server.

In Figure 4 a workflow model for the second use case is represented. It uses a so called WS DUNE plug-in that is an extension of the generic WS interface presented in Section 2.5. At the beginning of the workflow the activity *CreateInstance* invokes the WS DUNE plug-in to initialize a unique Instance-ID and some metadata like timestamps managed by the instance pool.

The activity *Unpack_dune-common* creates all required directories and copy software artifacts like the GNU build system into these directories. Afterwards, *Unpack_dune-grid* inserts the source code of the DUNE C++ framework into the created directories. *Unpack_dune-grid-howto* provides the source code for the concrete simulation to a specified directory. After these preparatory operations *Dunecontrol_All* builds the executable simulation application. This activity combines the generation of modules, code fragments, and files concerning steps 1 – 9. *Execute_finitevolume* invokes starts the interactive simulation application. Thus, we can control the substeps 10a to 10g with BPEL activities to calculate the ink diffusion simulation.

[16] http://www.mpi-forum.org/
[17] http://www.cs.fsu.edu/~engelen/soap.html

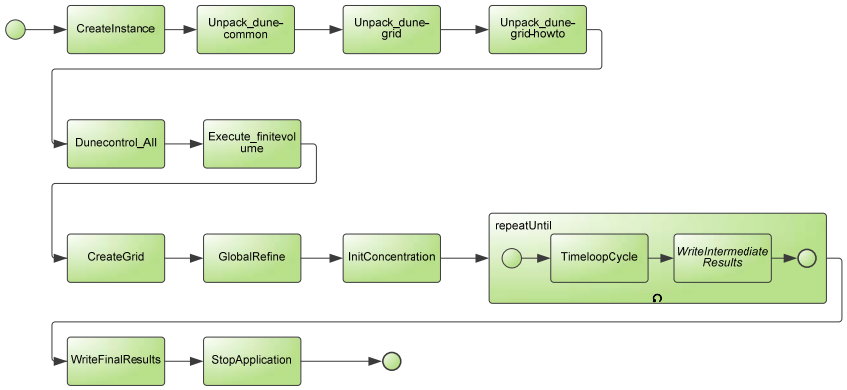



**Figure 4: Workflow to simulate the ink concentration in water.**

To simplify the execution of the use case we use a single core machine without the need for the MPIHelper (substep 10a). Therefore, *CreateGrid* and *GlobalRefine* initialize the simulation at substep 10b or 10c and 10d, respectively. Afterwards, *InitConcentration* (substep 10e) reads initial values from a database and integrates these values in the FEM grid. *TimeloopCycle* represents substep 10f and together with the enclosing *repeatUntil* activity it iteratively calculates single time steps in the simulation. For each time step *WriteIntermediateResults* writes intermediate result data in a database (substep 10g). Activity *WriteFinalResults* that is also part of substep 10g stores the final result data in a database and *StopApplication* finally cleans the workspace of all data fragments that are no longer needed, e.g. intermediate results or compiler output.

# 4. Scientific Domain Intended Extensions for Conventional Workflow Technology

In this section we present extensions of conventional workflow technology intended for the use of workflows in the scientific domain. Especially these adaptations meet unfulfilled requirements established by scientific simulation workflows.

## 4.1. Modeling Language

As starting point for the modeling of scientific workflows we use BPEL [5] since it is the de facto standard for modeling business workflows. In [13] the ability of BPEL to be used as modeling language for scientific workflows is already documented. It should be especially noted that BPEL and its support of transaction models upgrades the robustness of scientific workflows which is one of the goals in our work.

Applying conventional workflow technology for scientific simulations imposes the need for the introduction of new language constructs in order to achieve the

required expressiveness of workflow modeling languages. Existing simulation modeling languages are various since simulations are applied in various fields. Often the languages are close to mathematical representations, e.g. in MatLab. Higher modeling languages are mainly given by simulation libraries written in C++, e.g. DUNE, ODEMx[18]. As mentioned before usability is an important issue in the scientific domain. Therefore, some efforts were made in graphical modeling of simulations, e.g. Simulink[19].

