Data Provisioning in Simulation Workflows

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik
sowie dem Stuttgart Research Centre for Simulation Technology
der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Peter Reimann

aus Stuttgart

Hauptberichter:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Mitberichter:	Prof. Dr. rer. nat. Bertram Ludäscher
Tag der mündlichen Prüfung:	15. Dezember 2016

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart
2017

Acknowledgements

This thesis is dedicated to several people who have supported my work on it in various ways. First of all, I would especially like to thank my doctoral advisor, Prof. Dr.-Ing. habil. Bernhard Mitschang, who gave me the opportunity to work on this challenging topic in his research group. His constructive support, valuable advice, and many interesting discussions have significantly helped me to conduct my scientific research and to grow both as a researcher and as a person.

Furthermore, my sincere thanks go to the co-reviewer, Prof. Dr. rer. nat. Bertram Ludäscher, for spending time reading this thesis and for giving helpful suggestions and comments.

I have been working on this thesis while being employed as a scientific staff member at the Institute of Parallel and Distributed Systems and at the Cluster of Excellence in Simulation Technology (SimTech) of the University of Stuttgart. So, I would also like to thank the German Research Foundation (DFG) for financial support of the accompanying research project within the SimTech cluster.

It was a pleasure and a great time working together and having stimulating discussions with many colleagues at the University of Stuttgart. Special thanks go to Holger Schwarz for his extensive and reliable collaboration, as well as for his continuous and patient effort to improve the quality of my research. Furthermore, I would like to thank several persons who have published papers with me. Besides Bernhard Mitschang and Holger Schwarz, this mainly includes Pascal Hirmer, Dimka Karastoyanova, Jan Königsberger, Frank Leymann, Jorge Minguez, Michael Reiter, Stefan Silcher, Tim Waizenegger, Matthias Wieland, and Sema Zor. Likewise, many other current and former colleagues of the department Applications of Parallel and Distributed Systems helped me via miscellaneous advises and discussions, in particular Nazario Cipriani, Christoph Gröger, Philipp Janowski, Carlos Lübbe, Sylvia Radeschütz, Oliver Schiller, Christoph Stach, and Marko Vrhovnik. On this note, I would also like to thank the plenty collaboration partners from the SimTech cluster. This especially holds for Jörg Fehr, Fabian Franzelin, Katharina Görlach, Michael Hahn, Dimka Karastoyanova, Robert Krause, Johannes Kästner, Frank Leymann, Dirk Pflüger, Michael Reiter, Syn Schmitt, David Schumm, Mirko Sonntag, Karolina Vukojevic-Haupt, and Andreas Weiß. Finally, particular thanks go to Pascal Hirmer, Oliver Kopp, and Holger Schwarz for spending their precious time proof-reading this document and for their valuable comments.

I would also like to acknowledge the work of several students of the University of Stuttgart. In particular, my thanks go to the following students for developing prototypes of the concepts and methods proposed by this thesis: Christian Ageu, Stavros Aristidou, Andreas Bohrn, Daniel Brüderle, Marzieh Dehghanipour, Alexander Gessler, Michael Hahn, Eva Hoos, Wolfgang Hüttig, Savas Kalyoncu, Christoph Müller, Henrik Pietranek, René Rehn, Simon Remppis, Victor Riempp, Michael Schneidt, Xi Tu, Patrick von Steht, Florian Wagner, and Firas Zoabi.

In addition, I thank the administrations of the Institute of Parallel and Distributed Systems and of the Cluster of Excellence in Simulation Technology for their support in organizational and technical issues. Special thanks go to Ralf Aumüller, Manfred Rasch, Annemarie Rösler, Eva Strähle, Barbara Teutsch, and Christine Well who always helped me in case of questions or problems.

Further thanks go to Bertram Ludäscher again for hosting me during my visit at the University of Illinois at Urbana-Champaign and for working with me on aspects related to scientific workflows and provenance. I had a great time and learned much during this visit. Especially the following people cordially welcomed me and assisted me in many professional and organizational issues: Yang Cao, Janet Eke, Timothy McPhillips, Qian Zhang, as well as the whole GSLIS Help Desk team. Special thanks go to my new friends Ruth and David Krehbiel for accommodating me and thereby providing a pleasant personal environment.

Finally, I want to express my sincere gratitude to the closest persons to whom this thesis is especially dedicated: my parents Christa and Gunter Reimann. They always let me go my way and likewise continuously supported and encouraged me during my work on this thesis. Thank you from the bottom of my heart! I would also like to thank the rest of my family and all my friends who were by my side during all the years. Each of them helped me a lot, even if they are not aware of it.

Peter Reimann

Stuttgart, December 20, 2016

Contents

List of Abbreviations

Apache ODE	Apache Orchestration Director Engine
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
BPEL	Business Process Execution Language
BPM	Business Process Modeling
CAD	Computer-aided Design
CAE	Computer-aided Engineering
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Comma-separated Values
DAO	Data Access Object
DSL	Domain-specific Language
DUNE	Distributed and Unified Numerics Environment
ETL	Extraction, Transformation, Load (of data)
FEA	Finite Element Analysis
FEM	Finite Element Method
FLWOR	For, Let, Where, Order by, Return (XQuery statement)
FTP	File Transfer Protocol
grep	Globally search a Regular Expression and Print
GUI	Graphical User Interface
HPC	High Performance Computing
IT	Information Technology
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
NIST	National Institute of Standards and Technology
OASIS	Organization for the Advancement of Structured Information Standards
OGSA-DAI	Open Grid Services Architecture - Data Access and Integration framework
ODE	Ordinary Differential Equation
UDF	User-Defined Function
PDE	Partial Differential Equation
PDM	Product Data Management
PSM	Persistent Stored Modules

RAM	Random Access Memory
REST	Representational State Transfer
SC SimTech	Stuttgart Center for Simulation Sciences
SCP	Secure Copy
SDM system	Simulation Data Management system
SIMPL	SimTech, Information Management, Processes, and Languages
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSH	Secure Shell
sWfMS	Scientific workflow management system
TOSCA	Topology and Orchestration Specification for Cloud Applications
UDDI	Universal Description, Discovery, and Integration
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language
XPath	XML Path Language
XQuery	XML Query Language
XSD	XML Schema Definition
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations
VarChar	Variable Character String
W3C	World Wide Web Consortium
WfMS	Workflow management system
WS-Addressing	Web Services Addressing
WS-BPEL	Web Services Business Process Execution Language
WSDL	Web Services Description Language

Abstract

Computer-based simulations become more and more important in both industry and science. In particular, they may imitate real-world experiments or other systems, which would otherwise be too expensive or not feasible at all [Har96]. For instance, product development processes may benefit from simulations as virtual and economic alternatives to physically realized product tests [Hau81, CMPW07]. The usual way to imitate a real-world experiment or system is to mathematically model its temporal and/or spatial behavior, e. g., via a system of differential equations [BDH11]. Computer-based simulations often require the discretization of such mathematical simulation models using numerical methods [Hum90], such as the Finite Element Method (FEM) [ZTZ13]. Certain simulation tools, e. g., Matlab[1], may then be used to implement corresponding numerical calculations.

Nowadays, workflow technology is increasingly adopted to control the execution of computer-based simulations and of their numerical calculations [GSK$^+$11]. Corresponding simulation workflows mainly orchestrate the interaction with involved simulation tools. The input data needed by these tools often come from diverse data sources that manage their data in a multiplicity of proprietary formats [RRS$^+$11, RS14]. Due to this data heterogeneity, simulation workflows additionally have to compose many complex data provisioning tasks. These tasks filter and transform the input data in such a way that the used simulation tools are able to ingest them [RLSR$^+$06]. This is one reason why the effort to be spent on designing simulation workflows is considerably high [RLSR$^+$06, CBL11].

