A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design

Peter Reimann, Holger Schwarz, and Bernhard Mitschang

Institute of Parallel and Distributed Systems, University of Stuttgart, Germany

Abstract. Scientific workflows may be used to enable the collaborative implementation of scientific applications across various domains. Since each domain has its own requirements and solutions for data handling, such workflows often have to deal with a highly heterogeneous data environment. This results in an increased complexity of workflow design. As scientists typically design their scientific workflows on their own, this complexity hinders them to concentrate on their core issue, namely the experiments, analyses, or simulations they conduct. In this paper, we present a novel approach to a pattern-based abstraction support for the complex data management in simulation workflows that goes beyond related work in similar research areas. A pattern hierarchy with different abstraction levels enables a separation of concerns according to the skills of different persons involved in workflow design. The goal is that scientists are no longer obliged to specify low-level details of data management in their workflows. We discuss the advantages of this approach and show to what extent it reduces the complexity of simulation workflow design. Furthermore, we illustrate how to map patterns onto executable workflows. Based on a prototypical implementation of three real-world simulations, we evaluate our approach according to relevant requirements.

1 Introduction

Nowadays, *scientific workflows* are increasingly adopted to enable the collaborative implementation of scientific applications across various domains [24]. This includes applications such as experiments, data analyses, or computer-based simulations. *Simulation workflows*, as a sub-area of scientific workflows, are typically compositions of long-running numeric calculations [5]. These calculations realize mathematical simulation models, e. g., based on partial differential equations. To make simulations more realistic and their outcome more precise, scientists couple simulation models from several scientific domains in simulation workflows [8]. This allows them to cover various levels of granularity in simulation calculations. The calculations for each of the simulation models have to process huge data sets from diverse data sources and in multiple proprietary data formats [15]. Furthermore, the results from one simulation model are often used as input for other simulation models. As different simulation models and underlying simulation tools typically rely on different solutions for data handling, this even multiplies the complexity of data management in simulation workflows. So, these workflows have to embed many sophisticated data provisioning and data integration tasks.

R. Meersman et al. (Eds.): OTM 2014, LNCS 8841, pp. 21–38, 2014.

As an example, consider a simulation of structure changes in human bones, which combines a bio-mechanical simulation on the macro scale with a systems-biology simulation on the micro scale [8]. The workflow realizing this coupled simulation consists of more than 20 activities [15]. Two thirds of these activities deal with service calls and the complex data management necessary to provide data for these services. Today, simulation workflow design is typically carried out by the scientists themselves. Hence, they have to specify many low-level details of data provisioning, but they usually do not have the necessary skills in workflow design or data management. This hinders scientists to concentrate on their core issues, namely the development of mathematical simulation models, the execution of simulation calculations, and the interpretation of their results.

In a previous publication, we have presented the vision of a pattern-based approach to make simulation workflow design tailor-made for scientists [15]. This approach goes beyond related work for artifact-centric business process modeling [6,9], for workflow patterns [17], in the scientific workflow domain [11,13,26], and in the area of data integration [14,19]. Scientists using our approach select a few abstract patterns and combine them in their simulation workflows to describe only the main steps of these workflows. The adaptation to a concrete simulation scenario is achieved by a small set of pattern parameters that mainly correspond to terms or concepts scientists already know from their simulation models. Finally, rewrite rules transform such parameterized patterns into executable workflows. In this paper, we extend this vision of a pattern-based workflow design to a full-fledged and principled abstraction support for the complex data provisioning in simulation workflows. The main contributions are:

- We identify major requirements for the data provisioning in simulations.
- We derive a comprehensive set of patterns that may be used to alleviate the design of the data provisioning in several kinds of simulation workflows.
- We show how to organize patterns in a pattern hierarchy with different abstraction levels. As a major contribution, this pattern hierarchy facilitates a separation of concerns between different persons that are involved in workflow design, e. g., scientists, data management experts, or workflow engineers. According to his or her own skills, each person may choose the abstraction level s/he is familiar with and may provide other workflow developers with parameterized patterns and workflow fragments at this abstraction level.
- We discuss design considerations for a rule-based transformation of patterns into executable workflows and show its application to a concrete simulation.
- We evaluate the presented approach based on a prototypical implementation and based on its application to three real-world simulations.

In the remainder of this paper, we firstly detail our running example and exemplify major requirements for the data provisioning in simulations. In Section 3, we introduce the approach of pattern-based simulation workflow design and detail our set of patterns as well as the pattern hierarchy. The rule-based transformation of patterns into executable workflows is explained in Section 4. Section 5 covers information about our prototype and an evaluation of our approach. Finally, we discuss related work in Section 6 and conclude in Section 7.

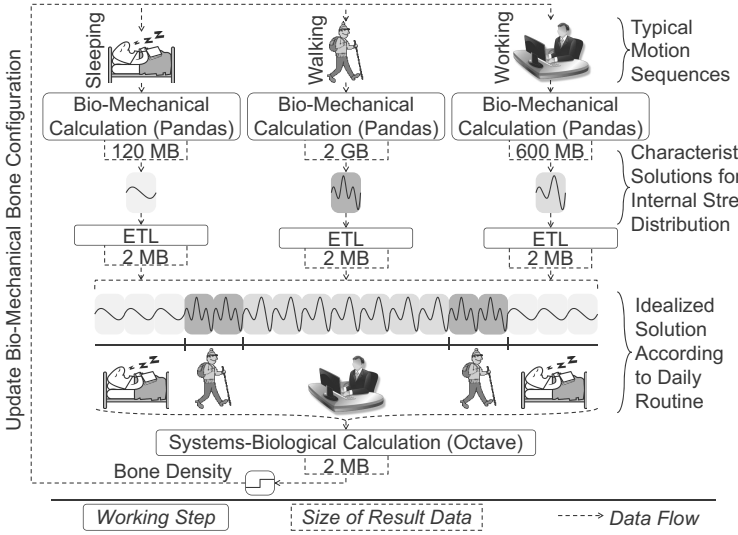


Fig. 1. Coupled simulation of structure changes in bones, cf. [8]

2 Data Complexity in Simulations

A solution to conquer the complexity of data provisioning in simulation workflow design has to consider a set of specific requirements. We introduce the discussion of these requirements by means of an example simulation.