In this section we present some modeling concepts specifically designed for scientific simulation workflows in general and for an increased expressiveness of BPEL for simulations in particular. In [34] the gentle reader can get further information about modeling language constructs for simulation workflows based on BPEL. In comparison to GriCoL, a graphical language for simulation workflows in Grids, we identified data handling and pipelining mechanisms on workflow level as well as different layers of abstraction, explicit data flow, and shared data as core concepts for simulation workflows that are currently not supported by BPEL.

### Data-centric Workflow Models

The data-centric character of scientific workflows demands adaptation of the language. Since BPEL is a control-driven language, control structures that scientists want to use in their workflows are already supported by BPEL. The specification of data-driven workflows is not completely addressed by BPEL until now. An extension, namely BPEL-D [17], exists that allows the modeling of *data dependencies* between activities in the BPEL process. In a pre-deployment step BPEL-D processes are translated into standard BPEL and hence the data dependencies are not used during execution. Since data dependencies in data-driven workflows require a delivery of control as well, BPEL-D is not adequate for the use in scientific workflows. However, it can be used as a starting point. Enabling the modeling of "real" data-driven workflows with BPEL is one future work issue.

The handling of huge amounts of data has impact on the performance during scientific workflow execution. For the most part this challenge must be met by the workflow runtime environment. However, there are possibilities in the modeling language to improve the handling of huge amounts of data. In [18] we presented an improvement of data handling by the introduction of *data references* that is recently implemented in the SimTech prototype (cf. Section 2.5). Dealing with huge amounts of data in BPEL processes results in lots of transmissions of large data sets which make the usage of BPEL very costly for scientific workflows. The majority of scientific data is of no importance for the process logic itself and thus can be ignored by the WfMS. By using data references the transfer of huge amounts of

[18] http://odemx.sourceforge.net/

[19] http://www.mathworks.com/products/simulink/?BB=1

data through the engine can be minimized or even avoided without losing relevant information. Figure 5 illustrates the idea to use data references.

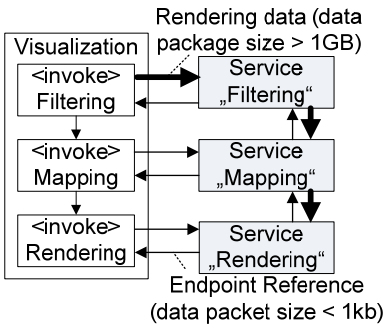



**Figure 5: A visualization workflow using data references.**

The scientific workflow in Figure 5 represents a general visualization process. It successively invokes a filtering service, a mapping service and a rendering service. A workflow engine in the WfMS is responsible for the execution of the visualization workflow and for the invocation of services. A visualization workflow receives a lot of data during the simulation. Consequently the execution of the visualization workflow in Figure 5 demands multiple exchanges of huge data sets through the engine. Usage of pointers in the visualization workflow shifts the responsibility for data transport to participating services and therefore scales down the amount of data that has to be transferred through the workflow engine.

### Simulation Workflow Container

Typically, simulations have complex character and often more than one workflow model is assembled in a so called choreography. Hence, the simulation represents a context for assembled workflows. Since in workflows there are variables that represent the state of the simulated system we need to support shared context data in simulation workflows. Therefore, we propose the introduction of *simulation workflow containers* that hold assembled workflows in choreographies together with shared data structures. Data structures are held in variables typed with XML schema. Different types of values result in different types of variables that hold data structures. State variables and simulation parameters represent shared context data and therefore embodied by reference variables that allow the storage of data values outside of the simulation model. Reference variables are visible for all workflows that are enclosed in the simulation workflow container. For an enhanced context it should generally be possible to use data references in simulation workflows.

When the simulation model is executed by a centralized workflow engine, the data value is held in the local storage of the engine. For distributed engines there is

the need for an optimized distribution of such shared data. At first the data value can be held on any node X and the other nodes send requests for the data whenever it is required. When choosing the node that stores the value the number of data transfers for reference resolution should be considered for an optimized performance in simulation execution. In general, the node with the minimal number of data transfers between all nodes should store the data value.

