Peter Reimann [b] · Michael Reiter [a] · Holger Schwarz [b] · Dimka Karastoyanova [a] · Frank Leymann [a]

SIMPL – A Framework for Accessing External Data in Simulation Workflows

Stuttgart, March 2011

[a] Institute of Architecture of Application Systems (IAAS)
University of Stuttgart,
Universitaetsstrasse 38
70569 Stuttgart, Germany
Firstname.Lastname@iaas.uni-stuttgart.de
www.iaas.uni-stuttgart.de

[b] Institute of Parallel and Distributed Systems (IPVS)
University of Stuttgart,
Universitaetsstrasse 38
70569 Stuttgart, Germany
Firstname.Lastname@ipvs.uni-stuttgart.de
www.ipvs.uni-stuttgart.de

Abstract Adequate data management and data provisioning are among the most important topics to cope with the information explosion intrinsically associated with simulation applications. Today, data exchange with and between simulation applications is mainly accomplished in a file-style manner. These files show proprietary formats and have to be transformed according to the specific needs of simulation applications. Lots of effort has to be spent to find appropriate data sources and to specify and implement data transformations. In this paper, we present SIMPL – an extensible framework that provides a generic and consolidated abstraction for data management and data provisioning in simulation workflows. We introduce extensions to workflow languages and show how they are used to model the data provisioning for simulation workflows based on data management patterns. Furthermore, we show how the framework supports a uniform access to arbitrary external data in such workflows. This removes the burden from engineers and scientists to specify low-level details of data management for their simulation applications and thus boosts their productivity.

Keywords Data Provisioning; Workflow; Scientific Workflow; Simulation Workflow; SIMPL

SIMPL – A Framework for Accessing External Data in Simulation Workflows

Peter Reimann[2], Michael Reiter[1], Holger Schwarz[2], Dimka Karastoyanova[1], and Frank Leymann[1]

[1] Institute of Architecture of Application Systems, University of Stuttgart
Firstname.Lastname@iaas.uni-stuttgart.de
[2] Institute of Parallel and Distributed Systems, University of Stuttgart
Firstname.Lastname@ipvs.uni-stuttgart.de

Abstract: Adequate data management and data provisioning are among the most important topics to cope with the information explosion intrinsically associated with simulation applications. Today, data exchange with and between simulation applications is mainly accomplished in a file-style manner. These files show proprietary formats and have to be transformed according to the specific needs of simulation applications. Lots of effort has to be spent to find appropriate data sources and to specify and implement data transformations. In this paper, we present SIMPL – an extensible framework that provides a generic and consolidated abstraction for data management and data provisioning in simulation workflows. We introduce extensions to workflow languages and show how they are used to model the data provisioning for simulation workflows based on data management patterns. Furthermore, we show how the framework supports a uniform access to arbitrary external data in such workflows. This removes the burden from engineers and scientists to specify low-level details of data management for their simulation applications and thus boosts their productivity.

1 Introduction

Workflows have long been used to meet the needs of IT support for business processes. Workflows are compositions of tasks by means of causal or data dependencies that are carried out on a computer using a workflow management system (WfMS) [LR99]. Recently, workflow technology has found application in the area of scientific computing and simulations for implementing complex scientific applications and the term *scientific workflow* has been coined [TDG07]. Simulations, as a subset of scientific applications, are typically compositions of complex calculations and data management tasks, which makes them good candidates for the realization as workflows. For instance, partial differential equations have to be solved to determine temporal or spatial changes of simulated objects, e.g., of the structure of a car in a crash test.

Accessing and provisioning huge amounts of heterogeneous and distributed input data as well as generating huge intermediate and final data sets are some of the major challenges of simulation workflows [TDG07][Gi07][DC08]. Typical data management activities in simulation workflows are extraction, transformation, and load operations (ETL) [Mü10].

In [Vr07], the authors discuss workflow technology as the key technology to cope with heterogeneous applications and data stores. In line with this argumentation and as proposed by [Ma05], our work is based on an ETL workflow approach, i.e., ETL operations of simulation workflows are modeled and executed via workflow technology.

Today, the data management and data provisioning of simulation applications is mainly accomplished in a file-style manner. These files show proprietary formats and inevitably have to be transformed into the appropriate format the simulations require. Most of current scientific workflow management systems (sWfMSs) lack a generic, consolidated, and integrated data management abstraction that can cope with huge and heterogeneous data sets. They use several specialized technologies, e.g., custom workflow activities or services, to access data. Lots of effort must be spent to find appropriate data sources and to specify and implement necessary data transformations, which brings in additional complexity for scientists. This is in particular true for simulations involving multiple domains since each domain has its own requirements and solutions for data handling and thus render the data source and application environment even more heterogeneous. A consolidated abstraction support would remove the burden from engineers and scientists to specify low-level details of data management for their simulation applications.

In this paper, we present SIMPL (SimTech – Information Management, Processes, and Languages) – an extensible framework that addresses the lack of abstraction and generality for data provisioning in current simulation workflow technology. SIMPL provides unified access methods to access arbitrary external data in simulation workflows while metadata describe the mappings between their interfaces and the concrete access mechanisms. At the modeling level, the framework extends the workflow language by a small set of activities that tightly embed data management operations for any kind of data source. When such an activity is executed, it uses the unified access methods of SIMPL to seamlessly access the specified data source. To further assist the workflow modeler in defining typical data management tasks in simulation workflows, we introduce data management patterns, e.g., patterns for ETL operations. In this paper, we show that these patterns in combination with the activities for data management and the unified access methods allow to define the data provisioning for simulations in multiple domains as well as for other scientific applications, such as biology, astronomy, or earthquake science. We discuss the extensibility of the SIMPL framework with respect to additional kinds of data sources and data management patterns. Furthermore, we illustrate the huge potential for a consolidated optimization that SIMPL makes possible as it combines the definition of activities for data management and simulation at the same level of abstraction.

The rest of this paper is organized as follows: Section 2 illustrates the motivation to enhance an existing architecture of sWfMSs by the SIMPL framework and shows its integration into this architecture. Afterwards, Section 3 provides details on major aspects of modeling data management tasks in simulation workflows, while Section 4 deals with the underlying approach to unify heterogeneous access mechanisms for different data sources. We then discuss the benefits and drawbacks of our framework and evaluate it via an example simulation workflow in Section 5. Related work is afterwards discussed in Section 6. Finally, Section 7 concludes and lists future work.

2 The SIMPL Framework

The SIMPL framework is designed as an extension to scientific workflow management systems. Hence, we first sketch the main components of such a system according to the architecture of sWfMSs introduced in [Gö11]. Afterwards, we discuss the motivation to enhance this architecture, illustrate the main aspects by means of a sample workflow, and show the architectural integration of the SIMPL framework.