One prevalent trend in simulation research even significantly increases this effort to be spent on designing data provisioning tasks in workflows. Scientists from several scientific domains aim at coupling their diverse mathematical simulation models [GZC14]. This makes it possible to cover various levels of detail in simulation calculations, thereby increasing the potential to produce precise results [Gat14]. For instance, a simulation of structure changes in bones is mainly governed by a bio-mechanical simulation model representing a bone on a macroscopic tissue scale [Kra14]. To make this simulation more realistic, the bio-mechanical model is coupled with a systems-biological model, which also considers how microscopic cell interactions influence the bone structure. Different scientific domains frequently use likewise different simulation tools to implement their simulation models [KAK$^+$13, RSM14b]. Individual tools rely on proprietary

[1]Matlab: `http://www.mathworks.com/products/matlab/index.html`

ways to manage their data, which even increases the heterogeneity of data sources and data formats. This makes it difficult to exchange data between the coupled simulation models and simulation tools. To make things worse, the various levels of detail involved in coupled simulation calculations require to aggregate, interpolate, or extrapolate data accordingly [Gat14]. All this multiplies the complexity of implementing data transformations in simulation workflows.

Nowadays, scientists conducting simulations typically need to design their simulation workflows on their own. Hence, they have to implement many low-level data transformations that realize the data provisioning for and the data exchange between simulation tools. For instance, a workflow realizing the data exchange between the above-mentioned bio-mechanical and systems-biological simulations composes more than 15 workflow tasks [RSM14b]. Around half of these tasks oblige scientists to specify sophisticated SQL, XPath, or XQuery statements. However, scientists usually do not have adequate skills in data management or data engineering. So, they waste time for workflow design, which hinders them to concentrate on their core issue, i. e., the simulation itself.

This thesis introduces several novel concepts and methods that significantly alleviate the design of the complex data provisioning in simulation workflows. Thereby, these concepts and methods make simulation workflow design especially tailor-made for scientists, which finally boosts their productivity. Most parts of this thesis correspond to revised and composite versions of previous author publications [RRS$^+$11, RSM11, RS13b, RS14, RSM14a, RSM14b, RWWS14].

The first introduced method addresses the issue that most existing workflow systems or related simulation tools offer diverse and proprietary data provisioning techniques. Some systems rely on application-specific services [TDG07, GSK$^+$11], while others provide scientists with customized workflow activities [LAB$^+$06, VSRM08]. To support as many applications as possible, most of the systems even offer multiple of these services or workflow activities [CBL11]. So, scientists are frequently overwhelmed with a multiplicity and diversity of available data provisioning techniques. This thesis discusses how to conquer this multiplicity and diversity by classifying available techniques into a small set of representative concepts. The resulting concepts are then compared with each other considering relevant functional and non-functional requirements for data provisioning in simulation workflows. One outcome of the classification and comparison is a set of guidelines that assist scientists in choosing proper data provisioning techniques for their workflows. In addition, this thesis discusses the features a workflow system has to support in order that scientists may apply the guidelines in practice. This correspondingly allows for deriving a set of essential missing features existing workflow systems do not support. Hence, another contribution is an extended simulation workflow system that offers all these mandatory features in a holistic way. Most extensions covered by this simulation workflow system implement the remaining concepts and methods introduced by this thesis.

One missing feature of existing workflow systems is that they lack a generic solution to data provisioning in simulation workflows. More precisely, they often do not support all kinds of data resources or data management operations required by computer-based simulations. The most generic solution among related work is offered by ETL technology enabling processes for extracting, transforming, and loading data [KRRT98]. This thesis therefore transfers the general ideas of ETL technology to conventional workflow technology [MMLW05]. The resulting ETL workflow approach offers a set of extensions to workflow languages that constitute the necessary generic solution in terms of supported data resources and operations [RRS+11]. However, ETL technology or underlying ETL tools usually overwhelm scientists again by offering a multiplicity of diverse ETL operators to be combined in data provisioning workflows. Hence, the ETL workflow approach proposed by this thesis is designed in a way that does not entail this decisive drawback of ETL technology. In fact, the corresponding extensions to workflow languages only cover four reasonable types of generic data management activities. Nevertheless, these data management activities allow for specifying a broad range of data management operations for any data resource. In particular, they support any operation that may be specified via the command languages offered by involved data resources, e. g., SQL statements or shell commands. This thesis additionally shows that the proposed activities are sufficient to design the data provisioning for virtually all simulation examples of various scientific domains.

The proposed generic data management activities still do not remove the burden from scientists to specify many complex data management operations. In fact, scientists need to describe such operations using the low-level command languages offered by involved data resources. Hence, this thesis introduces a novel pattern-based approach that even further enhances the abstraction support for simulation workflow design [RS13b, RS14, RSM14a, RSM14b]. Instead of specifying many workflow tasks, scientists only need to select a few number of abstract patterns to describe the high-level simulation process they have in mind. These patterns represent uses cases that are especially meaningful to scientists, e. g., coupling simulation models. Furthermore, scientists are familiar with the parameters to be specified for the patterns, because these parameters correspond to terms or concepts that are related to their domain-specific simulation methodology. Finally, a rule-based transformation approach offers flexible means to transform such high-level processes and patterns into executable simulation workflows. Another major feature of this new approach is a pattern hierarchy arranging different kinds of patterns according to clearly distinguished abstraction levels. This facilitates a holistic separation of concerns in workflow design. It provides a systematic framework to incorporate different kinds of persons and their various skills, e. g., not only scientists, but also data engineers. Altogether, this conquers the data complexity associated with simulation workflows, which allows scientists to concentrate on their core issue again, namely on the simulation itself.

The last contribution is a complementary optimization method to increase the performance and robustness of simulation workflows or other data-intensive workflows. Related approaches mainly aim at accelerating operations that process data externally to a workflow [VSS$^+$07, DG08, ZBKL10, LTP11, OdOV$^+$11, SWCD12, LQ14]. Furthermore, these approaches often assume and exploit dataflow-based descriptions of workflows to make proper optimization decisions. The novel method proposed by this thesis therefore addresses the neglected issue how to optimize local data processing tasks within workflows that are governed by control-flow-oriented languages [RSM11]. It introduces various techniques that partition relevant local data processing tasks between the components of a workflow system in a smart way. Thereby, such tasks are either assigned to the workflow execution engine or to a tightly integrated local database system. This thesis furthermore discusses the results of evaluating the effectiveness of these techniques via a set of experiments.

In summary, the concepts and methods proposed by this thesis fill several gaps in current research regarding data provisioning in simulation workflows. In particular, they provide a holistic abstraction alleviating the complex design of such workflows and this way making simulation workflow design especially tailor-made for scientists. This has also been confirmed by profound evaluations, which have been conducted while working on the proposed concepts and methods. These evaluations have been backed up by elaborate prototypical implementations and by their application to several real-world simulations, e. g., to the coupled simulation of structure changes in bones mentioned above [Kra14].

Deutsche Zusammenfassung

Computerbasierte Simulationen gewinnen sowohl im industriellen als auch im wissenschaftlichen Bereich mehr und mehr an Bedeutung. Dabei können sie insbesondere genutzt werden, um reale Experimente wie z. B. einen Crashtest nachzubilden, welche ansonsten zu teuer oder gar nicht durchführbar wären [Hau81, Har96]. Die Nachbildung eines realen Experiments erfolgt typischerweise über dessen zeit- und raumabhängige mathematische Modellierung, beispielsweise über ein System von Differenzialgleichungen [BDH11]. Solche mathematischen Simulationsmodelle müssen häufig mithilfe numerischer Methoden wie der Finite-Elemente-Methode (FEM) diskretisiert werden, damit sie von einem Computer berechnet werden können [Hum90, ZTZ13]. Simulationstools, z. B. Matlab[2], können dann genutzt werden, um entsprechende numerische Berechnungen zu implementieren.

Mittlerweile werden die für computerbasierte Simulationen notwendigen Interaktionen mit Simulationstools über Simulationsworkflows koordiniert [GSK+11]. Die Eingabedaten für Simulationstools kommen dabei häufig von verschiedenartigen Datenquellen, welche ihre Daten in einer Vielzahl proprietärer Formate verwalten [RRS+11, RS14]. Auf Grund dieser Heterogenität der Datenlandschaft enthalten Simulationsworkflows noch zusätzlich viele komplexe Schritte für die Datenbereitstellung, welche die Eingabedaten derart filtern und transformieren, dass sie von den genutzten Simulationstools eingelesen werden können [RLSR+06]. Dies ist eine Ursache für den hohen Aufwand, der in die Entwicklung von Simulationsworkflows gesteckt werden muss [RLSR+06, CBL11].