Some kinds of simulations additionally require synchronization of workflows assembled in a simulation workflow container. This means especially synchronization regarding a model property. A possible use case is the intention to model a time scale whose value and variation is equal in each workflow. Another possible use case is the parallel processing of one data item by multiple workflows or workflow instances.

Simulation workflow containers as stated above can be used as well for multi-scale and multi-physics simulations. In multi-scale simulations one model of one system is simulated on different scales. The simulation on different scales means different modifications of one system property, e.g. the variation of time or a length quantity. The execution of multi-scale simulations depends on the use case. Scientists want to run multi-scale simulations sequentially as well as in parallel. In contrast to multi-scale simulations, multi-physics simulations demand multiple models of one system. Each model represents another point of view on the system with another physical theory as basis. These different models are possibly coupled and executed sequentially or in parallel.

Multi-scale simulations can be realized by one simulation workflow container that holds the system model. The container is parameterized with the scale, i.e. the modification of the particular system property. With that it is possible to simulate multiple scales by running multiple instances of the container. These instances can be executed sequentially or in parallel.

Multi-physics simulation can be realized with workflows by nesting simulation workflow containers. The overall container involves one simulation workflow container for each simulation model. In that case the overall container assembles no simulation workflows while the child containers hold workflow choreographies. The support of coupling mechanisms between simulation models can be realized by message exchanges between simulation workflow containers.

### 4.2. Runtime Environment

The runtime environment offers means to execute workflows. This section discusses workflow execution on distributed and centralized workflow engines and shows how convenient data handling in workflows can be achieved.

#### Distributed vs. Central Execution

The architecture in Figure 2 and our current prototype rely on a centralized workflow engine while the used WSs can be distributed among different machines hosted by different organizations. This is the common approach in both conven-

tional and scientific workflow management. It is a known fact that scientific workflows are often long-running and that they usually process large data sets that have to be transmitted between services and workflow engine [12][14]. When a centralized workflow engine runs many workflow instances in parallel that deal with huge amounts of data (e.g. in a parameter sweep), it can become a bottleneck for the whole system even if the used services are distributed. Additionally, a malfunction of the workflow server means a breakdown of all running simulations until the server restarts. The employment of a decentralized workflow engine can address these and other problems. In our former work we investigated whether a workflow engine that is distributed with the help of a process space-based middleware is beneficial to the scientific domain [22]. The "engine" is a set of activity clients that can be installed on several machines (even at process runtime). They communicate via tokens that are written to and read from process spaces. That way control and data is exchanged. The process spaces themselves can also be installed on several machines. In such an infrastructure the processes can be executed even in the presence of server failures if redundant functionality (i.e. services) and data replication mechanisms are employed. The engine is no single bottleneck anymore. Intelligent distribution of activity clients on machines that host the used services can minimize the network load and hence speedup workflow execution. Of course a distributed workflow enactment has also downsides. Previously unconnected components have to be wired which results in an increased coordination and configuration effort. Furthermore new sources of failures are introduced such as the unavailability of process spaces or tokens that are written but never read because of faulted activity clients. In summary, decentralized workflow systems have many properties that can be advantageous for scientific applications. But it needs to be decided on a case basis if these advantages outbalance the described effort.

### Interaction with Data Stores and Huge Data Sets in Workflows

In the context of scientific and especially simulation workflows the management of data sets is a fundamental issue [23]. Typical challenges are to integrate heterogeneous data sources and to cope with huge amounts of data [21]. As mentioned earlier, we propose to rely on BPEL as starting point for a simulation workflow language. In BPEL, data is loaded into processes with WS invocations. The access to data therefore must be wrapped by WSs. While this functionality is desired in business scenarios, it provides a serious drawback for a practical use in the scientific domain. Scientists want to access data directly from within their workflows [34]. We therefore extended BPEL by special data management activities that can be used to directly access several kinds of external data sources, e.g. data stored in databases or CSV files. No WS wrappers are needed anymore. These activities are geared towards established data management patterns such as query, insert, update, or delete. It is possible to reference queried data with special Set Reference variables and hence leave this data in a data source (i.e. outside of the workflow