2.1 Scientific Workflow Management Systems

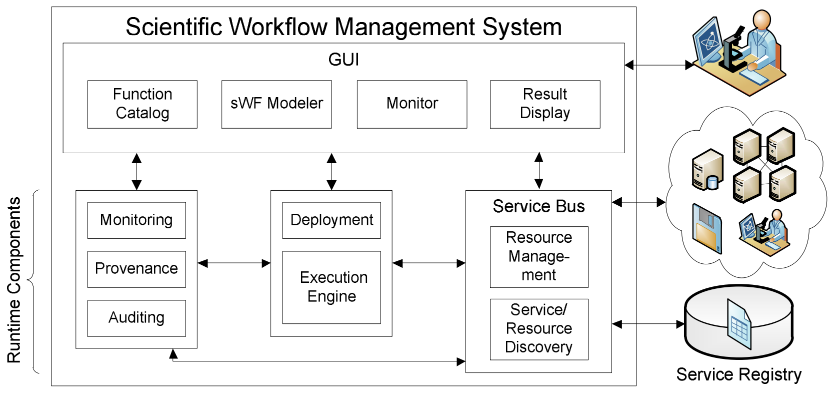


Fig. 1. Architecture of a scientific workflow management system, cf. [Gö11]

The architecture of scientific workflow management systems presented in Figure 1 is based on the workflow technology for business and production workflows as defined in [LR99]. The scientific workflow modeler (*sWF Modeler*) of the GUI supports the modeling of workflow specifications and corresponding deployment information. The *function catalog* provides a list of available services as well as a customizable set of easy-to-model functions that can be used in workflow models. With the help of the *monitor* component, users may constantly observe workflow executions and identify unexpected events or faults. The *result display* component presents the final outcome of simulations as well as intermediate results in a way appropriate for the user.

The *deployment* component transforms workflow models into engine-internal representations and installs them on the *execution engine* that executes instances of these workflows. The *auditing* component records runtime events related to workflows and activities, e.g., the start time of a workflow run. The *monitoring* component uses these events and indicates the states of workflow runs. The *provenance* component records data that goes beyond simple auditing information and that enables the reproducibility of workflow executions. The *service bus* primarily discovers and selects services that implement workflow activities, routes messages, and transforms data. Besides that, it connects workflows to other external, usually stateful resources, e.g., to data sources. The *resource management* component maintains metadata for such external resources as well as for services. The *service/resource discovery* component queries this metadata or external registries to find a list of candidate services or resources by means of descriptive information, e.g., semantic annotations. This list may be used by the function catalog of the GUI, for late binding of services and resources, or for rebinding of failed activities. This naturally implies the ability to use the modeling tool during the execution of workflow instances to enable ad-hoc changes of workflows [SK10].

In this architecture, scientific workflows may access and handle huge, heterogeneous, and distributed data objects, e.g., via services. However, the challenge still remains to provide a consolidated and integrated data management abstraction that is able to deal with such data objects. This abstraction support is one of the key requirements for scientific workflow management [TDG07][Gi07][DC08]. In the following, we illustrate this challenge using a bone remodeling simulation workflow.

2.2 Simulation Workflow for Bone Remodeling

Figure 2 shows the activities and relevant input and output data of a workflow for a bone remodeling simulation (BRS) that is used to research skeletal disorders, e.g., of human femur. The PANDAS framework calculates the structure of a bone under a specific load using the finite element method (FEM) [KME10]. The workflow is divided into three phases: preprocessing, solving, and post-processing.

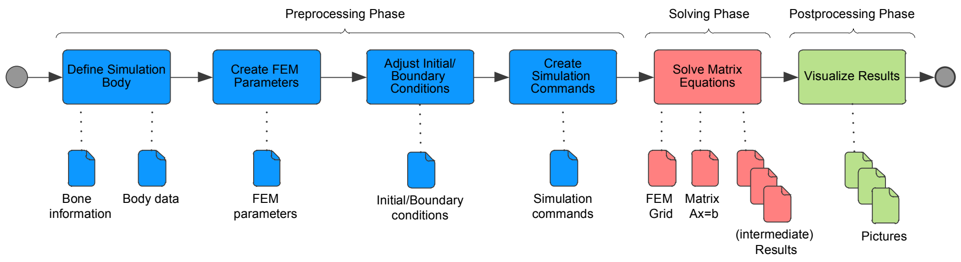


Fig. 2. Workflow for bone remodeling simulation

In the *preprocessing* phase, it starts by loading basic information about the bone to be simulated from different databases or file systems. Examples of this information are a bone structure and material parameters. The second activity extracts FEM parameters from a file, e.g., interpolation functions. Afterwards, the workflow adjusts initial conditions that configure the bone structure for the start time of the simulation. Furthermore, it defines boundary conditions, e.g., the time-dependent pressures from outside on the upper joint of the bone that correspond to the human way of moving. The last preprocessing activity writes a set of simulation commands to a file. For example, it chooses a matrix solver and defines the discretization of the continuous simulation time into n time steps t_1 to t_n. In practice, a simulation involves thousands of such time steps.

In the *solving* phase, the workflow uses the input to create and solve matrix equations for generating the intermediate and final results of the simulation. For each time step t_i, it creates an FEM grid that is the basis to set up matrix equations $\mathbf{Ax} = \mathbf{b}$ that are then solved. The FEM grid contains thousands or millions of mesh points and their relations. This mesh information is typically stored in main memory, but may also be persisted into files or databases for further usage in the post-processing phase. The latter also holds for the matrix **A** and the vectors **x** and **b**. The solving phase ends after time step t_n. The workflow then stores intermediate and final results based on the vectors **x** in comma separated value (CSV) files. The *post-processing* phase transforms these CSV files into another file format suitable for visualization tools.

Altogether, the workflow carries out a multiplicity of data management and data provisioning activities. These activities involve several huge data sets as well as heterogeneous data sources and data formats, e.g., databases, CSV files, unstructured text documents, and image files. Most of the data management operations are performed as manual tasks, implying a high error rate. A generic and consolidated data management abstraction would decrease this error rate. Furthermore, it would remove the burden from scientists to specify low-level details of data management.

2.3 Architecture and main Components of SIMPL

Figure 3 shows how the SIMPL framework extends a sWfMS to provide an abstraction for data management and data provisioning. For better readability, we leave out components of the sWfMS architecture that are not relevant for SIMPL. The *SIMPL core* component, embedded in the service bus, provides unified logical interfaces to any kind of data source. We enhance the resource management component with metadata that describe the mappings between these unified interfaces and the concrete and possibly heterogeneous access mechanisms. The *data management (DM) activity modeling plug-in* of the sWF modeler and the *DM activity execution plug-in* of the execution engine provide data management activities for simulation workflows. These activities may either be directly used in simulation workflows or they may be part of separate ETL workflows that encapsulate data provisioning processes for simulation workflows. The *DM pattern plug-in* of the function catalog assists the workflow modeler in defining the necessary data management operations. It contains abstract data management patterns that allow to model typical data provisioning tasks for simulation workflows. The following sections discuss the SIMPL components and plug-ins in detail.