2.1 Simulation of Structure Changes in Bones

As an example, we consider a simulation of time-dependent structure changes in bones, e. g., to support healing processes after bone fractures [8]. Fig. 1 shows how this simulation couples simulation models of the domains bio-mechanics and systems-biology, which involves lots of cooperative work across these domains. The bio-mechanical simulation model describes the mechanical behavior of a bone on a macroscopic tissue scale. However, it does not consider any cellular reactions within the bone tissue. This is where the systems-biological model comes into play, which determines the microscopic formation or resorption of the bone tissue as result of the stress-regulated interaction between cells.

The complexity of coupling simulations and providing the appropriate data for each of the simulation models is increased by the fact that typically separate simulation tools are employed for each of the models [8]. In our example, the Pandas framework[1], which is based on the Finite Element Method (FEM), offers a numeric implementation of the bio-mechanical simulation. The boundary conditions of this simulation correspond to the external load on the bone, e. g., caused by muscle forces. This external load depends on the motion sequence of

[1] `http://www.mechbau.uni-stuttgart.de/pandas/index.php`

the relevant person. Pandas converts the external load of each relevant motion sequence into a characteristic solution of the internal stress distribution within the bone tissue. For each numeric time step and for each nodal point of the FEM grid, it thereby stores up to 20 mathematical variables in a SQL database. This results in a data volume between 100 MB and several GBs per motion sequence.

The systems-biological calculation may be implemented via GNU Octave[2]. It gets an idealized solution as input that is composed of the characteristic solutions of Pandas according to the approximated daily routine of the person. Beforehand, ETL processes filter appropriate data from the database of Pandas and aggregate these data among all numeric time steps, e. g., by calculating average values. Furthermore, they define the partitioning of the data among multiple instances of Octave to enable parallelism, and they store the results in CSV-based files (comma-separated values). Octave uses these files as input, calculates the precise bone density until the end of one daily routine, and stores it in another CSV-based file. This file is then imported into the database of Pandas to update the bone configuration of the bio-mechanical simulation model with the bone density. The whole process is repeated for the next daily routine.

2.2 Requirements for Data Provisioning

Such a complex data environment is common for simulations that are coupled among different scientific domains, since each domain has its own requirements and solutions for data handling. Due to the heterogeneity and high amount of involved data, scientists have to implement a multiplicity of complex data provisioning and data integration tasks. Workflows may help to structure these tasks [5], but they do not remove the burden from scientists to specify many low-level details of data management. To design an executable workflow realizing the simulation depicted in Fig. 1, scientists need to specify at least 22 complex workflow tasks [15]. This includes 9 low-level tasks that filter, aggregate, sort, split, and transform data. Here, scientists need to define sophisticated SQL, XPath, or XQuery statements. Scientists have much knowledge in their simulation domain, but rather limited skills regarding workflow or data management, and especially regarding SQL, XPath, or XQuery. So, they often waste time for workflow design they actually want to spend on their core issue, namely the simulation.

These issues imply a set of essential requirements that have to be met to ensure a wide adoption of simulation workflow technology. To identify these requirements, we have analyzed several academic use cases described in literature (see [20,24]), different workflow systems, and a set of real-world simulation processes. Beyond the simulation illustrated in Section 2.1, this set of real-word processes covers the examples described by Fehr et al. [3] and Rommel et al. [16]. In the following, we discuss the most relevant requirements:

- Requirements relevant for workflow design:
  - To conquer the complexity inherently associated with simulation workflows, we mainly require an *abstraction support for the definition of data*

[2] `http://www.gnu.org/software/octave/`

provisioning and data integration tasks that is suitable for scientists. On the one hand, this abstraction support should release scientists from defining complex implementation details or a high number of workflow tasks. On the other hand, it should enable them to work with terms or concepts they already know from their simulation models, instead of using workflow or data modeling languages they are not familiar with [11].
  - The second requirement arises directly from the desire for coupling simulations and heterogeneous data environments of different scientific domains. To enable a seamless simulation coupling across arbitrary domains, an abstraction support has to be sufficiently *generic*, i. e., it has to consider all individual requirements of the respective domains [8,15].
- Requirements relevant for workflow execution:
  - The total size of the data involved in particular simulation runs ranges from a few MBs to several Terabytes. Furthermore, this may include a dynamically changing data volume, in particular for coupled simulations. Such a high and varying amount of data emphasizes the need for an *efficient data processing* and for *optimization opportunities* [2,12,21,25].
  - Scientists often make ad-hoc changes to workflows at runtime [22]. This requires an appropriate *monitoring* of workflow execution. Further relevant aspects are the reproducibility of a simulation and the traceability of its outcome, which has led to many *provenance* technologies [4].

In this paper, we propose a novel approach to solve the requirements regarding workflow design, which have largely been neglected in previous work. In the evaluation in Section 5, we nevertheless consider all four requirements.

3 Pattern-Based Simulation Workflow Design

In this section, we describe our approach that uses patterns to alleviate the design of simulation workflows. The core idea is that scientists select and combine only a few number of simulation-specific process patterns that represent high-level building blocks of simulation processes. These simulation-specific patterns combine several fine-grained workflow tasks, which reduces the number of tasks that are visible to scientists. Furthermore, they allow for a domain-specific and thus easy parameterization. After scientists have parameterized chosen patterns, these patterns need to be transformed into executable workflows. This is achieved by rewrite rules that recursively transform simulation-specific process patterns into more concrete workflow patterns, templates of workflow fragments, or services. Depending on their degree of implementation details, these workflow patterns, workflow fragments, and services may be provided by persons with different skills, e. g., by data management experts or workflow engineers. In the following subsections, we firstly discuss how to facilitate such a separation of concerns between different persons by organizing patterns in a pattern hierarchy with several abstraction levels. Afterwards, we illustrate the set of patterns and how they may be used for workflow design at each abstraction level. Information about the rule-based approach to pattern transformation are given in Section 4.