engine). Parts of that data can be loaded explicitly into processes if needed for process execution by a retrieve data activity. This avoids stuffing huge amounts of data not needed by a workflow into the BPEL engine. Another aspect of our approach is late binding of data sources. Data sources can be registered with non-functional properties in an extended UDDI registry. Data management activities can then be bound to concrete data sources at process runtime by matching specified requirements on data sources and provided features of registered data sources. This enables an increased robustness of the overall workflow execution in case data sources fail and alternative data sources with similar properties are available.

### 4.3. Flexibility Mechanisms

Quite a lot of research is already done to flexibility of business workflows and conventional workflow technology. Following approaches are especially interesting for scientific workflows. Note that we do not claim the list to be complete. (1) Late binding of services increases flexibility of workflows because concrete services are selected at execution time [26]. (2) The approach of parameterized processes [27], [28] is a BPEL extension to release interacting activities from the specification of service interfaces (port type and operation pairs). That way, services can be late-bound independent of their interfaces. (3) BPEL'n'Aspects [29] is a non-intrusive mechanism to adapt the logic dimension of BPEL process models or instances with the help of the AOP (aspect-oriented programming) techniques [30]. It is possible to extend workflows with additional behavior, to replace or delete elements (activities, transition conditions). Cumbersome instance migration operations [31] are not be needed. (4) The process space-based workflow enactment [32] (see Section 4.2) opens new possibilities for workflow logic adaptations during execution. Workflows can be modified by installing/uninstalling as well as and configuring/re-configuring activity clients.
Scientific workflows in general and simulation workflows in particular impose novel requirements on the workflow meta-model and execution environment. Such requirements are not yet accounted for in the discussed flexibility mechanisms. In the following we sketch considerations about needed extensions in the meta-model and execution environment for simulation workflows.

#### Modeling Language Extensions

The integration of sensors and hence streaming data into workflow imposes the need for a new connector, namely a data pipeline. In contrast to data or control connectors, pipeline edges can be evaluated several times and hence activity instances that are the source or target of a pipeline connection may be executed several times. This imposes new use cases for adaptation of workflows. For example, a concept is needed to exchange sensors or insert/delete sensor activities at runtime. Such a concept will differ very much from exchanging, inserting or deleting activities that invoke services because sensors continuously deliver streams of data

whereas the communication with services is based on sending and receiving messages.

### Execution Environment Extensions

Due to the need of enormous computing and storage capacities simulations are often conducted in Grids. With the introduction of the Web Services Resource Framework (WSRF) [7] Grid resources can be provided as stateful Web Services and can thus be orchestrated in service compositions (e.g. in a BPEL process). Such a stateful environment imposes new requirements on flexibility mechanisms. For example, when changing a running workflow by choosing another service that is to be invoked, it may happen that the service resides on another resource than the data it relies on. This implies that the data needs to be shipped between the resources. This data shipment can be implemented by the ESB transparently for the scientist. Currently, ESBs do not provide data shipping functionality because services used in business scenarios are self-containing logic. However, the specification of a data dependency between resources/services/tools would simplify the solution of this problem. Late binding and function shipping mechanisms could render the situation even more complex.

An important requirement of scientists on an execution environment for simulations is the reproducibility of results. A sWfMS with advanced flexibility capabilities therefore needs a mechanism to track (automatic or manual) changes that were made during execution of workflows. Tracking changes is currently not implemented by the presented approaches to flexibility. Another reason for the need of advanced flexibility features in scientific workflow management is the way scientists create and execute their experiments and simulations. They usually do not distinguish between models and instances of workflows. That means the focus of their work is on single workflow instances. Scientists often unconsciously switch between phases of modeling, execution and monitoring of workflows. They are unaware of the technical realization of their actions and in fact do not want to cope with such details. However, from a software engineering point of view it is desirable to consider these technical aspects. We therefore extended and modified the business workflow life cycle to reflect the needs of scientists (see Figure 6) [33]. This enhanced life cycle reveals that scientists transparently make use of two categories of adaptation operations when manually modifying their simulation workflows. First, adapting the structure of workflows (i.e. the logic dimension) entails a redeployment of the workflow or parts of it. Of course, deployment should be transparent for the scientists. It is nevertheless mentioned here because of its importance from a technical point of view [15]. Second, modifications on the functions dimension (e.g. changing criteria for selection of a service) can be conducted without a need for redeployment. In order to support scientists in developing simulations and experiments in the described trial-and-error manner, a tool is needed that combines the capabilities of a workflow modeling tool and a progress monitor. That means it allows modeling a workflow, executing it, and modifying it at runtime. Such a tool implements the proposed blending of model-