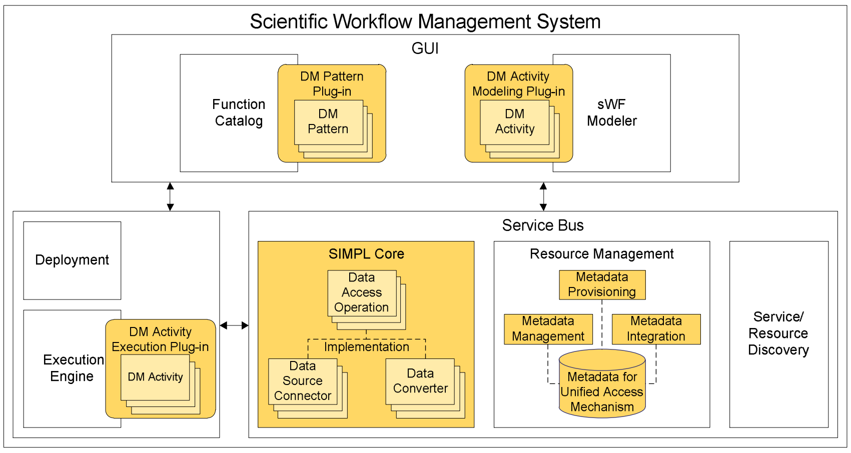


Fig. 3. The SIMPL framework integrated into a sWfMS architecture

3. Modeling Data Management for Simulation Workflows

In this section, we deal with major aspects of modeling data management tasks in simulation workflows. We introduce various extensions to workflow languages that allow for the definition of these tasks. The DM activity modeling plug-in makes these extensions available to the workflow modeler, whereas the DM activity execution plug-in covers their runtime behavior. Furthermore, we show how data management patterns facilitate the definition of data management tasks for simulation workflows.

3.1 Workflow Language Extensions for Data Management

The Business Process Execution Language (BPEL) [Oa07] is the de-facto standard to define and execute business processes based on the control-flow oriented orchestration of service interactions. In [AMA06], BPEL is recommended for modeling and executing scientific workflows and simulation workflows. The main benefits stated are its modular design, its flexibility regarding generic XML data types and late binding of services as well as the fault, compensation, and event handling capabilities. In addition, many BPEL engines offer further capabilities, such as user interaction, workflow monitoring, or recovery of workflows. Due to these benefits of BPEL and in line with previous work, we define the Business Process Execution Language extension for Data Management (BPEL-DM) that extends BPEL by further activity types. We call activities of these new types *data management (DM) activities*. They reflect workflow tasks with embedded data management operations that are seamlessly issued against data sources. The major activity types of BPEL-DM are: *IssueCommand*, *RetrieveData*, and *WriteDataBack*. Each of these activities calls the SIMPL core and sends the data management operation to it in order to deal with heterogeneous data source access mechanisms.

In the following, we use the term *data source* for a system that stores and manages data, e.g., a database or a file system. A data source receives and executes *DM command*s. Examples are SQL statements, shell commands of operating systems, or paths to files. The latter are used to load the content of a file into the process context of the workflow. Each of the DM activities has a BPEL variable as input parameter referring to the data source that executes the embedded DM command. We name such BPEL variables *data source reference variables*. A reference is a *logical data source descriptor* that is either a logical name or a document describing some functional or non-functional requirements for a data source. A logical name describes exactly one data source that is associated with the name in the resource management component. A requirements description can be used for choosing and binding a data source at runtime.

A data source manages several *data containers*. Each container is an identifiable collection of data, e.g., a table in a database system or a file in a file system. *Data container reference variables* refer to a data container via a logical name. The resource management component maps this name to a concrete locator that uniquely identifies the container within the data source. A *data set variable* acts as target container for loading data into the process context of a workflow. Appropriate XML schema definitions specify the contents of these variables and must cope with the differences between several kinds of data sources. For example, we use an *XML RowSet* structure for any table-oriented data, such as data from an SQL database or from a CSV-based file. XML database systems, as another example, may already provide certain XML schema definitions or they may need to store arbitrary XML data within BPEL variables.

We now detail on the three DM activity types. The *IssueCommand* activity can be used for data manipulation or data definition, for example. Besides the data source reference, it has a DM command as additional input parameter and issues this command against the specified data source. The engine that executes the activity expects a notification whether the DM command has been executed successfully by the data source or not. After a notification of success, the engine continues workflow execution according to the specified control flow. In case of a failure, it enables fault handling mechanisms.

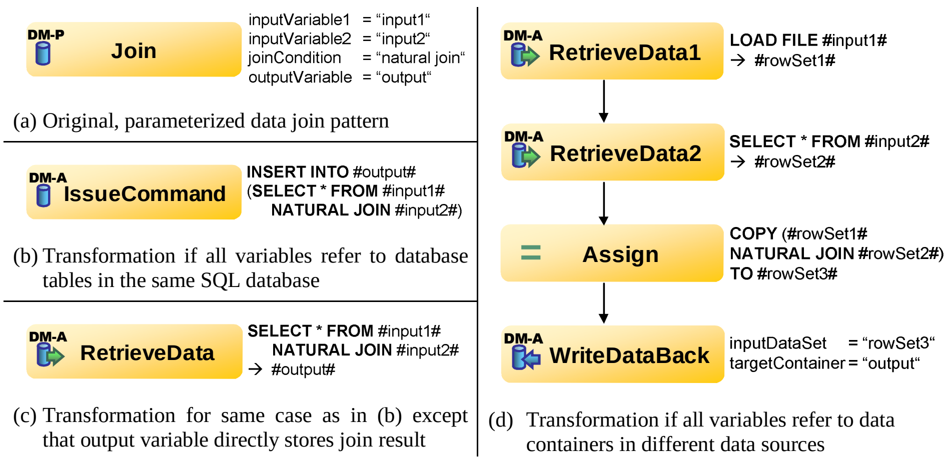
The *RetrieveData* activity also has a DM command as input parameter that is passed to the data source. The DM command must produce data. For instance, it may be a SELECT statement or a path to a file. When the data source has executed the command successfully, the result data is transmitted back to the execution engine. An additional input parameter of the activity defines a data set variable that stores this result data. In case of a failure, the execution engine is notified and enables fault handling mechanisms.

The *WriteDataBack* activity is the counterpart of the *RetrieveData* activity. It writes data from the process context of a workflow back to a data source. The activity accepts one identifier for a data set variable and one for a data container reference variable as input parameters. It stores the data set of the first variable in the data container referred by the second one. As a result, the execution engine gets a notification of success or failure and proceeds like in the case of the *IssueCommand* activity.

Data container reference variables may furthermore be used as parameters in DM commands of the *IssueCommand* or *RetrieveData* activities, e.g., in the FROM clause of an SQL SELECT statement. The same holds for other BPEL variables, e.g., string or integer variables used for comparisons in predicates. The workflow execution engine resolves all these variables, i.e., it reads the variable value and inserts this value at the position within the command where the variable has been referenced beforehand. In order to identify a variable in a DM command and distinguish it from other command items, the variable is marked by surrounding hash marks (e.g., '#'). Regarding a data container reference variable, only the logical name of the data container is inserted in the command. The SIMPL core is later responsible for mapping this name to the data source-specific container identifier by querying the resource management component.

3.2 Abstraction Support through Data Management Patterns

The SIMPL framework provides a set of data management patterns that cover major data provisioning tasks for simulation workflows. The workflow designer picks appropriate patterns from a list provided by the DM pattern plug-in of the function catalog. He/she is then assisted in defining the concrete data management operation for each chosen pattern in a semi-automatic approach instead of defining all details of the operation on his/her own. In the following, we illustrate the pattern-based approach via an example.