Ein weitverbreiteter Trend im Bereich computerbasierter Simulationen erhöht diesen Aufwand für die Entwicklung von Simulationsworkflows sogar noch. Wissenschaftler von verschiedenen Anwendungsgebieten versuchen, ihre unterschiedlichen mathematischen Simulationsmodelle miteinander zu koppeln [GZC14]. Dadurch können sie mehrere Detailgrade in Simulationsberechnungen abdecken, was wiederum das Potenzial erhöht, präzise und realitätsnahe Ergebnisse zu erhalten [Gat14]. Eine Simulation von Strukturänderungen in Knochen koppelt beispielsweise eine biomechanische Gewebesimulation mit einer präziseren systembiologischen Simulation, welche auch die mikroskopische Interaktion zwischen Knochenzellen berücksichtigt [Kra14]. Unterschiedliche Anwendungsgebiete verwenden typischerweise ebenso unterschiedliche Simulationstools, um ihre Modelle zu berechnen [KAK+13, RSM14b]. Verschiedene Tools verwalten ihre Daten

[2]Matlab: `http://de.mathworks.com/products/matlab/`

dabei auf jeweils proprietäre Weise, was wiederum die Heterogenität an Datenquellen und Datenformaten erhöht. Dies erschwert genauso den Datenaustausch zwischen gekoppelten Simulationsmodellen sowie die Implementierung dafür notwendiger Datentransformationen. Insbesondere erfordern die unterschiedlichen Detailgrade in verschiedenen Simulationsberechnungen komplexe Aggregationen, Interpolationen oder Extrapolationen der auszutauschenden Daten [Gat14].

Die an den Simulationsergebnissen interessierten Wissenschaftler müssen ihre Simulationsworkflows i. d. R. selbst entwickeln. Folglich müssen sie dabei auch viele komplexe Datentransformationen implementieren, welche die Datenbereitstellung für und den Datenaustausch zwischen Simulationsmodellen bzw. Simulationstools realisieren. Ein Workflow für den Datenaustausch zwischen den oben erwähnten biomechanischen und systembiologischen Simulationen besteht z. B. aus mehr als 15 Workflowschritten, von denen ungefähr die Hälfte die Spezifikation komplexer SQL-, XPath- oder XQuery-Anweisungen erfordern [RSM14b]. Hierfür verschwenden Wissenschaftler viel zu viel Zeit, die sie eigentlich für ihre Kernfragestellung aufbringen möchten, nämlich die Simulation selbst.

Die vorliegende Dissertation führt mehrere neuartige Konzepte und Methoden ein, welche die Entwicklung der komplexen Datenbereitstellung in Simulationsworkflows maßgeblich erleichtern und damit die Produktivität von Wissenschaftlern erhöhen. Einzelne Teile dieser Dissertation entsprechen dabei einer überarbeiteten und zusammengesetzten Version früherer Veröffentlichungen des Autors [RRS$^+$11, RSM11, RS13b, RS14, RSM14a, RSM14b, RWWS14].

Die erste neu eingeführte Methode befasst sich mit dem Problem, dass die meisten existierenden Workflowsysteme sehr viele proprietäre Datenbereitstellungstechniken anbieten, z. B. unterschiedliche anwendungsspezifische Services oder spezielle Workflowaktivitäten [LAB$^+$06, TDG07, VSRM08]. Wissenschaftler werden daher häufig von dieser Vielzahl an diversen Datenbereitstellungstechniken überwältigt [CBL11]. Die vorliegende Dissertation diskutiert, wie diese Vielzahl und Diversität der Techniken durch deren Klassifikation in eine kleine Menge repräsentativer Konzepte beherrschbar gemacht wird. Die resultierenden Konzepte werden außerdem unter Berücksichtigung relevanter funktionaler und nichtfunktionaler Anforderungen miteinander verglichen. Ein Ergebnis dieses Vergleichs ist ein Leitfaden, welcher Wissenschaftler bei der Wahl passender Datenbereitstellungstechniken unterstützt. Weiterhin diskutiert die vorliegende Arbeit, welche Features Workflowsysteme aufbieten müssen, damit dieser Leitfaden in der Praxis angewendet werden kann. Dies schließt ebenso die Diskussion derjenigen Features ein, die von existierenden Systemen nicht angeboten werden. Ein weiterer Beitrag dieser Dissertation entspricht daher einem erweiterten Workflowsystem, das alle relevanten Features auf ganzheitliche Weise unterstützt. Die meisten Erweiterungen dieses Workflowsystems implementieren die weiteren neu eingeführten Konzepte und Methoden.

Existierende Workflowsysteme bieten insbesondere keine generische Lösung für die Datenbereitstellung in Simulationsworkflows. Genauer gesagt unterstützen sie häufig nicht alle von computerbasierten Simulationen benötigten Arten von Datenressourcen oder Datenmanagementoperationen. Unter verwandten Arbeiten finden sich nur ETL-Technologien (Extraktion, Transformation und Laden von Daten), welche adäquat generisch einsetzbar sind [KRRT98]. Die vorliegende Dissertation überträgt daher die grundlegenden Ideen von ETL-Technologien auf konventionelle Workflowsprachen [MMLW05]. Die resultierenden Erweiterungen dieser Workflowsprachen bieten die benötigte generische Lösung bzgl. unterstützter Datenressourcen und Operationen [RRS+11]. Allerdings überwältigen ETL-Technologien bzw. gängige ETL-Tools Wissenschaftler wieder mit einer Vielzahl diverser ETL-Operatoren, welche in Datenbereitstellungsworkflows verwendet werden können. Das von der vorliegenden Dissertation eingeführte Konzept berücksichtigt diesen Aspekt, indem die Erweiterungen von Workflowsprachen lediglich vier sinnvolle Arten von generischen Datenmanagementaktivitäten umfassen. Nichtsdestoweniger ermöglichen diese Datenmanagementaktivitäten die Spezifikation vielfältiger Datenmanagementoperationen für beliebige Datenressourcen. Insbesondere unterstützen sie jede Operation, welche über die von Datenressourcen angebotenen Anfrage- oder Befehlssprachen spezifiziert werden können, z. B. SQL bzw. Shell-Sprachen. Die vorliegende Dissertation zeigt zudem, dass solche Aktivitäten ausreichen, um die Datenbereitstellung für beliebige Simulationsbeispiele aus verschiedenen Anwendungsgebieten zu spezifizieren.

Mit den vorgeschlagenen Datenmanagementaktivitäten müssen Wissenschaftler nach wie vor viele komplexe Datenmanagementoperationen über die ihnen eher unbekannten Anfrage- bzw. Befehlssprachen von Datenressourcen spezifizieren. Aus diesem Grund führt die vorliegende Dissertation einen neuartigen Pattern-basierten Ansatz ein, welcher die Abstraktionsunterstützung für die Entwicklung von Simulationsworkflows noch deutlich verbessert [RS13b, RS14, RSM14a, RSM14b]. Anstatt unzählige Workflowschritte umzusetzen, müssen Wissenschaftler nur eine kleine Anzahl abstrakter Patterns auswählen, um die wichtigsten Schritte ihres Simulationsprozesses zu beschreiben.

Diese Patterns verkörpern Anwendungsfälle, welche speziell auf Wissenschaftler zugeschnitten sind, z. B. die Kopplung von Simulationsmodellen. Weiterhin können Wissenschaftler die Patterns über Parameter spezifizieren, die ihnen bereits aus ihrer anwendungsspezifischen Simulationsmethodologie vertraut sind. Ein regelbasierter Transformationsansatz bietet schließlich flexible Mechanismen, um solche abstrakten Simulationsprozesse und Patterns auf ausführbare Workflows abzubilden. Ein weiterer wesentlicher Beitrag ist eine Pattern-Hierarchie, welche verschiedene Patterns gemäß klar voneinander abgrenzbaren Abstraktionsebenen anordnet. Dies bietet ein Rahmenwerk, um nicht nur Wissenschaftler, sondern auch andere Personen mit speziellen Fähigkeiten aus Bereichen wie der Datenverwaltung in die Entwicklung von Simulationsworkflows mit einzubeziehen.