ing, execution, adaptation, and monitoring phases of the life cycle of scientific workflows. A major part of the solution is to execute incomplete workflows (or workflow fragments) in a straight forward manner. A finished workflow fragment execution must not be considered completed. It must be possible to continue modeling of the workflow and to resume its execution. In conventional workflow technology, deploying and executing workflow fragments is currently an unsolved issue.

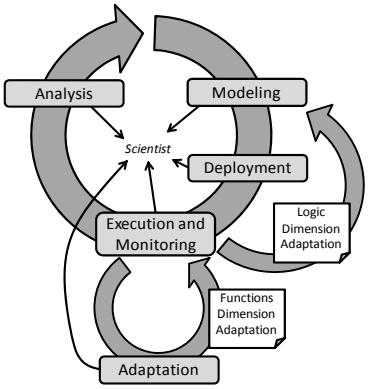



**Figure 6: Life cycle of scientific workflows geared towards the technical realization with the help of conventional workflow technology.**

# 5. Conclusion

In this chapter we discussed the qualification of established workflow technology for the evolving application domain of e-Science as well as the state of the art of scientific workflow management systems. We discussed in detail the qualification of conventional workflow technology for e-Science based on the most important requirements of scientific workflows and simulations workflows in particular. We outlined benefits as well as arising technical problems and gave recommendations as of how to solve these problems. Furthermore, by means of the life cycles for business and scientific workflows we explained that conventional workflow technology needs thorough extensions in order to be suitable for application in scientific experiments, computations and simulations. We presented a workflow system architecture that is based on conventional workflow technology and tailored to the needs of scientists and explained how the foreseen system meets general requirements of scientists and scientific workflows. Further, a prototype that implements the architecture and is especially designed to support the modeling and execution of simulation workflows was introduces. An example scenario of “ink diffusion in water” that runs on the prototype demonstrate the feasibility of the approach.

Because of the data-centric character of scientific workflows conventional workflow modeling languages (e.g. BPEL) need adaptations when being applied

in the scientific domain. In order to efficiently handle huge amounts of data in scientific workflows we introduced data references in BPEL and conventional workflow technology in general. Furthermore, we draw on an existing approach to integrate explicit data flow in BPEL since modeling languages for scientific workflows have to support explicit data flow modeling constructs. Prospectively, conventional workflow languages need to be adapted in order to handle data streams. Scientists want to divide large data sets in components and process a stream of these components in a scientific workflow. Furthermore, scientists want to integrate sensor data in scientific workflows which also establishes the need for stream data in the modeling language for scientific workflows. Especially for simulations we introduced the modeling concept of simulation workflow containers holding an assembly of simulation workflows and shared context data. In future this concept will be implemented and integrated in the presented prototype. Nevertheless, for simulation conventional workflow languages need further extensions, e.g. in order to efficiently handle parameter searches. In addition we will pay special attention on further development of choreographies for simulation since they represent eligible matches (cf. for example [20]).

In the runtime environment we primarily intend to extend our approach to integrate data and process provenance for scientific workflows. The starting point is the presented integrated database environment for workflow and simulation data based on a special Web Services interface. Furthermore, we plan to develop methods for performance optimization especially by optimizing the global resource utilization in scientific workflows.