(a) Original, parameterized data join pattern

(b) Transformation if all variables refer to database tables in the same SQL database

(c) Transformation for same case as in (b) except that output variable directly stores join result

(d) Transformation if all variables refer to data containers in different data sources

Fig. 4. Data join pattern and its transformation into executable workflow specifications

Figure 4(a) shows a pattern that represents a join of two data sets. Instead of defining concrete data management operations that execute the join, the modeler only needs to set some parameter values, i.e., two input variables, one output variable, and a join condition. Each of the input or output variables may hold data within the process context of the workflow, e.g., via a data set variable of BPEL-DM. Another option is that they refer to external data, e.g., via a data container reference variable. The respective data sets of the two input variables are joined according to the specified join condition. The result of this join is stored in the output variable or in the data container it refers to.

A set of rewrite rules specifies the transformations of abstract and parameterized patterns into workflow parts that carry out the necessary data management operations, e.g., via DM activities of BPEL-DM. The given parameters of the patterns and metadata that describe the characteristics of the data sources to be accessed, e.g., their query capabilities, determine which rewrite rule is to be applied for a certain pattern. Figure 4 shows three rewrite rules for our join example. If the two input variables and the output variable of the join pattern refer to database tables in one and the same SQL database, an *IssueCommand* activity of BPEL-DM with an embedded set-oriented INSERT statement may execute the join (Figure 4(b)). In case the output variable directly stores the join result, we use a *RetrieveData* activity with a SELECT statement (Figure 4(c)). If all three variables refer to data containers in different data sources, the transformation becomes more complex. Assume that we need to perform a join between the content of a CSV-based file and a relational database table and that another database table is the target container for the join result. Then, we may use two *RetrievaData* activities that load the contents of the two input data containers into the process context of the workflow. A subsequent BPEL assign activity joins them, and a *WriteDataBack* activity stores the join result into the target database table (Figure 4(d)).

As described above, metadata about the data sources to be accessed are one basis for deciding on the rewrite rule to be applied for a certain pattern. Hence, we must not apply rewrite rules until it is clear which data sources the data management tasks need to access. In case of a static data source binding during deployment time, we apply rewrite rules shortly before this deployment phase. If data sources are bound at runtime, we will convert each pattern into a single process fragment and use process fragment technology to dynamically integrate this fragment into an already executing workflow [EUL09].

Besides our join example, the DM pattern plug-in contains further patterns and according rewrite rules for typical data provisioning tasks of simulation workflows. This covers patterns for the transmission of data from one resource to another or for ETL operations [Mü10][TDG07]. ETL operations may be loading or retrieving a bulk of data, filtering a data set as well as joining, merging, or normalizing two data sets.

4 Unified Access Mechanism

Now, we illustrate the approach to unify different kinds of data source access mechanisms. This includes the SIMPL core and its unified logical interfaces to data sources, the metadata to map these interfaces to the underlying access mechanisms, and the interaction between the components of the service bus during data source access.

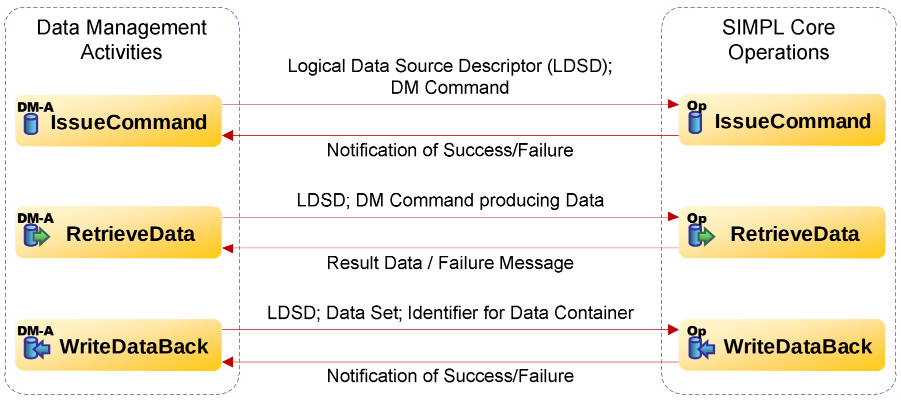

Data Management Activities
SIMPL Core Operations
Logical Data Source Descriptor (LDSD); DM Command
DM-A IssueCommand
Op IssueCommand
Notification of Success/Failure
LDSD; DM Command producing Data
DM-A RetrieveData
Op RetrieveData
Result Data / Failure Message
LDSD; Data Set; Identifier for Data Container
DM-A WriteDataBack
Op WriteDataBack
Notification of Success/Failure

Fig. 5. Data sent between DM activities and SIMPL core operations

4.1 SIMPL Core

The SIMPL core defines a set of generic operations to access arbitrary data sources, i.e., the specifications of the operations are independent of the underlying kinds of data sources. They are geared to the DM activities of BPEL-DM and named accordingly: *IssueCommand*, *RetrieveData*, and *WriteDataBack*. Each DM activity calls the SIMPL core operation that shares the name of the activity. Figure 5 shows which contents of the input parameters of each activity are sent from the workflow execution engine to the corresponding SIMPL core operation and which message or data the activity expects as a reply. Regarding the interaction between workflows and data sources, the SIMPL core operations only forward DM commands, result data, or notifications. They do not implement any complex data transformations or analyses as this would contradict our assumption of workflow activities seamlessly accessing data sources.

Each SIMPL core operation expects a logical data source descriptor as input in order to identify the data source where the data management operation is to be executed. The *IssueCommand* operation gets a DM command as further input, and the *RetrieveData* operation a DM command that produces data. The *WriteDataBack* operation expects a data set and an identifier of the data container to insert the data set, e.g., a logical container name. The *IssueCommand* and *WriteDataBack* operation both deliver a message to the workflow execution engine that indicates whether the data management operation has been successfully executed or not. The *RetrieveData* operation delivers the result data produced by the input DM command in case of success. The workflow execution engine may then store this result data in the data set variable specified for the calling *RetrieveData* activity. In case of failure, the operation delivers a failure message.

Different kinds of data sources rely on different access routines and further properties for data access, e.g., different authentication mechanisms or query capabilities. Hence, the generic access operations of the SIMPL core have to be implemented for concrete data sources or sets of data sources. *Data source connectors* provide this implementation and account for the specific properties of data sources. For example, we use a data source connector for data sources that are based on JDBC access mechanisms. Another connector supports the application programming interface of a certain file system. Some data sources do not support all SIMPL core operations. For instance, sensor nets do not allow for writing data back as they are only able to deliver data. In such a case, the corresponding data source connectors do not provide these operations as well.

The SIMPL core additionally provides *data converters* that transform data from the output format of a data source connector to an XML-based format for the process context of the workflow and vice versa. For instance, a data converter transforms data between the JDBC result set format and the *XML RowSet* format of BPEL-DM. Such data converters may be used for data retrievals or for writing data back to a data source, i.e., for the *RetrieveData* and *WriteDataBack* operations and activities.