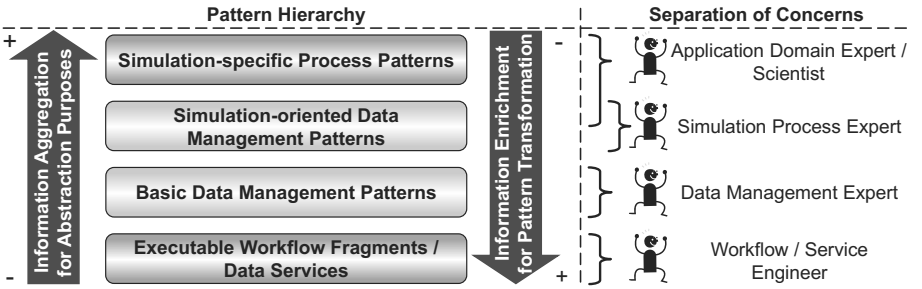


Fig. 2. Hierarchy of data management patterns with different abstraction levels

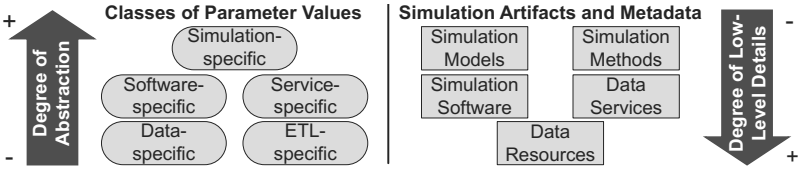


Fig. 3. Main classes and degree of abstraction of parameter values for patterns

3.1 Pattern Hierarchy and Separation of Concerns

Fig. 2 shows the afore-mentioned pattern hierarchy ranging from simulation-specific process patterns to executable workflow fragments and data services. To identify individual patterns and to properly arrange them in the hierarchy, we have analyzed the same uses cases as for the identification of requirements discussed in Section 2.2. In the following, we mainly focus on data management patterns that are relevant for data provisioning in simulation workflows.

As a major contribution, the clear distinction between individual abstraction levels in the pattern hierarchy enables a separation of concerns according to the skills of different persons involved in workflow design. Fig. 2 illustrates a possible separation of concerns between application domain experts (in our case scientists), simulation process experts, data management experts, and workflow or service engineers. According to his or her skills, each person may choose the abstraction levels s/he is familiar with. The person then may provide other persons with templates of parameterized patterns or workflow fragments at chosen levels. Thereby, the individual patterns serve as medium to communicate the requirements between different abstraction levels. For example, workflow or service engineers may offer executable workflow fragments or services at the lowest level of the hierarchy. In doing so, they only need to know the basic data management patterns provided by data management experts one level above, but they do not need to be aware of simulation-specific patterns at the two top levels.

With an ascending level of the pattern hierarchy, more information about data management operations and data management technology is aggregated. Hence, workflow developers need to know less about such operations and technology

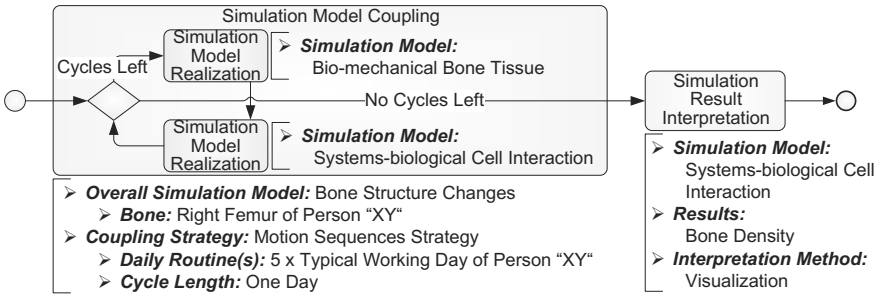


Fig. 4. Process model based on simulation-specific process patterns and their parameters for the simulation of structure changes in bones illustrated in Section 2.1

and may specify more abstract parameter values of patterns. Fig. 3 classifies parameter values and relates the resulting main classes according to their degree of abstraction, ranging from simulation-specific to data- or ETL-specific values. In line with current research efforts, we transfer the core idea of artifact-centric business process modeling [6,9] to simulation workflows and to corresponding simulation artifacts, such as mathematical simulation models or simulation software. Furthermore, we augment the artifact-centric idea by our pattern-based approach to workflow design. So, workflow developers may use simulation artifacts and their properties at different abstraction levels to specify parameter values of patterns. Additional metadata explicitly describe these simulation artifacts. This assists workflow developers in that they may choose values of some of the pattern parameters from the metadata. In the following, we detail patterns and their parameters at the individual levels of our pattern hierarchy.

3.2 Simulation-Specific Process Patterns

The top level of the pattern hierarchy shown in Fig. 2 comprises *simulation-specific process patterns* that focus on use cases scientists are interested in. Fig. 4 illustrates how such patterns may be used to describe the process of the bone simulation introduced in Section 2.1. The whole process consists of only four patterns that may be completely parameterized by *simulation-specific* values. These values correspond to artifacts scientists know from their domain-specific simulation methodology and are thus familiar with. Major examples of such artifacts are *simulation models* or mathematical variables of these models, as well as *simulation methods* such as the FEM. As such artifacts are often domain-specific, ontologies may be a good basis for metadata describing these artifacts [11,26]. Altogether, the strong relation to simulation-specific artifacts and to the use cases scientists are interested in makes simulation-specific process patterns good candidates to be selected and parameterized by scientists (see Fig. 2).