Der letzte Beitrag ist eine komplementäre Optimierungsmethode, um die Effizienz und Robustheit von Simulationsworkflows oder anderen datenintensiven Workflows zu erhöhen. Verwandte Ansätze zielen hauptsächlich darauf ab, Operationen zu optimieren, welche Daten extern zu einem Workflow verarbeiten [VSS$^+$07, DG08, ZBKL10, LTP11, OdOV$^+$11, SWCD12, LQ14]. Weiterhin erwarten diese Ansätze häufig eine datenflussbasierte Beschreibung von Workflows, um korrekte Optimierungsentscheidungen treffen zu können. Die von der vorliegenden Dissertation vorgeschlagene Methode befasst sich daher mit der vernachlässigten Fragestellung der Optimierung der lokalen Datenverarbeitung in Workflows, welche über einen Kontrollfluss modelliert werden [RSM11]. Diese Methode führt einige neuartige Techniken ein, die relevante lokale Datenverarbeitungsschritte in zielgerichteter Weise zwischen den Komponenten eines Workflowsystems aufteilen. Solche Datenverarbeitungsschritte werden dabei entweder einer Workflow-Engine oder einem in das Workflowsystem integrierten Datenbanksystem zugewiesen. Weiterhin diskutiert die vorliegende Dissertation Ergebnisse einer auf Experimenten basierenden Evaluation dieser Techniken.

Zusammenfassend schließen die von der vorliegenden Dissertation eingeführten Konzepte und Methoden mehrere Forschungslücken bezogen auf die Datenbereitstellung in Simulationsworkflows. Insbesondere stellen sie Wissenschaftlern eine ganzheitliche Abstraktion zur Verfügung, welche die Entwicklung solcher Workflows maßgeblich vereinfacht. Dies wurde auch durch eine tiefgreifende Evaluation der vorgeschlagenen Konzepte und Methoden bestätigt. Diese Evaluation wurde mithilfe gut ausgearbeiteter prototypischer Implementierungen sowie deren Anwendung auf mehrere reale Simulationen, wie z. B. auf die oben erwähnte Simulation von Strukturänderungen in Knochen, durchgeführt.

Chapter 1

Introduction

Nowadays, computer-based simulations are increasingly adopted in both industry and science. In particular, simulations may imitate the spatial and/or temporal behavior of usually real-world experiments, processes, or other systems [Har96]. The main motivation is to provide additional and better means to investigate the imitated systems or related issues. For example, computer-based simulations may serve as virtual and economic alternatives to physically realized and expensive experiments, such as crash tests in product development [Hau81]. Due to this increased interest in simulations, many interdisciplinary research initiatives have been established to foster existing or to develop new simulation approaches and technologies. An example is the Stuttgart Center for Simulation Sciences (SC SimTech)[1].

Typically, mathematical simulation models represent the objects or processes to be simulated as well as their behavior, e. g., based on a system of differential equations [BDH11]. Such mathematical models often describe this behavior in a continuous way. So, they cannot be directly realized and solved by computers, i. e., in a computer-based simulation. Instead, they need to be discretized using numerical methods, e. g., the Finite Element Method (FEM) [ZTZ13]. These numerical discretizations may then be implemented by simulation tools or frameworks, such as DUNE (Distributed and Unified Numerics Environment)[2].

Another related research area deals with scientific workflows [TDG07]. These workflows compose a set of tools or services to implement scientific applications, such as experiments, data analyses, or computer-based simulations. The input

[1]SC SimTech: `http://www.simtech.uni-stuttgart.de/index.en.html`
[2]DUNE: `http://www.dune-project.org/`

data needed by these tools or services often come from diverse data sources that manage their data in a multiplicity of proprietary data formats [RLSR+06, RRS+11, RS14]. For that purpose, scientific workflows have to carry out many complex data provisioning tasks, which filter and transform heterogeneous input data in such a way that underlying tools or services can properly ingest them. This is one reason why workflow developers have to spend a considerably high effort for designing scientific workflows [RLSR+06, CBL11].

The focus of this thesis is on *simulation workflows*, which are a sub-area of scientific workflows in that they control the execution of computer-based simulations [GSK+11]. So, they typically compose a set of long-running numerical calculations that realize mathematical simulation models. Similar as general scientific workflows, simulation workflows also carry out many complex data provisioning tasks that prepare input data as needed by the respective simulation tools or frameworks [RS14]. However, one prevalent trend in simulation research even increases the resulting effort to be spent on workflow design. To make simulations more realistic, scientists couple mathematical simulation models from several scientific domains in simulation workflows. For example, a simulation of structure changes in bones combines a bio-mechanical tissue simulation with a more fine-grained systems-biological approach [KSR+11, Kra14]. Coupling different simulation models allows scientists to cover various levels of granularity in simulation calculations [GZC14]. This in turn increases the potential to produce more precise results and to better understand the system to be simulated.

In coupled simulations, the result data of one simulation model are often used as input for other simulation models, and these models are usually implemented by different simulation tools [KAK+13, RSM14b]. Different simulation models and simulation tools typically rely on different solutions for data handling, which makes the implementation of data transformations even more sophisticated. Furthermore, the various levels of granularity involved in coupled simulation calculations make it necessary to aggregate, interpolate, or extrapolate data accordingly [Gat14]. This even multiplies the already high complexity of designing data provisioning tasks in simulation workflows.

Today, scientists or engineers often want or even need to design their simulation workflows on their own. Hence and as described above, they have to specify many complex, low-level details of data provisioning. So, scientists or engineers waste time for workflow design, which hinders them to concentrate on their core issues, namely the development of mathematical simulation models, the execution of simulation calculations, and the interpretation of their results. This complexity of simulation workflow design constitutes the major motivation of this thesis.

The main contribution of this thesis is a set of novel concepts and methods that make simulation workflow design especially tailor-made for scientists or engineers conducting simulations. In particular, these concepts and methods significantly alleviate the design of the complex data provisioning in simulation workflows. This way, scientists or engineers may focus on their actual core issues again, which in turn considerably boosts their productivity.

In the following section, the major motivation considered by this thesis is exemplified by means of the running example of simulating structure changes in bones [KSR$^+$11, KAK$^+$13, Kra14]. Afterwards, Section 1.2 comprises a discussion of more detailed challenges that have to be considered by concepts and methods for data provisioning in simulation workflows. Furthermore, the state of current work is summarized and how this current work meets the discussed challenges. Concrete contributions of this thesis and of the proposed concepts and methods are illustrated in Section 1.3. Section 1.4 depicts the outline of this thesis.

1.1 Running Example and Major Motivation

The general motivation to simulate structure changes in bones comes from medical science. For example, such a simulation may be used to support healing processes after bone fractures. One possible approach is to combine ideas from biology and mechanical engineering into a bio-mechanical simulation model [KME10, EKM11, Kra14]. This model describes the mechanical behavior of a bone on a macroscopic tissue scale. The Pandas software[3], which is based on the FEM, offers its numerical implementation. Figure 1.1 shows the main steps of a workflow realizing this bio-mechanical simulation [RS14]. In the first four steps (blue in Figure 1.1), the workflow prepares all input data the Pandas software requires. This includes (1) the geometrical shape of the bone, (2) some material parameters, (3) boundary conditions, and (4) certain FEM-specific parameters. These data come from unstructured text files, XML databases, CSV-based files (comma-separated values), and SQL databases, respectively. The workflow filters appropriate data from these diverse data sources and transforms them into proprietary text files and CSV-based files the Pandas software can handle.