4.2 Metadata for Mappings to Heterogeneous Access Mechanisms

We enhance the resource management component of the service bus with metadata about data sources. These metadata describe the mappings between the unified interfaces of the SIMPL core and the underlying and possibly heterogeneous data source access mechanisms. Four kinds of objects may be registered in the resource management component: data sources, data containers, data source connectors, and data converters. Figure 6 shows the classification of the corresponding metadata as well as the cardinalities of associations between individual metadata classes.

A *logical source name* is unique for each data source and acts as its identifier within the SIMPL framework. It can be used as logical data source descriptor within workflows, e.g., in a data source reference variable of BPEL-DM. This constitutes an abstraction offered to the modeler since he/she does not need to deal with real interfaces or security entities. The *interface description* contains information about the interface of the data source, in particular an endpoint to access it. The *security entities*, such as usernames and passwords, enable authorized data source access. The *description of further* *functional or nun-functional properties* typically includes properties of the data source like the maximally expected response time. Such properties may refer to requirements specified in a logical data source descriptor in order to perform a late binding of data sources. The *data container objects* describe the containers that are managed by the associated data source. They have a *logical container name* assigned that acts as a container reference in workflows, e.g., in a data container reference variable of BPEL-DM. This name is mapped to the concrete *local container identifier* that uniquely identifies the container within the data source.

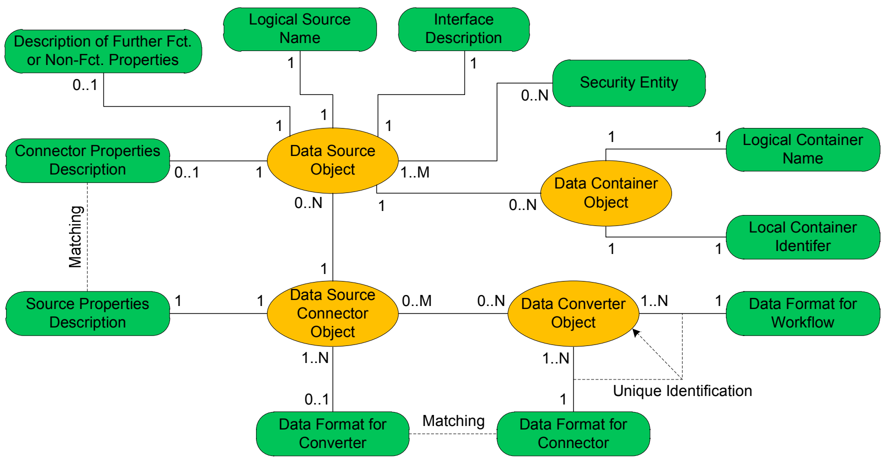


Fig. 6. Classification of metadata to unify heterogeneous data source access mechanisms

As described in Section 4.1, *data source connectors* implement the SIMPL core operations for the data sources they are associated with. Connectors may also be used for multiple data sources, e.g., one connector for all JDBC-based database systems. There might be multiple implementations for a single data source connector registration or data converter registration. In that case, one of these implementations has to be chosen during data source access via additional selection mechanisms, e.g., via the approach of [Ka07]. However, we do not further deal with this aspect for the sake of simplicity.

When a data source is registered or when its registration is updated, the user may directly associate a connector to it. If the user is not sure which connector may handle the data source, he/she may use its *connector properties description*. It describes the properties a connector must have in order to connect to the data source. A similar description, i.e., the *source properties description*, is associated with each data source connector. It describes the properties a connector expects from associated data sources. For instance, both properties descriptions name the SIMPL core operations the associated data source and data source connector support. They are matched to each other to decide on the correct connector for the data source. The same matching can be used when a connector registration is added or updated to find all data sources the connector may handle.

A data source connector is furthermore associated with a description of a *data format for a converter*. It denotes the data format in which the connector delivers output data to a requestor or expects input data from it. A data converter has a similar data format description associated. These data format descriptions are used to map connectors and converters to each other during the registration of either objects or the update of a registration. So, only those connectors and converters are associated with each other that rely on the same data format. The second data format description associated to a data converter denotes the format in which the converter expects input sent from a workflow and in which it sends its output back to the workflow, e.g., *XML RowSet* of BPEL-DM. The pair of data formats associated with a converter defines between which formats it is able to transform data. As a constraint, this format pair uniquely identifies a data converter object, i.e., there is at most one converter object for each possible pair.

We enhance the resource management component with the functionalities metadata management, metadata provisioning, and metadata integration (see Figure 3). *Metadata management* ensures a persistent and transactional storage of the metadata as well as the management of the metadata schema. The *metadata provisioning* provides metadata information to other components of the sWfMS. It offers a query interface for one or more query languages, e.g., SQL. Besides, it may also offer further repository services that go beyond simple query answering. For example, a service may execute a series of queries that each resolves a selection rule that is used for late binding of data sources. The *metadata integration* is responsible for integrating metadata from internal and external metadata sources and for dealing with according heterogeneities, in particular regarding the metadata schemas and their contents. Such metadata sources can be, e.g., users that access the resource management component via the GUI, external registries that also describe data sources, and the external data sources themselves. Each of these sources may register metadata objects with associated metadata. They may also be asked to complement the metadata after another party has registered an object. For example, a user may register a data source, which then provides all data containers it manages.

4.3 Data Source Access using the SIMPL Framework

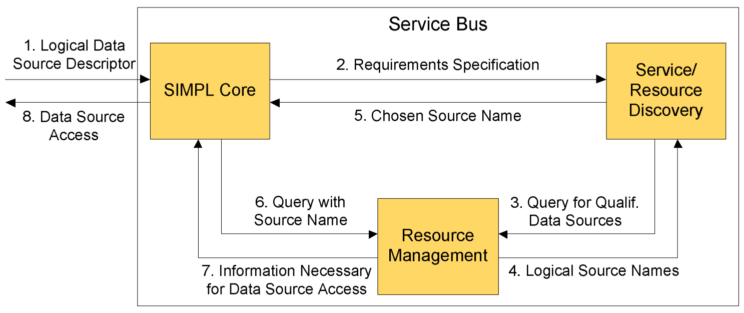

Service Bus
1. Logical Data Source Descriptor
8. Data Source Access
SIMPL Core
2. Requirements Specification
5. Chosen Source Name
Service/ Resource Discovery
6. Query with Source Name
3. Query for Qualif. Data Sources
Resource Management
7. Information Necessary for Data Source Access
4. Logical Source Names

Fig. 7. Interaction of service bus components to prepare data source access

When a workflow accesses a data source via a logical data source descriptor, the SIMPL framework needs to map this descriptor to all information that is necessary for data source access. This information consists of the interface description, a security entity, the suitable data source connector, and the suitable data converter. Furthermore, it needs to map logical names of referenced data containers to local container identifiers. In the following, we describe how the components of the service bus interact with each other to achieve this mapping. Figure 7 shows this interaction in case the logical data source descriptor sent to the SIMPL core (step 1) contains a requirements specification for a data source. In case it contains a logical name, we skip steps 2 to 5.