The main part of the process shown in Fig. 4 is a *Simulation Model Coupling Pattern* having the coupled simulation models for *bone structure changes* described in Section 2.1 as first parameter assigned. In addition, it defines the concrete *bone* to be simulated. The coupling pattern embeds one *Simulation Model*

Table 1. Simulation-oriented Data Management Patterns and their parameters

Pattern	*Parameters (<n..m> indicates cardinality)*
Simulation-oriented Data Provisioning	– *Simulation model* <1> – *Mathematical variables* <1..n> – *Target* <1>: reference to software instance or service
Simulation-oriented Data Interoperability	– *Simulation models* <2> – *Relationship between mathematical variables:* • *From first to second model* <1..n> • *From second to first model* <0..n> – *Partitioning mode* <0..1>
Parameter Sweep	– *Parameter List* <1> – *Operation* <1>: workflow fragment, pattern, or service to be executed for each parameter in the list – *Completion Condition* <0..1> – *Parallel* <0..1>: "yes" / "no", default is "no"

Realization Pattern for each of the two coupled simulation models, as well as the order in which they are executed. Such a pattern represents the complete realization of a simulation model, from initial data provisioning over calculations to result de-provisioning. The whole process executes the bio-mechanical model before the systems-biological one and repeats these executions for a certain number of coupling cycles. A further parameter of the overall coupling pattern completes this control flow definition by assigning the *coupling strategy* to be applied, e. g., the strategy illustrated in Fig. 1. The next parameter defines the *daily routines* and its composition of individual motion sequences. In addition, the *cycle length* determines the frequency in which the whole process re-iterates among both coupled simulation models. In our example, we consider a cycle length of one day and a total duration of five days, each corresponding to the typical working day of the relevant person. When all cycles, i. e., all days have been simulated, the process starts the result interpretation by means of a corresponding pattern. The pattern parameters specify which *results* of which *simulation model* shall be interpreted via which *interpretation method*, e. g., a visualization of the time-dependent bone density calculated by the systems-biological simulation.

3.3 Simulation-Oriented Data Management Patterns

Table 1 shows examples of *simulation-oriented data management patterns* that retain the abstraction level of simulation-specific values for most of their parameters. So, they may still be parameterized by scientists. Nevertheless, these patterns focus on use cases related to data management. As depicted in Fig. 2, this is where simulation process experts come into play who may assist scientists with orchestrating these patterns in their workflows. The *Simulation-oriented Data Provisioning Pattern* abstracts from data provisioning processes for simulation calculations or output visualizations, e. g., as part of the process patterns Simulation Model Realization or Simulation Result Interpretation shown in Fig. 4.

Table 2. Basic Data Management Patterns and their parameters

Pattern	*Parameters (<n..m> indicates cardinality)*
Data Transfer and Transformation	– *Sources* <1..n>: references to data containers – *Targets* <1..n>: references to data containers – *ETL processes for sources and targets* <0..n>
Data Iteration	– *Data set* <1>: one or a set of data containers – *Operation* <1>: workflow fragment, pattern, or service to be executed for each relevant subset of the data set – *Resources* <0..n>: e. g., references to data resources – *Completion Condition* <0..1> – *Parallel* <0..1>: "yes" / "no", default is "no" – *Data split* <1>: e. g., data-specific partitioning mode or parameters of Data Transfer and Transformation Pattern – *Data merge* <0..1>: similar to data split

The data to be provisioned is represented by a *simulation model* and by a set of its *mathematical variables*. In the example depicted in Fig. 1, the mathematical variables serving as input for the bio-mechanical simulation include its boundary conditions, e. g., the muscle forces. Only the *target* of the data provisioning needs to be specified via less abstract software- or service-specific parameter values, i. e., via a reference to a software instance or to a service that needs the data as input. Metadata describing software and services (and also data resources) as simulation artifacts may be based on common repository solutions.

The *Simulation-oriented Data Interoperability Pattern* defines data dependencies between *two simulation models* via simulation-specific parameter values [15]. This includes the *relationship between their mathematical variables*. Furthermore, a *partitioning mode* may define whether and how underlying data should be partitioned, e. g., to enable parallel executions of subsequent calculations. The *Parameter Sweep Pattern* supports processes that iterate over a *list of* simulation-specific *parameters* and execute an *operation* for each parameter in this list [12]. An optional *completion condition* defines whether and when the iteration should be finished before the whole parameter list is processed [7]. Another pattern parameter indicates whether the operation shall be executed in *parallel* or not.

3.4 Basic Data Management Patterns

On the way to executable workflows, these simulation-oriented patterns are intermediately mapped onto *basic data management patterns* whose parameters only may have service-, data-, or ETL-specific values. According to the separation of concerns depicted in Fig. 2, data management experts may use their knowledge to provide such less abstract parameter values or templates of them. Examples of basic data management patterns are listed in Table 2. The *Data Transfer and Transformation Pattern* may implement the Simulation-oriented Data Provisioning Pattern. The *sources* and *targets* of the corresponding data

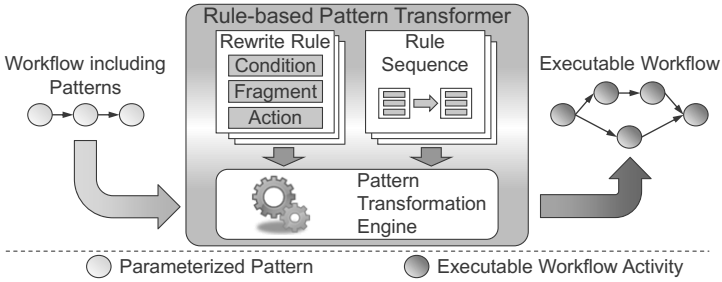


Fig. 5. Processing model of the rule-based transformation of patterns

provisioning process are specified as data-specific references to data containers, i. e., to identifiable collections of data, such as database tables or files. Furthermore, the pattern typically covers *ETL processes* including ETL operations to extract data sets from the source data containers, to transform these extracted data sets, and to load the transformation results into the target data containers.