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

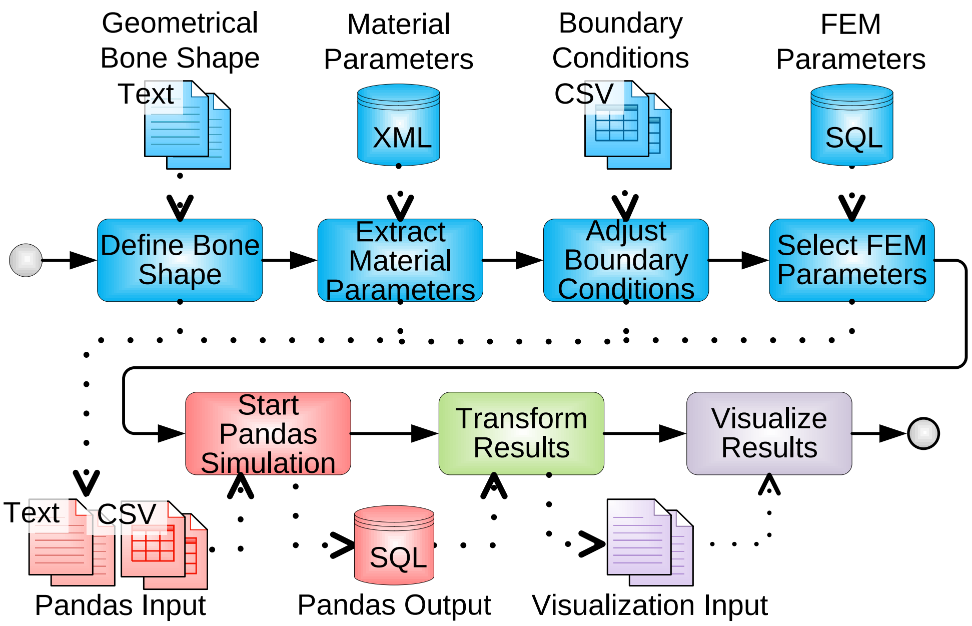


Figure 1.1: Main steps of a bio-mechanical simulation workflow to investigate structure changes in bones; cf. [RS14].

During the red workflow step in Figure 1.1, Pandas calculates several output variables of the bio-mechanical simulation model. This includes, amongst others, the internal stress distribution within the bone tissue and the medically important bone density [Kra14]. For each numerical time step and at each evaluation point of the numerical space discretization, Pandas calculates up to 20 of such variables and stores the resulting values in a SQL database. Finally, scientists want to interpret this simulation outcome by means of a visualization (purple step in Figure 1.1). For that purpose, the workflow filters relevant data from the SQL database (green step). Furthermore, it transforms these data into file formats the used visualization tool is able to read.

The bio-mechanical simulation workflow shown in Figure 1.1 carries out five complex data provisioning tasks accessing several heterogeneous data sources (blue and green workflow steps). To design this workflow, scientists have to specify many low-level details of data management operations. This mainly concerns operations to filter appropriate data from the data sources and to convert the data into different formats. Here, scientists need to define complex SQL, XPath, or XQuery statements, or they need to employ scripting or programming languages to implement the operations. Scientists have much knowledge in their simulation domain, but rather limited skills regarding the implementation of complex and low-level data management operations [RLSR+06, CBL11]. In particular, they are often not familiar with languages such as SQL, XPath, or XQuery. Hence, they waste time when trying to adopt and learn these languages, but they actually want to spend this time on their core issue, namely the simulation itself.

The resulting complexity of workflow design is even multiplied in case the output data of one simulation is used as input of another one. The bio-mechanical simulation model does not consider any cellular reactions within the bone tissue. This is however necessary to calculate the bone structure as precisely as possible. A solution to this problem is to couple the bio-mechanical model with a systems-biological model [KSR+11, KSW+12, Kra14]. This systems-biological model gets the internal stress distribution calculated by Pandas as part of its input [Kra14]. It then determines the precise change of the bone density 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 often separate simulation tools are employed for each of the models [KAK+13]. The systems-biological simulation may be implemented by GNU Octave[4]. This tool requires only a few variables calculated by Pandas. In addition, it requires its data in CSV-based files and in a different level of granularity regarding the time scale. A workflow realizing the data exchange between Pandas and Octave composes more than 15 workflow tasks [RSM14b]. Around half of them are low-level tasks that (1) filter appropriate data from the database of Pandas, (2) aggregate them among all numerical time steps, and (3) export the data into CSV-based files. Here, scientists need to define even more sophisticated SQL, XPath, or XQuery statements than for the workflow depicted in Figure 1.1.

Such a complex data environment is common for simulations that are coupled across different scientific domains, since each domain has its own requirements and solutions for data handling. Scientists not only waste time for workflow design, but the increased number and complexity of workflow tasks often hinders them to design such workflows at all. As a consequence, they often cannot conduct simulations that couple different simulation models. However, this is usually needed to produce precise simulation results and to make simulations more realistic. Hence, adequate concepts and methods that alleviate the design of complex data provisioning tasks are essential for a wide adoption of simulation technology. This again confirms the major motivation considered by this thesis.

[4]GNU Octave: `http://www.gnu.org/software/octave/`

1.2 Challenges and State of Current Work

A systematic research method requires the identification and discussion of detailed challenges that arise from the major motivation of this thesis, i. e., from the high complexity of designing data provisioning tasks in simulation workflows. The identification of detailed challenges has been backed up by a comprehensive literature study, including the investigation of several academic use cases for simulations (e. g., see [Har96, GZC14]). Another basis has been a profound analysis of different workflow systems and of a set of real-world simulations. Beyond the running example illustrated above, this set of real-word simulations covers the examples described by Fehr et al. [FE11], Franzelin et al. [FDP15], Rommel et al. [RK11], and Wagner [Wag10]. The following sub-sections discuss the resulting challenges, as well as the corresponding state of current work.

1.2.1 Diversity of Available Data Provisioning Techniques

Scientific workflow systems usually offer several means to realize data access and data provisioning in workflows. For example, different services may encapsulate access to data sources and provide operations on this data, e. g., for data extraction [TDG07, GSK+11]. Some workflow systems additionally provide workflow extension activities to seamlessly access external data sources without an intermediate service layer [LAB+06, BJA+08, VSRM08]. However, the solutions offered by these systems are often proprietary. While services are typically tailor-made for specific applications, most of the workflow extension activities are customized for certain data sources or data management operations. To support as many applications as possible, most of the workflow systems even offer multiple of these proprietary services or customized workflow activities [CBL11].

So, scientists designing simulation workflows are faced with a large set of diverse data provisioning techniques. For a given data provisioning task in a simulation workflow, they have to choose the right technique from this set and have to decide how to properly integrate it into the workflow. The diversity and complexity of available data provisioning techniques may however overwhelm scientists [CBL11]. It may induce them not to leverage existing techniques at all, but still to implement the necessary data provisioning on their own. As a consequence, scientists need some kind of guidelines that help them to choose appropriate data provisioning techniques for given workflow tasks.

1.2.2 Multiplicity and Complexity of Low-Level Data Management Operations

Most of the data provisioning techniques offered by workflow systems still require scientists to specify a multiplicity of complex low-level data management operations in workflow tasks. For instance, this concerns sophisticated SQL, XPath, or XQuery statements realizing complex data filters or aggregations, as discussed in Section 1.1. This calls for an adequate data provisioning abstraction that removes the burden from scientists to specify many complex data management operations in their workflows. The above-mentioned analysis to identify detailed challenges for this thesis has revealed that such a data provisioning abstraction for simulation workflow design should provide scientists with the following benefits:

1. As illustrated in Section 1.1, one reason why scientists need to spend an increased effort on simulation workflow design is a high number of involved workflow tasks. So, an abstraction support should significantly reduce the number of tasks that are visible to scientists in their workflows [ZBML09].
2. An even more challenging issue is the high complexity of individual workflow tasks – especially when scientists need to employ programming languages or data modeling techniques they are not familiar with. Workflow design tools should offer a small set of abstract workflow building blocks that are meaningful to scientists [MBZL09]. For instance, these workflow building blocks should correspond to use cases scientists are interested in. In simulations, such a uses case may be coupling different simulation models [RSM14a].
3. Furthermore, workflow design tools should allow for a domain-specific parameterization of workflow building blocks [LAG03]. More precisely, they should enable scientists to work with terms or concepts they already know from their simulation methodology or simulation models [RSM14b].
4. The last issue arises from the desire to couple simulations of different scientific domains. To enable a seamless simulation coupling across arbitrary domains, an abstraction support has to be sufficiently generic [RSM14a]. This means it has to consider all individual requirements of the various domains, especially in terms of the used data sources or data management operations. Furthermore, the workflow building blocks offered by design tools should be widely re-usable in different domains [CBL11].