Furthermore, special attention will be paid on flexibility aspects. This promises both an improved robustness of the system as well as an explorative workflow modeling for scientists (i.e. trial-and-error modeling). Mechanisms to flexibility have impact on modeling and execution of workflows. Since we want to pursue an engineering solution, we want to make use of existing flexibility approaches. We therefore sketched existing candidates and argued that these need extensions in order to satisfy requirements of scientific and simulation workflows. Currently, we are working on the blending of modeling, execution, and monitoring phases of workflows to support the trial-and-error approach of workflow development of scientists. The first prototype will allow changing of parameters at runtime. As next step, we want to integrate the BPEL'n'Aspects approach into the prototype in order to provide more complex change operations. A challenge will be the development of a generic format for tracking changes, obtaining information about automatic and manual modifications in order to allow reproducibility and confidence in simulation results, and a method to visualize changes in a practical way.

## References

[1] F. Leymann, D. Roller: *Production Workflow: Concepts and Techniques*. Prentice Hall, Englewood Cliffs, NJ, 1999.

[2] I.J. Taylor, E. Deelman, E.B. Gannon, M. Shields (ed.): *Workflows for e-Science - Scientific Workflows for Grids*. Springer, 2007.
[3] G. Decker, O. Kopp, F. Leymann, M. Weske: Interacting services: from specification to execution. In: Data & Knowledge Engineering. Vol. 68(10), Elsevier Science Publishers, 2009.
[4] R. Chinnici, J.-J. Moreau, A. Ryman, S. Weerawarana: *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. 2007.
[5] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, F. Curbera, M. Ford, Y. Goland, A. Guízar, N. Kartha, C. K. Liu, R. Khalaf, D. König, M. Marin, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, A. Yiu: *Web Services Business Process Execution Language Version 2.0*. 2007.
[6] D. Churches, G. Gombas, A. Harrison, J. Maassen, C. Robinson, M. Shields, I. Taylor, I. Wang: *Programming Scientific and Distributed Workflow with Triana Services*. Concurrency and Computation: Practice and Experience. Special Issue on Scientific Workflows, 2005.
[7] S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, I. Sedukhin: *Web Services Resource (WS-Resource) V1.2*. OASIS, 9 December 2004.
[8] T. Oinn, M. Greenwood, M. Addis, M. Nedim Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat, C. Wroe: *Taverna: Lessons in Creating a Workflow Environment for the Life Sciences*. Concurrency and Computation: Practice and Experience 2006, 18(10):1067-110.
[9] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, S. Mock: *Kepler: An Extensible System for Design and Execution of Scientific Workflows.* SSDBM, 2004.
[10] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny: *Pegasus: Mapping Scientific Workflows onto the Grid*. Lecture Notes in Computer Science, Volume 3165/2004, Second European AcrossGrids Conference, Springer, 2004, pp. 11-20.
[11] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, Y. Simmhan: *The Trident Scientific Workflow Workbench*. In: IEEE eScience Conference, 2008.
[12] R. Barga, D. Gannon: *Scientific versus Business Workflows*. In: [2], 2007.
[13] A. Akram, D. Meredith, R. Allan: *Evaluation of BPEL to Scientific Workflows*. In Cluster Computing and the Grid (CCGrid), pages 269–274. IEEE Computer Society, 2006.
[14] B. Ludäscher, M. Weske, T. McPhillips, S. Bowers: *Scientific Workflows: Business as Usual?* 7th Intl. Conf. on Business Process Management (BPM), LNCS 5701, Ulm, Germany, 2009.
[15] M. Sonntag, D. Karastoyanova, F. Leymann: *The Missing Features of Workflow Systems for Scientific Computations.* In: Proceedings of the 3rd Grid Workflow Workshop (GWW) (to appear), 2010.
[16] L. Moreau, B. Clifford, J. Freire, Y. Gil, P. Groth, J. Futrelle, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, *The Open Provenance Model Core Specification (V1. 1)*. Future Generation Computer Systems, 2009.