The underlying principle of the *Data Iteration Pattern* is the iteration over a *data set* and the execution of an *operation*, where this data set serves as input. The operation may be executed on a set of *resources*, until an optional *completion condition* holds, and either in *parallel* or not. A *data split* parameter defines how to partition the data set among the resources, e. g., to enable parallelism based on MapReduce [1]. A possible value of this data split parameter is a data-specific partitioning mode, e. g., the equal distribution of tuples in a database table. As alternative, this data split may also be defined by means of parameters of a Data Transfer and Transformation Pattern. In a similar way, a *data merge* parameter may define how to integrate the results of the operation back into the data set. Altogether, this pattern may realize the Simulation-oriented Data Interoperability Pattern and the Parameter Sweep Pattern (see Section 4.2).

4 Rule-Based Pattern Transformation

The proposed pattern-based approach to simulation workflow design needs to be complemented by a strategy that transforms abstract patterns into executable workflows. In the following, we discuss basic design considerations of a rule-based transformation of patterns. Afterwards, we illustrate this pattern transformation for the simulation of structure changes in bones discussed in Section 2.1.

4.1 Basic Design Considerations of the Pattern Transformation

The processing model of the rule-based pattern transformation, shown in Fig. 5, extends some basic ideas of Vrhovnik et al. [25]. It traverses the graph of parameterized patterns given by the workflow. For each pattern, the *pattern transformation engine* tries to apply *rewrite rules*. Each rule includes a *condition* part that specifies the circumstances under which the remaining parts of the rule

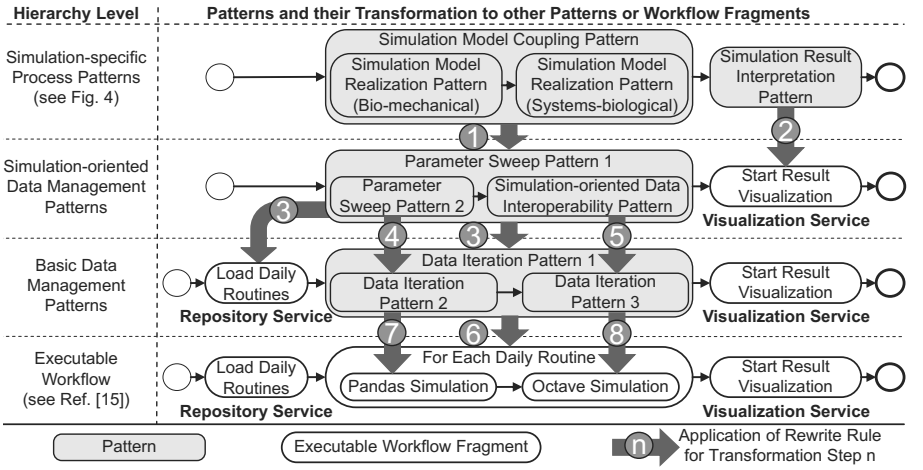


Fig. 6. Main transformation steps from the simulation-specific process model shown in Fig. 4 to the executable workflow illustrated by Reimann et al. [15]

may be applied to the pattern. The *fragment* part of the rule then identifies a template of a workflow fragment via a query to a workflow fragment library [18]. The *action* part defines transformation steps that add implementation details to the workflow fragment. Thereby, this action part may also map parameter values from high to low abstraction levels as shown in Fig. 3. Finally, the rule application replaces the pattern with the resulting workflow fragment. This workflow fragment may either be executable, which finishes the transformation of the pattern, or it may embed other patterns. In the latter case, the pattern transformer applies further rewrite rules to the workflow fragment and to its patterns until all final workflow fragments are executable. In addition, each level of the pattern hierarchy is associated with a *rule sequence* defining the set of rules that may be applied to corresponding patterns and the order in which these rules are tested for applicability. The first rule in a sequence whose condition part evaluates to true is applied to a pattern, while all remaining rules are neglected. Note that rewrite rules are not defined by scientists, but by other experts shown in Fig. 2, i. e., usually by those that also provide relevant workflow fragments.

4.2 Pattern Transformation in the Example Simulation

Now, we discuss how the pattern transformation is applied to the simulation-specific process model depicted in Fig. 4. Fig. 6 shows the main transformation steps over simulation-oriented data management patterns and basic patterns to the executable workflow. The workflow resulting from the simulation-specific process model after the transformation steps 1 and 2 consists of three simulation-oriented data management patterns and one service call. As a result of transformation step 1, the overall Parameter Sweep Pattern 1 implements the control flow of the coupling strategy that is specified as a parameter of the Simulation Model Coupling Pattern shown in Fig. 4. The simulation-specific parameter list

of the Parameter Sweep Pattern (see Table 1) defines the list of daily routines of the relevant person. The pattern sequentially iterates over this list and executes the two embedded patterns for each day. The first embedded pattern, i. e., Parameter Sweep Pattern 2, iterates in parallel over the list of motion sequences of the current day. The operation to be executed for each motion sequence is a service that implements the bio-mechanical Simulation Model Realization Pattern. The subsequent Simulation-oriented Data Interoperability Pattern abstracts from the data integration process required to couple the bio-mechanical and the systems-biological simulation models. It therefore defines the relationships between their mathematical variables as described by Reimann et al. [15]. Some patterns may also be directly mapped to an executable workflow fragment or service without traversing the whole pattern hierarchy depicted in Fig. 2. In our example, this is the case for the Simulation Result Interpretation Pattern, which is mapped to the afore-mentioned final service call (transformation step 2 in Fig. 6).