The SIMPL core sends the requirements specification to the discovery component (2). The latter queries the resource management component to map the requirements to a set of data source names that identify data sources meeting the claimed requirements (3 and 4). The discovery component then chooses one of these names based on selection criteria in the requirements specification and sends the chosen name back to the SIMPL core (5). The latter queries the resource management component with the source name to retrieve the above-mentioned information that is necessary for data source access (6 and 7). This information is used to access the data source and to execute the SIMPL core operation that is identified by the calling workflow activity (8), e.g., an *IssueCommand* activity calls the *IssueCommand* operation. Strictly speaking, we execute the implementation of this operation as provided by the data source connector identified before.

If the workflow performs a data retrieval or write back, we will need to identify exactly one of the converters that are associated with the identified connector. For that purpose, the workflow engine sends the data type of the BPEL variable that holds the data to be retrieved or written back to the SIMPL core. This data type determines the correct workflow-specific format of the converter. This data format and the connector-specific format assigned to the resolved connector uniquely identify the correct converter.

5 Discussion and Evaluation

In this section, we discuss the benefits and drawbacks of the SIMPL framework. In particular, our discussion covers generality issues of SIMPL, its extensibility, and optimization opportunities for data management in simulation workflows. Afterwards, we evaluate SIMPL via the example workflow for bone remodeling of Section 2.2.

5.1 Generic Data Management for Simulation Workflows

The generic access operations of the SIMPL core, the metadata to describe data sources, and the logical data source descriptors provide a uniform access to arbitrary heterogeneous data sources. This eases the integration of further data management techniques in addition to BPEL-DM. Besides that, we can port the SIMPL core and the metadata management to other sWfMS implementations, which may use different workflow engines, different workflow languages, or even different solutions for modeling data management and data provisioning. The high degree of portability of the framework is basically achieved by its architecture based on clearly separated components and plug-ins extending a sWfMS.

The DM activities of BPEL-DM offer common functionality for data access, data manipulation, data definition, and for writing data back to a data source. Furthermore, SIMPL includes a multitude of data management patterns as further abstraction support.

Together with the uniform data source access and the portability provided by SIMPL, these data management and data access patterns constitute a generic data management solution for simulation workflows. This generality enables SIMPL to be used in multiple domains of simulations or other scientific applications, such as biology and astronomy, and even in the business domain, e.g., for business ETL workflows.

In contrast to our approach, one could provide data services to accomplish the data management for simulation workflows. These services usually offer efficient means for data management functionality specific to a small set of domains or problems. Hence, they do not provide a consolidated, generic, and flexible way to define data management for multi-domain simulation workflows. Nevertheless, since we use BPEL as workflow language, we allow services to be the implementation of data management tasks as well. Furthermore, BPEL processes also offer their functions via service interfaces. So, our approach may even be used to define data services that support special needs of certain domains or problems.

To the best of our knowledge, no other approach includes an abstraction support to define data management operations that is based on generic data management patterns. Typically, workflow modelers have to define low-level details of data management, such as concrete DM commands. By distinguishing the data management operations that are necessary for simulations between different abstract patterns, we can reduce the degrees of freedom in defining the respective operations. This eases the definition and implementation of abstraction mechanisms for individual patterns. Furthermore, the patterns can be seen as building blocks for composing data provisioning workflows, e.g., ETL workflows, via process fragment technology [EUL09]. This increases flexibility at runtime and reduces modeling costs at build time.

5.2 Extensibility of SIMPL

The specifications of the SIMPL core operations and of the DM activities of BPEL-DM are independent of the underlying data sources. The same holds for the logical data source descriptors and the logical data container names. Hence, they do not need to be extended or adapted when SIMPL should support additional kinds of data sources. We only need to add according data sources connectors as well as data converters and XML schema definitions for data set variables. Furthermore, the implementations of connectors and converters as well as the XML schema definitions can typically be derived from already existing implementations or definitions.

In the same sense, we may extend or customize BPEL-DM by additional activities. Like data services, these activities could account for specific needs of a certain scientific domain or problem. To do that, we need to add new SIMPL core operations and their implementations by data source connectors, but only if the already existing operations are not suitable. In order to add a new data management pattern to the DM pattern plug-in of the function catalog, suitable rewrite rules have to be defined. These rules describe how the pattern is to be converted into executable workflow parts. Altogether and in contrast to previous approaches, e.g., see the approaches compared in [Vr08], we can typically reuse much of the already existing code for extending SIMPL.

5.3 Optimization of Data Management for Simulation Workflows

Our BPEL-DM approach combines the definition of activities for data management and simulation at the same level of abstraction. This offers a huge potential for a consolidated optimization at both the workflow and the data processing level. In [Vr07], the authors present a flexible approach to optimize workflows with embedded data management operations, in particular SQL statements. Independent of the underlying data sources, this approach shows a huge optimization potential that induces significant performance improvements for workflows. Furthermore, it can be easily applied to other approaches for embedded data management operations, e.g., to our BPEL-DM approach. Due to these optimization options, the SIMPL framework is well suited for a data management and data provisioning abstraction efficiently dealing with huge, heterogeneous, and distributed data objects.

5.4 The Bone Remodeling Workflow in the SIMPL Framework

As a proof of concept, we developed a prototype that implements SIMPL and relevant parts of the associated sWfMS architecture. This prototype uses the Eclipse BPEL Designer[1] as scientific workflow modeler and the Apache Orchestration Director Engine[2] (ODE) as workflow execution engine and deployment component. Based on the prototype, we implemented the workflow for a bone remodeling simulation (BRS) presented in Section 2.2. Its activities involve several heterogeneous data sources, e.g., databases, CSV files, unstructured documents, or image files. The SIMPL core and the metadata of the resource management provide a uniform access to these data sources.

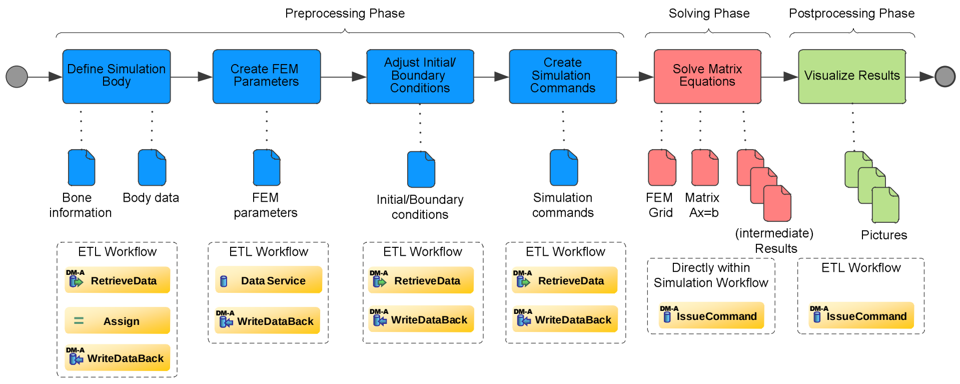


Fig. 8. Workflow for bone remodeling simulation enhanced with SIMPL

Figure 8 shows the BRS workflow using the SIMPL framework. In particular, we show the BPEL-DM activities implementing the main steps of the workflow and its data provisioning at the bottom of the figure. In the preprocessing phase, the workflow creates more than ten input data files each with a size up to one gigabyte. Without the framework, scientists have to select all needed input data, transform the data, and store

[1] http://www.eclipse.org/bpel/
[2] http://ode.apache.org/

results into files in the PANDAS environment. SIMPL helps to automate these tasks, thereby reducing the error rate. Input data for the BRS activity *Define Simulation Body*, e.g., bone information or material parameters, are typically stored in public databases or in private file systems. The simulation workflow invokes a separate ETL workflow that converts the data and transfers it into the PANDAS environment. It consists of *RetrieveData* activities that load the input data into BPEL variables. Afterwards, the workflow transforms the data via an assign activity, and *WriteDataBac*k activities write the results into the target files. The activities *Adjust Initial/Boundary Conditions* and *Create Simulation Commands* operate according to the same procedure except that they read their input data from structured CSV files. Hence, they also use *RetreiveData* and *WriteDataBack* activities for data selection and transmission.