Existing scientific workflow systems mainly deal with research issues such as workflow scheduling in grid environments, provenance, or dataflow optimization [DSS+05, FSC+06, LAB+06, OGA+06, BJA+08]. Related work from the scientific workflow domain makes use of ontologies to allow for a domain-specific parameterization of workflow tasks, i. e., the parameters of these tasks correspond to abstract terms known from the ontologies [LAG03, BL05, MCD+05, WAH+07]. Hence, these approaches already provide the third benefit listed above. In addition, logical rules may define mappings of ontology terms to executable workflow tasks, e. g., realizing data format conversions [LAG03, BL05, RLSR+06, ZBML09]. In other words, the ontology terms and logical rules also cover the first above-listed benefit in that they hide a few low-level tasks and thus reduce the number of tasks that are visible to scientists. However, neither the scientific workflow systems nor related approaches completely offer the second and fourth benefits listed above. None of them provides scientists with abstract and meaningful workflow building blocks that are tailored to computer-based simulations. Instead, they mainly focus on service calls or customized workflow activities realizing data analysis pipelines. In addition, the sole use of ontologies entails that these approaches may be restricted to certain application domains, e. g., life sciences or geophysics, instead of providing a more generic solution.

Artifact-centric approaches to business process modeling leverage data as first-class citizens in processes [NC03, Hul08]. The data is represented by business artifacts that correspond to business-relevant and domain-specific objects, e. g., a customer order. These approaches bear resemblance to the ontology-based approaches from the scientific workflow domain. They thus have very similar gaps regarding the above-listed benefits of a data provisioning abstraction.

Common tools to define ETL processes (Extraction, Transformation, Load), e. g., as provided by Pentaho[5], offer a generic solution supporting various kinds of data sources and data management operations. However, they neither reduce the number of tasks in ETL processes or workflows, nor do they enable scientists to work with abstract, domain-specific, and meaningful workflow building blocks.

In conclusion, a consolidated data provisioning abstraction for simulation workflows that offers all of the discussed benefits has largely been neglected in previous work. It is however necessary to conquer the data complexity associated with simulation workflows and to make simulation workflow design tailor-made for scientists. This therefore constitutes the second challenge of this thesis.

[5]Pentaho: `http://www.pentaho.com/`

1.2.3 Efficient Data Processing and Optimization

In the bio-mechanical simulation workflow depicted in Figure 1.1, the size of result data generated by Pandas ranges between 10 MBs and multiple GBs. The actual data size varies according to the desired accuracy of the FEM-based calculation, i. e., according to the number of numerical time steps and number of spatial evaluation points. In the more complex scenario coupling bio-mechanical and systems-biological simulations, the bio-mechanical calculation is executed several times, each time with different boundary conditions [Kra14, RSM14a]. Different boundary conditions may require completely diverse accuracies of calculation. Hence, the size of result data may significantly vary across individual bio-mechanical simulation runs [RSM14a]. Furthermore, this multiplies the total data size generated by all bio-mechanical simulation runs to several Terabytes.

Such huge and dynamically changing data volumes are common for computer-based simulations, emphasizing the need for an efficient data processing in simulation workflows. Related work in the area of data-intensive applications discusses various optimization approaches. This includes approaches based on workflow re-structuring [VSS$^+$07, OdOV$^+$11, SWCD12], on MapReduce [DG08, ZBKL10, LTP11], or on cloud computing technologies [LQ14]. One challenge is to investigate whether and how such approaches may be transferred to data processing in simulation workflows. Furthermore, new optimization approaches should be developed, where existing approaches are not applicable or sufficient.

1.2.4 Data Quality

Mathematical simulation models approximate the system they imitate in that they usually describe only a portion of this system [Har96]. Numerical discretizations and their implementation in simulation tools often bring in additional uncertainties. The accuracy of simulation models and their discretizations often correlate to the quality of simulation result data [RTD$^+$12]. Scientists are frequently confronted with an additional optimization problem: a trade-off between low execution time of calculations (low accuracy) and high quality of their results (high accuracy). Nevertheless, not only the mathematical or numerical accuracy is decisive, but also the quality of input data of a calculation [RBD$^+$11]. This makes the optimization problem even more complicated in case the result data of one simulation is used as input of another one, i. e., in coupled simulations.

It is crucial to enable scientists to monitor and control data quality at each individual step of their simulation workflows. As an example solution, Reiter et al. propose a framework to specify data quality requirements directly in workflows, e. g., at control flow edges [RBD$^+$11, RBKK14]. In addition, they discuss various techniques to control and improve data quality, such as quality-driven workflow navigation or service selection [RBKK12, RBKK14].

1.2.5 Monitoring and Provenance Support

Scientists often make ad-hoc changes to workflows at runtime [SK10]. From a technical point of view, such ad-hoc changes are possible thanks to approaches enabling dynamic replacements or modifications of workflow parts [SK11, SK13, GSAH$^+$15]. However, scientists often still pose the questions when to modify which parts of their workflows in which way. This should be facilitated, e. g., by user-friendly methods to monitor workflow execution directly at runtime [SK10].

A related challenge is to ensure the reproducibility of a simulation as well as the traceability of its outcome [HTT09]. This is typically based on technologies and systems to capture, manage, monitor, and analyze provenance information [DF08, FKSS08, CVDK$^+$12]. Provenance information describes the detailed execution history of all phases and steps of scientific workflows. In particular, this covers the origin of input data and how these data are processed by workflows.

1.3 Contributions of this Thesis

As illustrated in the following sub-sections, the contributions offered by this thesis address the first three challenges discussed in Sections 1.2.1 to 1.2.3, i. e., the diversity of available data provisioning techniques, the multiplicity and complexity of low-level data management operations in workflows, and the need for an efficient data processing and for corresponding optimization approaches. So, this thesis does not explicitly discuss solutions to the issues related to data quality or monitoring and provenance support. This constraint is necessary to focus on the most severe issues regarding an abstraction support for the complex data provisioning in simulation workflows. Nevertheless, all five challenges, i. e., also the remaining two challenges, are used throughout the whole thesis to derive a comprehensive set of requirements for evaluating individual contributions.

1.3.1 Diversity of Available Data Provisioning Techniques

The first contribution addresses the challenge discussed in Section 1.2.1, i. e., scientists rarely leverage the various data provisioning techniques offered by workflow systems due to their high diversity and complexity. This thesis conquers this diversity and complexity by a systematic classification and comparison of available data provisioning techniques. Firstly, it identifies and defines three generic data provisioning concepts that are representative for the techniques offered by a multitude of workflow systems. In addition, the resulting data provisioning concepts are compared considering relevant functional and non-functional requirements for data provisioning in simulation workflows. The comparison results are finally used to derive a set of guidelines that assist scientists in choosing data provisioning techniques for their workflows. These guidelines also help scientists to focus on the most appropriate data provisioning concept, instead of being overstrained by a multitude of low-level techniques.

Moreover, the comparison results and the derived guidelines are used in this thesis to discuss essential features simulation workflow systems have to offer, as well as missing features of currently available workflow systems. This thesis additionally introduces an extended simulation workflow system that offers all these mandatory features in a holistic way – including those that are not offered by currently available systems. This extended workflow system corresponds to the SIMPL framework[6] [RRS+11, RS13b, RS14, RSM14a, RSM14b]. The next sub-sections describe more detailed contributions offered by SIMPL.

1.3.2 Multiplicity and Complexity of Low-Level Data Management Operations

Table 1.1 summarizes how major related work discussed in Section 1.2.2 covers the four identified benefits to be offered by a data provisioning abstraction for simulation workflow design. Furthermore, it compares related work with the SIMPL framework that is proposed by this thesis. The table again shows that none related approach offers all four benefits in a consolidated fashion. Furthermore, meaningful workflow building blocks that are tailored to computer-based simulations are not provided at all. Nevertheless, some of the approaches

[6] SIMPL stands for SimTech, Information Management, Processes, and Languages [RRS+11], but it is also an apronym for a *simple* data management in workflows.

Table 1.1: Comparison of the SIMPL Framework with major related work.