After transformation steps 3 to 5, the workflow consists of three basic data management patterns and two service calls as shown in Fig. 6. At this abstraction level, the patterns are specified mainly via data-, service-, and ETL-specific parameter values. Finally, rewrite rules of the last transformation steps 6 to 8 transform these basic patterns into executable workflow fragments to create the executable workflow illustrated by Reimann et al. [15].

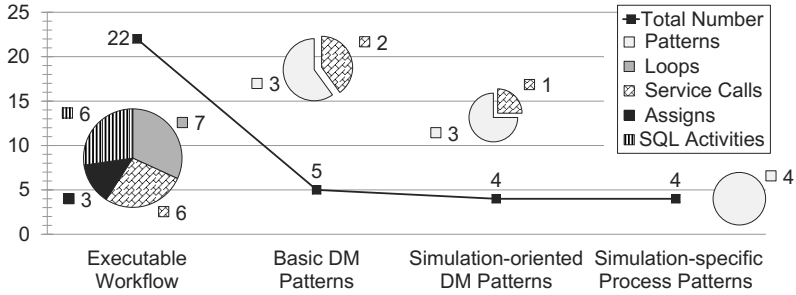
5 Evaluation and Discussion

In this section, we discuss the results of our evaluation of the pattern-based approach to simulation workflow design. This evaluation has been backed by a prototypical implementation, which we have applied to three real-world simulations. This includes the bone simulation introduced in Section 2.1, as well as the examples described by Fehr et al. [3] and Rommel et al. [16]. The prototype is based on the workflow language BPEL [7], on the workflow design tool Eclipse BPEL Designer[3] Version 0.8.0, and on the workflow execution engine Apache Orchestration Director Engine[4] (ODE) Version 1.3.5. We use a PostgreSQL[5] Version 9.2 database system to store the metadata of simulation artifacts depicted in Fig. 3. For the rule-based pattern transformation depicted in Fig. 5, we have extended the JRuleEngine[6] to support rewrite rules and rule sequences for the three example simulations stated above. In the following, we evaluate to what extent our approach fulfills the requirements regarding workflow design and the requirements regarding workflow execution discussed in Section 2.2.

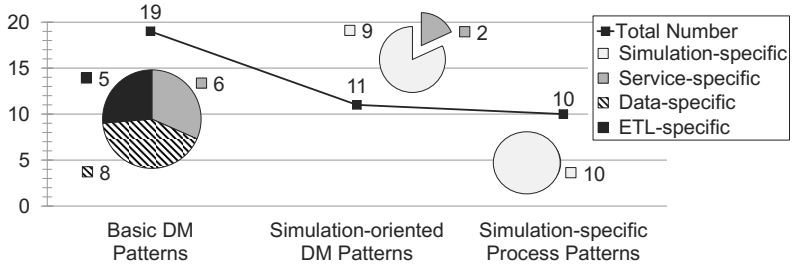
5.1 Assessment of the Abstraction Support

Fig. 7a shows the number of workflow tasks to be specified for the bone simulation introduced in Section 2.1 at the different abstraction levels depicted in

[3] http://www.eclipse.org/bpel/
[4] http://ode.apache.org/
[5] http://www.postgresql.org/
[6] http://jruleengine.sourceforge.net/



(a) Number of workflow tasks



(b) Number of pattern parameters

Fig. 7. No. of workflow tasks and parameters to be specified for the example workflow at different abstraction levels; The partitioning among different types of workflow tasks or parameters is shown in pie charts; "DM" means "data management"

Fig. 6. Furthermore, pie charts in Fig. 7a illustrate the partitioning of these workflow tasks among abstract *patterns* and among different types of executable tasks. With an increasing design complexity, the executable tasks range from simple *loop* structures over *service calls* to complex BPEL *assigns* or *SQL activities* [25]. BPEL assigns implement complex XPath or XQuery expressions, while SQL activities cover even more sophisticated SQL statements [15]. If scientists were designing the executable simulation workflow at the lowest abstraction level as illustrated by Reimann et al. [15], they would need to define a total number of 22 workflow tasks and many complex implementation details. In particular, this would include 9 sophisticated tasks for assigns and SQL activities with complex XPath, XQuery, or SQL statements. Since scientists are typically not familiar with defining such low-level statements, they would not accept this huge design effort. As shown in Fig. 7a, the three pattern abstraction levels depicted in Fig. 6 not only reduce the complexity of workflow tasks, but also their total number to 5 and 4, respectively. This constitutes a total reduction by a factor of 5.5.

We have furthermore analyzed the number and complexity of pattern parameters to be specified at each of the three pattern abstraction levels. Fig. 7b shows these numbers of pattern parameters and illustrates their partitioning according to their complexity, i. e., according to the parameter classes given in Fig. 3. With an increasing abstraction level, the total number of pattern parameters

is nearly divided in half. The basic data management patterns mainly require data- and ETL-specific parameter values, namely 13 of all 19. The other abstraction levels however completely neglect these complex parameter classes. In fact, they mainly consider simulation-specific parameter values that abstract from any implementation details and that are particularly suitable for scientists.

With respect to the other simulation examples described by Fehr et al. [3] and Rommel et al. [16], we have gained nearly similar results as reported in Fig. 7a and 7b. The reduced number and complexity of both workflow tasks and pattern parameters represent a considerable simplification for simulation workflow design. The distribution of these numbers and of the complexity among individual abstraction levels also complies with the skills of different persons involved in workflow design, as it is depicted in Fig. 2. This finally shows the suitability of the separation of concerns introduced by our pattern hierarchy.