[17] R. Khalaf: *Supporting business process fragmentation while maintaining operational semantics: a BPEL perspective*. PhD thesis, University of Stuttgart. 2008.
[18] M. Wieland, K. Görlach, D. Schumm, F. Leymann: *Towards Reference Passing in Web Service and Workflow-based Applications.* Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009). 109-118 (2009).
[19] B. Wassermann, W. Emmerich, B. Butchart, N.Cameron, L. Chen, J. Patel: *Sedna: A BPEL-Based Environment for Visual Scientific Workflow Modeling*. In: [2], 2007.
[20] A. Barker, P. Besana, D. Robertson, J. Weissman: *The Benefits Of Service Choreography For Data-Intensive Computing.* In: Proceedings of the 7th International Workshop on Challenges of Large Applications in Distributed Environments (CLADE'09), in conjunction with HPDC'09: The 18th International Symposium on High Performance Distributed Computing, pages 1-10. ACM, 2009.
[21] D. Pennington, D. Higgins, A.T. Peterson, M.B. Jones, B. Ludäscher, S. Bowers: Ecological niche modeling using the Kepler workflow system. In: [2], 2007.
[22] M. Sonntag, K. Görlach, D. Karastoyanova, F. Leymann, M. Reiter: *Process Space-based Scientific Workflow Enactment*. In: International Journal of Business Process Integration and Management (IJBPIM) Special Issue on Scientific Workflows (to appear), Inderscience Publishers, 2010.
[23] E. Deelman, A. Chervenak: *Data Management Challenges of Data Intensive Scientific Workflows*. Proc. IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID '08), pp.687-692, 2008.
[24] N. Currle-Linde, P. Adamidis, M. Resch, F. Bös, J. Pleiss: *GriCoL: A language for scientific grids.* In: Proceedings of the 2nd IEEE International Conf. on e-Science and Grid Computing, 2006.
[25] D. Karastoyanova, F. Leymann: *Making scientific applications on the Grid reliable through flexibility approaches borrowed from service compositions.* In: N. Antonopoulos, G. Exarchakos, A. Liotta (Eds.), Handbook of research on P2P and Grid systems for service-oriented computing: Models, methodologies and applications (Information Science Publishing, 2010).
[26] S. Weerawarana, F. Curbera, F. Leymann, D.F. Ferguson, T. Storey: *Web Services Platform Architecture: Soap, WSDL, WS-Policy, WS-Addressing, WS-Bpel, WS-Reliable Messaging and More*. Prentice Hall, 2005
[27] D. Karastoyanova: *Enhancing flexibility and reusability of web service flows through parameterization*. PhD thesis, TU Darmstadt and University of Stuttgart, 2006.
[28] D. Karastoyanova, F. Leymann, A.P. Buchmann: An Approach to Parameterizing Web Service Flows. In: B. Benatallah, F. Casati, P. Traverso (Eds.): Proc. 3rd Intl. Conf. on Service Oriented Computing (ICSOC'2005).
[29] D. Karastoyanova, F. Leymann: *BPEL'n'Aspects: Adapting Service Orchestration Logic.* In: Proceedings of the 7th International Conference on Web Services (ICWS 2009)
[30] G. Kiczales: *Aspect-Oriented Programming*. In: Proceedings of ECOOP'97, Finland, 1997.

[31] B. Weber, S. Rinderle, M. Reichert: *Change Patterns and Change Support Features in Process-Aware Information Systems*. In: Proceedings of Conference on Advanced Information Systems Engineering (CAiSE), 2007.
[32] D. Martin, D. Wutke, F. Leymann: *A Novel Approach to Decentralized Workflow Enactment*. In: Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference (EDOC 2008), 2008.
[33] M. Sonntag, D. Karastoyanova: *Next Generation Interactive Scientific Experimenting Based on the Workflow Technology*. In: Proceedings of the 21st IASTED International Conference Modelling and Simulation (MS 2010), 2010.
[34] M. Sonntag, K. Görlach, D. Karastoyanova: *Towards Simulation Workflows With BPEL: Deriving Missing Features From GriCoL*. In: Proceedings of the 21st IASTED International Conference Modelling and Simulation (MS 2010), 2010.

# Index