	Ontology-based scientific workflows	**Artifact-centric business processes**	**ETL tools**	**SIMPL** *ETL workflows*	**SIMPL** *Pattern-based workflows*
Reduced number of workflow tasks	moderate reduction	moderate reduction	no	no	significant reduction
Meaningful workflow building blocks	no	no	no	no	yes
Domain-specific parameterization	yes	yes	no	no	yes
Generic support of any domain	often specific	often specific	largely generic	largely generic	largely generic

at least offer a portion of the benefits. So, it may be advantageous to combine and augment the general ideas of different approaches in order to provide a holistic data provisioning abstraction for simulation workflow design. The SIMPL framework exactly fills this gap in current research. As shown in Table 1.1 and in the following sub-sections, SIMPL is subdivided into two complementary approaches: (1) an ETL workflow approach [RRS+11] and (2) a pattern-based approach to simulation workflow design [RS13b, RS14, RSM14a, RSM14b].

1.3.2.1 Generic ETL Workflow Approach

As discussed in Section 1.2.2, common ETL tools support various kinds of heterogeneous data resources and data management operations. Taking up the idea of Maier et al. [MMLW05], SIMPL combines this generic ETL technology with conventional workflow technology into an ETL workflow approach [RRS+11]. Such an approach enables the definition of operations for both simulation calculations and data provisioning at the same level of abstraction, i. e., at the workflow level. This particularly leads to a seamless design environment that removes the burden from scientists to get accustomed to many different tools or technologies.

Firstly, a data access service – as integral part of the SIMPL framework – offers generic operations for a unified access to arbitrary external data resources. Each of these operations covers one of the most common use cases for data access in simulation workflows, i. e., (1) data manipulation or data definition, (2) data retrieval into the workflow, (3) writing data back to external resources, and (4) data transfers between several external resources. Technical details, e. g., how to connect to a specific data resource, are covered by an extensible set of plug-ins, which implement the generic access operations for concrete data resources.

At the workflow design level, SIMPL extends the workflow language as well as workflow design and execution tools by a small set of data management activities. These activities correspond to the four generic operations of the data access service and offer the respective functionality directly within workflows. Workflow developers may assign an external data resource to such an activity and specify a command in the command language of this resource, e. g., a SQL statement or a shell command to access files. During activity execution, this command is issued over the data access service against the specified data resource.

This thesis discusses the following contributions of the SIMPL framework and especially of the resulting ETL workflow approach to a generic solution for data provisioning in simulation workflows:

- The generic data access service and its plug-in mechanism facilitate a seamless extension by additional and arbitrary kinds of data resources.
- The four access operations offered by this service, as well as the corresponding use cases for data access are common for most data provisioning scenarios in simulation workflows. They are thus sufficient to support a majority of simulation examples in any scientific domain.
- The data management activities allow for specifying a wide range of operations for data provisioning. In fact, they support any operation that may be specified via the command languages offered by involved external data resources e. g., SQL statements or shell commands.

1.3.2.2 Pattern-based Approach to Simulation Workflow Design

The ETL workflow approach illustrated above helps scientists to define workflows without being forced to provide any technical details of data access mechanisms, as the data access service already covers these details. However, scientists still have to specify many sophisticated data management operations, e. g., in terms of complex SQL or XQuery statements. As shown in Table 1.1, ETL workflows do neither reduce the number of visible workflow tasks nor do they provide scientists

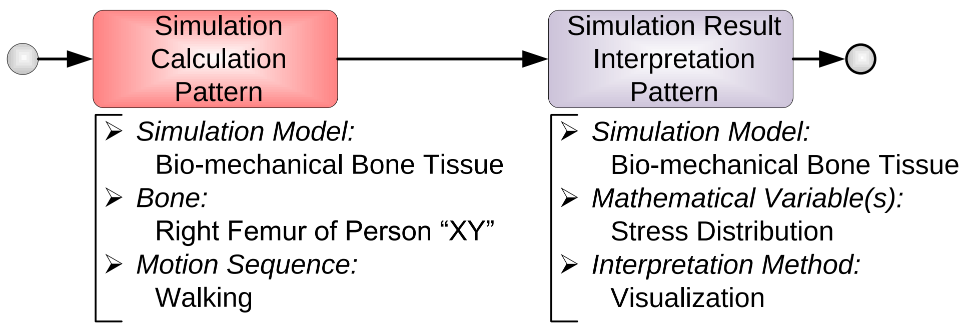

Simulation Calculation Pattern
Simulation Result Interpretation Pattern
Simulation Model: Bio-mechanical Bone Tissue
Bone: Right Femur of Person "XY"
Motion Sequence: Walking
Simulation Model: Bio-mechanical Bone Tissue
Mathematical Variable(s): Stress Distribution
Interpretation Method: Visualization

Figure 1.2: Pattern-based design of the simulation workflow shown in Figure 1.1.

with abstract and meaningful workflow building blocks that allow for a domain-specific parameterization. SIMPL fills this gap by a pattern-based approach to simulation workflow design offering a full-fledged and principled abstraction support for the complex data provisioning in simulation workflows [RS13b, RS14, RSM14a, RSM14b]. Scientists using this approach select a small set of abstract patterns and combine them in their simulation workflows to describe only the main steps of these workflows. The patterns thereby completely remove the burden from scientists to specify any low-level details of data provisioning.

To illustrate this idea, Figure 1.2 shows how such patterns may alleviate the design of the bio-mechanical simulation workflow depicted in Figure 1.1. This pattern-based workflow design goes beyond related work and provides scientists with all of the four benefits listed in Section 1.2.2 as follows (see also Table 1.1):

1. Each pattern combines several low-level workflow tasks, which reduces the number of tasks that are visible to scientists. To design the bio-mechanical simulation workflow, scientists only need to specify two patterns as shown in Figure 1.2, instead of the seven original workflow tasks (see Figure 1.1). This constitutes a significantly higher reduction of the number of tasks than it is provided by major related work, e. g., in the areas of ontology-based scientific workflows [LAG03] and artifact-centric business processes [Hul08].
2. As a unique selling point that is not provided by related work at all, the two patterns shown in Figure 1.2 are particularly meaningful to scientists conduction simulations. They represent two of the main use cases these scientists are interested in: (1) the execution of simulation calculations based on a mathematical model and (2) the interpretation of their results.
3. The adaptation to a concrete simulation scenario is achieved by a small set of pattern parameters. Here, the core ideas of ontology-based scientific

workflows [LAG03] and artifact-centric business processes [Hul08] are transferred to simulation workflows. So, all pattern parameters correspond to concepts or artifacts scientists already know from their simulation models or domain-specific methodology. The first pattern is parameterized by a *simulation model* to be calculated for a concrete *bone* and for a *motion sequence* determining how this bone moves over time. The parameters of the second pattern indicate which *mathematical variables* of which *simulation model* shall be interpreted via which *method*.

4. The two patterns shown in Figure 1.2 also represent uses cases that are common for the simulation domain. This thesis discusses how these patterns and all other developed patterns may be re-used to alleviate the design of various simulation workflows of different domains [Wag10, FE11, RK11, Kra14, FDP15]. Finally, all patterns may be implemented by the ETL workflow approach discussed in Section 1.3.2.1 and offering a generic solution in terms of supported data resources and data management operations.

To make such abstractly modeled patterns executable, SIMPL comprises an extensible set of rewrite rules that specify the transformation of patterns into executable workflow fragments [SKK+11, RSM14a]. An additional pattern hierarchy organizes various patterns at different abstraction levels. This allows for a multi-step and thus more generic rule-based transformation of patterns. For instance, the simulation-specific patterns shown in Figure 1.2 may firstly be mapped onto more fine-grained data-oriented patterns. These data-oriented patterns define a data provisioning process mainly in terms of low-level ETL operations [RSM14a]. In a similar way, rewrite rules map such data-oriented patterns onto executable workflow fragments that finally implement the data provisioning.

As another major contribution, the pattern hierarchy and its clear distinction of patterns according to their degree of abstraction facilitate a separation of concerns between different persons that may now be involved in workflow design. 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 templates of patterns and workflow fragments at the chosen level. For instance, the patterns shown in Figure 1.2 are good candidates to be selected and parameterized by scientists. Data engineers may in turn use their expertise to provide these scientists with templates of the fine-grained data-oriented patterns. Altogether, this conquers the data complexity associated with simulation workflows, and it significantly reduces the time scientists have to spend on workflow design.