5.2 Generic Data Management in Workflows

We also used our prototype to asses whether our patterns are generic enough to cover all relevant data provisioning steps in different simulations [3,16]. In the example described by Rommel et al. [16], we have been able to use the different *simulation-specific process patterns* depicted in Fig. 4 to describe the whole simulation process. This is because these patterns cover common use cases in simulations. In a similar way as shown in Fig. 6, we also mapped them to the Parameter Sweep Pattern and Simulation-oriented Data Interoperability Pattern, as well as to the Data Iteration Pattern. Model reduction, i. e., the reduction of the number of degrees of freedom in a simulation model to reduce computational costs, is another important issue for simulations [3]. This is one of the rare use cases that benefits from a new simulation-specific process pattern incorporating domain-specific parameters. These pattern parameters are (1) the simulation model to be reduced, (2) the concrete reduction method to be employed, (3) the desired number of degrees of freedom, and (4) the required quality of the reduced model. The patterns at lower hierarchy levels nevertheless consider more generic parameters that are independent of the concrete simulation domain. Using our prototype, we again mapped the domain-specific model reduction pattern to the Parameter Sweep Pattern and subsequently to the Data Iteration Pattern.

5.3 Efficient Data Processing and Optimization

The rule-based pattern transformation may cause an overhead when applied at workflow runtime. We have carried out experiments to evaluate this overhead. Our prototype ran on a 64 bit system with 32 GB RAM and an Intel Xeon CPU with 8 cores. We have run the pattern transformation for the 8 transformation steps depicted in Fig. 6 100 times and have determined its average duration. To assume a worst-case scenario, our implementation of the pattern transformation checks 100 rules for each transformation step before the 100^{th} rule is applied. The resulting worst-case duration of the pattern transformation is 0.36 seconds. We have also measured the duration of the executable workflow for the simulation of

Table 3. Overhead of pattern transformation for its worst-case duration of 0.36 seconds

Number of tuples	1 million	10 million	100 million
Workflow duration	140 seconds	1410 seconds	14095 seconds
Worst-case overhead	0.26 %	0.026 %	0.0026 %

structure changes in bones [15]. The workflow and the simulation services it calls ran in parallel on seven computers, each equipped with the hardware resources described above. We have executed the workflow 10 times and for a varying number of tuples as Pandas typically stores them in its output database table, i. e., 1, 10, and 100 million tuples. Table 3 illustrates the respective average workflow durations. Furthermore, it shows the overheads caused by the pattern transformation for its worst-case duration of 0.36 seconds. The maximum overhead is 0.26 % and thus negligible compared to the workflow duration.

The rule-based approach for pattern transformation furthermore enables a seamless integration of rule-based optimization decisions [12,25]. When we increase the number of tuples Pandas stores in its database to 1 billion, our measurements reveal that the workflow realizing the simulation of structure changes in bones [15] lasts about six hours. The ETL processes that are depicted in Fig. 1 and that are part of this workflow may be executed independently for each nodal point of the FEM grid and thereby drastically reduce the data sizes. This makes these ETL processes good candidates to be realized via MapReduce [1,10]. Rewrite rules may choose such more efficient realizations at workflow runtime.

5.4 Monitoring and Provenance Support

While our approach reduces the number and complexity of workflow tasks that are visible to scientists, this may cause a problem for monitoring and provenance. The workflow execution environment is only aware of the more complex executable workflows resulting from the pattern transformation. This may lead to a missing correlation between patterns visible in a workflow design component and audit or provenance information captured by an execution environment. Scientists might be confused when they monitor workflow executions or try to reproduce simulations. A possible solution to this problem could be a view concept on workflows [23]. After each application of a rewrite rule in the pattern transformation, the input pattern is not replaced by the resulting workflow fragment, but it is defined as view on this fragment. Such views enable an aggregation of audit or provenance information to recover their correlation to individual patterns [23]. Furthermore, views also allow scientists to look into specific low-level details of data provisioning tasks if this is required for monitoring purposes.

6 Related Work

According to our focus on data provisioning in workflows, we now discuss related work for data- and artifact-centric business process modeling, for workflow patterns, as well as from the areas of scientific workflows and data integration.

Artifact-centric approaches to business process modeling leverage data as first-class citizens in processes [6,9]. The data is represented by so-called business artifacts that correspond to business-relevant objects, e. g., a customer order or an invoice. The artifacts manage relevant information about these business objects and about their life cycle, and they control how services may be invoked on the objects. As described in Section 3.1, we transfer the core idea of business artifacts to simulation workflow design and to simulation artifacts, e. g., mathematical simulation models. In addition, we augment the artifact-centric idea by our pattern-based approach to workflow design and by the separation of concerns with clearly distinguished abstraction levels as depicted in Fig. 2. From the perspective of scientists, the patterns significantly reduce both the number and complexity of workflow tasks. This is an important issue for the acceptability of simulation workflow technology. Altogether, the combination of an artifact-centric approach with explicitly described patterns of typical workflow tasks considerably improves the way of how to design simulation workflows.

Russel et al. discuss common workflow data patterns [17]. The authors primarily consider fine-grained patterns serving as benchmark to evaluate and compare different workflow languages or workflow systems. For instance, some of the patterns deal with the question whether workflow activities may exchange data by value or by reference. Due to their less technical background, scientists would typically not accept such fine-grained patterns as building blocks for simulation workflow design. Instead, these patterns rather classify implementation details of executable workflow fragments at the lowest level of our pattern hierarchy.

Related work from the scientific workflow domain makes use of ontologies to allow for an abstract parameterization of scientific workflow tasks, i. e., the parameters of these tasks correspond to domain-specific terms or concepts known from the ontologies [11,26]. Furthermore, logical rules may define mappings of such abstract workflow tasks or ontology terms to executable workflows, services, or low-level data types [11,13]. However, the mentioned approaches only moderately reduce the number of workflow tasks that are visible to scientists, while our patterns significantly reduce both the number and the complexity of these tasks. Furthermore, the sole use of ontologies often entails that these approaches are restricted to certain application domains, e. g., life sciences or geophysics. In contrast and as discussed in Section 5.2, our patterns are generic enough to be applied in various domains of simulation applications.