To perform the FEM, the BRS activity *Create FEM Parameters* has to select certain interpolation functions from an unstructured text document that summarizes all available functions. SIMPL is based on forwarding DM commands to data sources, but the underlying file system does not support executing extractions of unstructured data via DM commands. Hence, SIMPL is not able to directly select these functions. We need data services that select all necessary information and store it into workflow variables. A *WriteDataBack* activity subsequently copies this data into the PANDAS environment.

During the solving phase, the activity *Solve Matrix Equations* calculates matrix equations for several time steps. For each step, it stores all relevant data, i.e., the intermediate results, the FEM grid, and the matrix and vectors into a database inside the PANDAS environment. For selected time steps, a repeatedly executed *IssueCommand* activity in the simulation workflow persists snapshots of these data for further processing, e.g., for analyzing or for recovery purposes. A typical BRS produces 100 or even more of such data snapshots each with a size of about two megabytes. After the last time step, the *IssueCommand* activity stores the final result.

The last BRS activity *Visualize Results* transforms these results and other data, such as the FEM grid, into a format suitable for visualization tools. For example, it joins the FEM grid data and the simulation results for selected time steps of the solving phase in order to create images that combine this information. To automate this, *IssueCommand* activities of an ETL workflow select the necessary data and transform and match it.

The BRS workflow benefits from SIMPL in various ways. SIMPL provides a uniform access to all involved heterogeneous data sources, i.e., databases, CSV files, unstructured text documents, and image files. Furthermore, it allows to automate the data management activities that have previously been performed manually. This reduces the error rate and the time the scientists have to spend for these activities. We chose ETL workflows for the preprocessing and the post-processing phase since they may transfer data directly between the involved data sources. So, we can reduce costs for transmitting high amounts of data and decrease workload for data processing within the simulation workflow. During the solving phase, an *IssueCommand* activity could even transform all FEM data into formats suitable for different solvers, e.g., parallel solvers, i.e., SIMPL helps to switch between these different solvers. Altogether, SIMPL offers a data management abstraction that is well suited for simulation workflows such as the BRS.

6 Related Work

Federated information systems integrate different kinds of data sources and provide a homogeneous schema for heterogeneous source systems [Bu99]. However, they typically involve multiple and sophisticated integration processes that have to be executed for each data source access. In simulation applications the sources are highly heterogeneous and we need to cope with huge amounts of data. Thus, complex integration processes may show poor performance. In that case, a peer-to-peer-based approach seems more suitable as it employs less complex integration processes between pairs of data sources. The generality of our approach recommends it to be used for both a federated and a peer-to-peer-based solution. This also holds for conventional ETL tools. But in contrast to our approach, they rather work with various access mechanisms and data management operators that are specific for a certain kind of data source.

Scientific applications recently adopted the facilities of grid infrastructures as well as the Service Oriented Architecture (SOA) [TDG07]. The most prominent solution for grid- and service-based data management is the Open Grid Services Architecture – Data Access and Integration[3] framework (OGSA-DAI). It encapsulates heterogeneous and distributed data sources via services that provide access abstractions for the data sources. A user may define data integration workflows that orchestrate interactions with these services. However, the workflows and workflow tasks of OGSA-DAI are implemented directly in programming languages. If simulation workflows that rely on conventional workflow technology use OGSA-DAI as data management solution, they will not exploit the optimization potential of a consolidated definition of the processing activities for data management and simulation [Vr07]. Furthermore, the abstraction support offered by OGSA-DAI only relies on the customized abstractions offered by the individual services, while we provide a generic and unified abstraction mechanism.

The Scientific Data Management Center (SDM Center) offers an end-to-end data management approach that mainly deals with efficiently analyzing data produced by scientific simulations or experiments [Sh07]. It offers efficient and parallel access routines to storage systems and technologies to support the better understanding of data. The latter comprises, for example, routines for specialized feature discovery, algorithms for parallel statistical data analysis, and efficient indexes over large and distributed data sets. On top of this, the Kepler sWfMS provides the robust automation of processes for generating, collecting, and storing the results of simulations or experiments as well as for data post-processing and analyzing the results [Lu06]. In contrast to the SDM Center, our approach does not deal with data analysis, but with data provisioning for simulation workflows and an appropriate abstraction support. Kepler also offers workflow activities to seamlessly access data sources, in particular for local file systems, relational database systems, and data streams coming from sensor networks. However, each of these activities directly deals with the heterogeneities regarding access mechanisms of the considered data sources instead of using generic and unified interfaces.

[3] http://www.ogsadai.org.uk/index.php

The scientific workflow management system VisTrails, focuses on the exploration and visualization of results of simulations or experiments as well as on modeling, executing, and optimizing visualization workflows [Fr06]. It supports tracking revisions of workflows, i.e., scientists or engineers may interactively adjust their workflows. In order to maintain the history of workflow execution, data processing, and workflow revisions, VisTrails captures data and workflow provenance and links them to each other and to the produced data [Ko10]. This enables reproducibility of processes and simplifies the exploration of different versions of a workflow as well as its results. In contrast to the framework presented in this paper, VisTrails does not focus on data provisioning aspects and abstractions that are necessary for executing all phases of simulations.

Microsoft Trident is a general-purpose scientific workflow workbench [Ba08]. It is built on top of the Microsoft Windows Workflow Foundation[4] (Windows WF), a workflow environment based on the control-flow oriented Extensible Orchestration Markup Language (XOML). Trident enhances Windows WF with functionality needed for scientific workflow management, e.g., automatic provenance capture and the possibility to model data dependencies between workflow tasks. The activity library of Windows WF enables customized activity types that could provide a seamless access to data sources or further abstractions for defining data management operations. However, they have to be implemented by the modeler himself or shared between several activity developers. SIMPL offers abstractions via data management patterns that are automatically converted into executable workflow parts. As an alternative to such custom activity types, Trident uses services for data access. Similar to OGSA-DAI, this complicates optimizations over the whole spectrum from the workflow to the data processing level. Furthermore, Trident workflows may use Dryad for data provisioning [Is07]. Following the approach of MapReduce [DG04], Dryad supports programmers in efficiently using multiple resources for executing data-intensive and data-parallel applications without knowing anything about concurrent programming. However, Dryad does not deal with data management abstractions in our sense of a generic solution.