Federated information systems integrate diverse data sources and provide a uniform data schema and a uniform query language [19]. Such uniform schemas and languages however still do not remove the burden from scientists to specify low-level details in terms of complex queries, e. g., based on SQL or XQuery. Furthermore, federated schemas often lack the generality needed to support different scientific domains. Common ETL tools or approaches to schema mapping and matching may help to define ETL processes in basic data management patterns [14]. However and as discussed in Section 3, scientists rather prefer simulation-oriented data management patterns or even simulation-specific process patterns that do not explicitly consider ETL processes.

7 Conclusion and Outlook

Simulations often involve multiple, highly complex data provisioning and data integration tasks, particularly when they are coupled among different scientific domains. In order that scientists are able to concentrate on their core issue, namely on simulation modeling, an abstraction support for this complex data management is indispensable. In this paper, we have presented a novel approach for a pattern-based design of the data provisioning in simulation workflows that conquers the data complexity in such workflows. A pattern hierarchy with different abstraction levels enables a separation of concerns according to the skills of different persons involved in workflow design. For example, scientists select and parameterize a few abstract, simulation-specific process patterns to only describe the core aspects of simulation processes. We use a transformation approach based on a set of rewrite rules that map such abstract process models and patterns onto executable simulation workflows. A prototypical implementation and its application to several simulation applications, e. g., to bio-mechanical and systems-biological problems, has served as basis to evaluate this approach. The patterns significantly reduce both the number and the complexity of workflow tasks that are visible to scientists. The generality of patterns furthermore enables their seamless application in various simulation domains.

As part of future work, we will even increase this generality by working on additional patterns and by applying our approach to other application domains than simulations, e. g., to business processes. To further evaluate the abstraction support and user satisfaction offered by our approach, we will conduct usability studies with real scientists. Furthermore, we will investigate how to integrate optimization decisions into rewrite rules to increase the efficiency of workflows.

Acknowledgments. The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1). We also thank our colleagues Dimka Karastoyanova and Frank Leymann for their valuable cooperation.

References

1. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Communications of the ACM 51(1) (2008)
2. Deelman, E., et al.: Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed Systems. Scientific Programming 13(3) (2005)
3. Fehr, J., et al.: Simulation Process of Flexible Multibody Systems with Non-modal Model Order Reduction Techniques. Multibody System Dynamics 25(3) (2011)
4. Freire, J., et al.: Provenance for Computational Tasks: A Survey. Computing in Science and Engineering 10(3) (2008)
5. Görlach, K., et al.: Conventional Workflow Technology for Scientific Simulation. In: Guide to e-Science. Springer, London (2011)
6. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: Proc. of the 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE), Monterrey, Mexico (2008)
7. Jordan, D., Evdemon, J.: Web Services Business Process Execution Language Version 2.0, OASIS Standard (2007)

8. Krause, R., et al.: Scientific Workflows for Bone Remodelling Simulations. Applied Mathematics and Mechanics 13(1) (2013)
9. Künzle, V., Reichert, M.: PHILharmonicFlows: Towards a Framework for Object-aware Process Management. Journal of Software Maintenance and Evolution: Research and Practice 23(4) (2011)
10. Liu, X., Thomsen, C., Pedersen, T.B.: ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 96–111. Springer, Heidelberg (2011)
11. Ludäscher, B., Altintas, I., Gupta, A.: Compiling Abstract Scientific Workflows into Web Service Workflows. In: Proc. of the 15th International Conference on Scientific and Statistical Database Management, Cambridge, MA, USA (2003)
12. Ogasawara, E.S., et al.: An Algebraic Approach for Data-Centric Scientific Workflows. In: Proc. of the 37th International Conference on Very Large Data Bases (VLDB 2011), Seattle, WA (2011)
13. Radetzki, U., et al.: Adapters, Shims, and Glue – Service Interoperability for in Silico Experiments. Bioinformatics 22(9) (2006)
14. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. International Journal on Very Large Data Bases (VLDB Journal) 10(4) (2001)
15. Reimann, P., Schwarz, H., Mitschang, B.: Data Patterns to Alleviate the Design of Scientific Workflows Exemplified by a Bone Simulation. In: Proc. of the 26th International Conference on Scientific and Statistical Database Management (2014)
16. Rommel, J.B., Kästner, J.: The Fragmentation-Recombination Mechanism of the Enzyme Glutamate Mutase Studied by QM/MM Simulations. Journal of the American Chemical Society 26(133) (2011)
17. Russell, N., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Workflow Data Patterns: Identification, Representation and Tool Support. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 353–368. Springer, Heidelberg (2005)
18. Schumm, D., et al.: Process Fragment Libraries for Easier and Faster Development of Process-based Applications. Systems Integration 2(1) (2011)
19. Sheth, A.P.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. In: Proc. of the 17th International Conference on Very Large Data Bases (VLDB 1991), Barcelona, Spain (1991)
20. Shoshani, A., Rotem, D.: Scientific Data Management: Challenges, Technology, and Deployment. Computational Science Series. Chapman & Hall (2009)
21. Simitsis, A., et al.: Optimizing Analytic Data Flows for Multiple Execution Engines. In: Proc. of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD 2012), Scottsdale, AZ, USA (2012)
22. 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, Prague, Czech Republic (2010)
23. Sonntag, M., et al.: Views on Scientific Workflows. In: Proc. of the 10th International Conference on Perspectives in Business Informatics Research (2011)
24. Taylor, I., Deelman, E., Gannon, D.: Workflows for e-Science - Scientific Workflows for Grids. Springer, London (2007)
25. 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)
26. Wolstencroft, K., et al.: The myGrid Ontology: Bioinformatics Service Discovery. Int. Journal on Bioinformatics Research and Applications 3(3) (2007)