Besides the activity library of Windows WF, IBM and Oracle also provide workflow activities that directly embed data management operations as part of their workflow products [Vr08]. In contrast to SIMPL, these products do not offer abstractions via data management patterns and are restricted to SQL statements, while we support any kind of data source. The external variables of Apache ODE are another approach to seamlessly access data sources from within workflows. These variables can be mapped to one row of a table in a database that offers an interface following Java Database Connectivity[5] (JDBC). This way, workflows may perform tuple-oriented retrievals and manipulations on the mapped row. However, set-oriented operations have to be defined via additional workflow constructs, e.g., loop activities. In [Vr07], the authors proof that such a loop-based execution of several tuple-oriented operations shows weak performance related to a set-oriented SQL statement that is wholly executed by the database system. Our approach supports set-oriented operations by directly integrating SQL statements into the workflow definition.

[4] http://www.windowsworkflowfoundation.eu/
[5] http://java.sun.com/products/jdbc/overview.html

7 Conclusion and Future Work

In this paper, we introduced SIMPL – an extensible framework that provides a generic and consolidated abstraction for data management and data provisioning in simulation workflows. It unifies heterogeneous interfaces to different data sources via logical data source descriptors, generic access operations, and metadata for mappings to concrete data source access mechanisms. We demonstrated that this provides the core functionality to uniformly access arbitrary data sources and enables an easy development and integration of concrete data management techniques. Based on this, the BPEL-DM activities allow for the definition and execution of common data management and data provisioning tasks for simulation workflows. Further abstraction support is provided by means of generic data management patterns, e.g., patterns for ETL operations. In addition to a data source access via services, BPEL-DM offers the combined definition of the processing activities for data management and simulation at the same level of abstraction. This enables optimizations over the whole spectrum from the workflow level to the data level, inducing significant performance improvements of workflows. Altogether, the SIMPL framework removes the burden from engineers and scientists to specify low-level details of data management for their simulations. It helps them to cope with the information explosion intrinsically associated with simulation applications and boosts their productivity.

In future, we will extend the optimization approach for workflows with embedded data management operations of [Vr07] to be applicable to the data management in simulation workflows. For that purpose, we will work on a set of optimization rules that are suitable for simulation workflows and for DM activities of BPEL-DM. Scientists may use several parameterized data management patterns within their workflows. Our approach converts each pattern into an executable workflow part in isolation from all other patterns. This may result in a variety of process fragments for data management and data provisioning that show further optimization potential when considered together. To exploit this optimization potential, we will combine the conversion of data management patterns with the optimization approach for data management.

Acknowledgement: The authors would like to thank the German Research Foundation (DFG) for financial support of the projects within the Cluster of Excellence in Simulation Technology at the University of Stuttgart.

Bibliography

[AMA06] Akram, A.; Meredith, D.; Allan, R.: Evaluation of BPEL to Scientific Workflows. In: Proc. of the 6th International Symposium on Cluster Computing and the Grid (CCGRID '06), Singapore, Malaysia, 2006.

[Ba08] Barga, R. et. al.: The Trident Scientific Workflow Workbench. In: Proc. of the 4th International Conference on e-Science, Indianapolis, Indiana, 2008.

[Bu99] Busse, S. et. al.: Federated Information Systems: Concepts, Terminology and Architectures. Research Report of the Faculty of Computer Science at the Technische Universität Berlin, 1999.

[DC08] Deelman, E.; Chervenak, C.: Data Management Challenges of Data-Intensive Scientific Workflows. In: Proc. of the 8th International Symposium on Cluster Computing and the Grid (CCGRID '08), Washington, DC, 2008.

[DG04] Dean, J.; Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Proc. of the 6th Symposium on Operating Systems Design and Implementation, San Francisco, California, 2004.

[EUL09] Eberle, H.; Unger, T.; Leymann, F.: Process Fragments. In: On the Move to Meaningful Internet Systems: OTM 2009, Part I. Springer, 2009.

[Fr06] Freire, J. et. al.: Managing Rapidly-Evolving Scientific Workflows. In: Proc. of the 1st International Provenance and Annotation Workshop (IPAW), Chicago, Illinois, 2006.

[Gi07] Gil, Y. et. al.: Examining the Challenges of Scientific Workflows. In: IEEE Computer, vol. 40, no. 12, December 2007.

[Gö11] Görlach, K. et. al.: Conventional Workflow Technology for Scientific Simulation. To appear in: Yang, Y. (ed.); Wang, L. (ed.); Jie, W. (ed.): Guide to e-Science. Springer, 2011.

[Is07] Isard, M. et. al.: Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In: Proc. of the European Conference on Computer Systems (EuroSys), Lisbon, Portugal, 2007.

[Ka07] Karastoyanova, D. et. al.: Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. In: Proc. of the 2nd International ICDE Workshop on Service Engineering (SEIW 2007), Istanbul, Turkey, 2007.

[KME10] Krause, R.; Markert, B.; Ehlers, W.: A Porous Media Model for the Description of Adaptive Bone Remodelling. In: Proc. of the 81st Annual Meeting of the International Association of Applied Mathematics and Mechanics, Karlsruhe, Germany, 2010.

[Ko10] Koop, D. et. al.: Bridging Workflow and Data Provenance using Strong Links. In: Proc. of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Heidelberg, Germany, 2010.

[LR99] Leymann, F.; Roller, D.: Production Workflow: Concepts and Techniques. Prentice Hall, Englewood Cliffs, NJ, 1999.

[Lu06] Ludäscher, B. et. al.: Scientific Workflow Management and the Kepler System. In: Concurrency and Computation: Practice and Experience, vol. 18, issue 10, 2006.

[Ma05] Maier, A. et. al.: On Combining Business Process Integration and ETL Technologies. In: Gesellschaft für Informatik (ed.): Datenbanksysteme in Business, Technologie und Web, Karlsruhe, Germany, 2005.

[Mü10] Müller, C.: Development of an Integrated Database Architecture for a Runtime Environment for Simulation Workflows. Diploma Thesis, University of Stuttgart, 2010.

[Oa07] OASIS: Web Services Business Process Execution Language Version 2.0, 2007.

[Sh07] Shoshani, A. et. al.: SDM Center Technologies for Accelerating Scientific Discoveries. In: Proc. of the Scientific Discovery through Advanced Computing Conference (SciDAC 2007), Boston, Massachusetts, 2007.

[SK10] Sonntag, M.; Karastoyanova, D.: Next Generation Interactive Scientific Experimenting Based on the Workflow Technology. In: Proc. of the 21st IASTED International Conference on Modelling and Simulation (MS 2010), Prague, Czech Republic, 2010.

[TDG07] Taylor, I.; Deelman, E.; Gannon, D.: Workflows for e-Science - Scientific Workflows for Grids. Springer, London, UK, 2007.

[Vr07] Vrhovnik, M. et. al.: An Approach to Optimize Data Processing in Business Processes. In: Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria, 2007.

[Vr08] Vrhovnik, M. et. al.: An Overview of SQL Support in Workflow Products. In: Proc. of the 24th International Conference on Data Engineering (ICDE 2008), Cancún, México, 2008.