1.3.3 Efficient Data Processing and Optimization

The main focus of this thesis is on a principled abstraction support conquering the complexity of data provisioning in simulation workflows. Therefore, its intention is not to offer full-fledged optimization approaches that improve the efficiency of data processing in simulation workflows. Nevertheless, it discusses whether and how the individual contributions of SIMPL described above may facilitate the application of existing optimization approaches to simulation workflows (see Sections 5.5.4 and 6.6.4). This mainly includes optimization approaches mentioned in Section 1.2.3, i. e., approaches based on workflow re-structuring [VSS+07, OdOV+11, SWCD12], on MapReduce [DG08, ZBKL10, LTP11], or on cloud computing technologies [LQ14].

The last contribution is a new method to improve both the performance and robustness of the local data processing within simulation workflows or other data-intensive workflows [RSM11]. It is targeted at workflows that are described by control-flow-oriented workflow languages [GSK+11], because corresponding optimization approaches have largely been neglected in previous work [Mül10, Wag11]. So, it fills a gap in current research, where existing approaches are not applicable or sufficient. The new method introduces various techniques that partition the local data processing tasks to be performed during workflow execution in a smart way. This mainly encompasses the execution of variable assignments, expression evaluations for control flow decisions, and service calls. Thereby, such tasks are either assigned to the workflow execution engine or to a tightly integrated local database engine. The effectiveness of these techniques is evaluated by applying them to various test scenarios, including a real-world scientific application [Wag10].

1.4 Outline of this Thesis

Most parts of this thesis correspond to revised and composite versions of previous author publications [RRS+11, RSM11, RS13b, RS14, RSM14a, RSM14b, RWWS14]. These publications are also respectively cited at relevant occasions in the above Section 1.3. This Section 1.4 summarizes the outline of the rest of this thesis.

The next *Chapter 2* provides background information about related terms, concepts, and technologies that are relevant for this thesis. This mainly encompasses information about three separate topics: (1) computer-based simulations and mathematical simulation modeling, (2) data resources that are used most commonly in simulations, as well as (3) workflow technology, workflow languages, and workflow systems.

Chapter 3 then combines these three topics in order to circumscribe the major application area considered in this thesis, i. e., data management and data provisioning in simulation workflows. It introduces additional real-world simulation examples that constitute the main uses cases of this thesis. Furthermore, it provides more in-depth discussions of common data characteristics and data management patterns of computer-based simulations.

The subsequent Chapters 4 to 7 detail the contributions of this thesis, which are summarized in Section 1.3. Thereby, these chapters not only illustrate the respective approaches and their design considerations. In addition, each chapter covers comprehensive discussions of related work, and especially profound evaluations of the proposed concepts and methods. These evaluations have been backed up by elaborate prototypical implementations and by their application to several real-world simulations, as mentioned above.

Chapter 4 provides more insights regarding the contribution introduced in Section 1.3.1. So, it deals with the challenge that most existing workflow systems overwhelm scientists with a multiplicity of diverse data provisioning techniques. It discusses how to classify and compare existing techniques in order to conquer this multiplicity and diversity. One outcome of this discussion is a set of guidelines that assist scientists in choosing data provisioning techniques for their workflows. Furthermore, Chapter 4 derives essential features that are missing in currently available workflow systems. It therefore introduces an extended workflow system that offers all missing features in a comprehensive way.

One missing feature of currently available workflow systems is that they do not support all kinds of data resources or data management operations required by computer-based simulations. *Chapter 5* therefore introduces and assesses a set of extensions to workflow languages that address this lack of generality [RRS+11]. As discussed in Section 1.3.2.1, these extensions – so-called data management activities – combine conventional workflow technology with ETL technology into an ETL workflow approach [MMLW05]. This ETL workflow approach allows for specifying a broad range of data management operations for any data resource directly within workflows.

The data management activities proposed in Chapter 5 still do not remove the burden from scientists to specify many complex data management operations. Hence and as illustrated in Section 1.3.2.2, *Chapter 6* proposes a novel pattern-based approach that offers a consolidated abstraction support for this complex data management [RS13b, RS14, RSM14a, RSM14b]. This approach especially provides scientists with all essential benefits listed in Section 1.2.2. Another major feature of this new approach is the above-mentioned pattern hierarchy arranging different kinds of patterns according to their degree of abstraction. This facilitates a holistic separation of concerns to incorporate different kinds of persons and their various skills into workflow design.

Chapter 7 then discusses the complementary optimization method introduced in Section 1.3.3. This novel method aims at improving the performance and robustness of local data processing tasks within simulation workflows or other data-intensive workflows. Thereby, it is especially targeted at workflows that are described by control-flow-oriented languages [GSK+11], which has largely been neglected in previous work [Mül10, Wag11].

Finally, *Chapter 8* concludes this thesis with a summary of its major contributions. Furthermore, it lists promising opportunities for future research.

Chapter 2

Background

This chapter provides information about terms, concepts, and technologies that are important to understand the content of this thesis. In Section 2.1, relevant background information about computer-based simulations, mathematical simulation models, and typical phases of simulation workflows are given. Section 2.2 circumscribes important aspects of data resources that are used for computer-based simulations, as well as the capabilities these resources offer in terms of their command or query languages. Finally, Section 2.3 presents an overview of workflow technology, workflow languages, and workflow systems. A few parts of this chapter are revised versions of excerpts of previous author publications that are cited at affected locations [RRS$^+$11, RSM11, RS13b].

2.1 Computer-based Simulations

Hartmann defines the terms simulation and computer(-based) simulation as follows [Har96]:

> "Simulations are closely related to dynamic models. More concretely, a simulation results when the equations of the underlying dynamic model are solved. This model is designed to imitate the time-evolution of a real system. To put it another way, *a simulation imitates one process by another process.* In this definition, the term 'process' refers solely to some object or system whose state changes in time. If the simulation is run on a computer, it is called a *computer simulation.*"

This thesis mainly focuses on such *dynamic simulations* that imitate the behavior or evolution of usually real-world experiments, processes, or other systems over a certain period of time. In contrast, *static simulations* investigate the imitated systems at rest, instead of considering their time-dependent evolution [Har96]. However, dynamic simulations are much more realistic, because the underlying systems – especially those investigated in natural sciences or engineering – are inherently dynamic [Har96].

Computer simulations (also referred to as *computer-based simulations* in this thesis) are frequently used to replace physically realized experiments or other investigations. The main motivation is that these physical experiments are often too costly or too time-consuming [Har96]. This for instance concerns car development processes, where computer-based simulations may serve as virtual and economic alternatives to physically realized crash tests [Hau81].[1] Other phenomenons cannot be investigated in sufficient detail at all without relying on simulations, e. g., due to technical, theoretical, pragmatical, or ethical issues [Har96]. For instance, the time frame of the formation and evolution of galaxies is much too big so that physical experiments investigating this kind of phenomenon are not feasible at all.

According to the definition of Hartmann cited above, the imitated system and its evolution over time are typically described by a mathematical simulation model. Most commonly, such models are composed of a system of ordinary or partial differential equations (ODEs or PDEs, respectively) [BDH11]. ODEs or PDEs very often do not have a known analytic solution [Hum90], which is mainly due to the fact that they describe the time-dependent evolution of the imitated system in a continuous way. So, they usually cannot be directly solved by computers. Instead, they need to be discretized and approximated using numerical methods, such as the Finite Element Method (FEM) [GRS07, ZTZ13]. The following Section 2.1.1 exemplifies these two aspects of mathematical simulation models and their numerical implementation via the running example of a bone simulation illustrated in Section 1.1. Nevertheless, the focus is not on mathematical or numerical details, but on aspects that are relevant for the data provisioning in simulations. Afterwards, Section 2.1.2 depicts the typical and most important phases of simulation workflows.

[1]Note that a physical crash test is also a simulation, because it imitates the behavior of a car and its passengers during a real-world car accident on streets. However, it is not a computer-based simulation, because it is based on another physical experiment, and not on calculations in a computer.

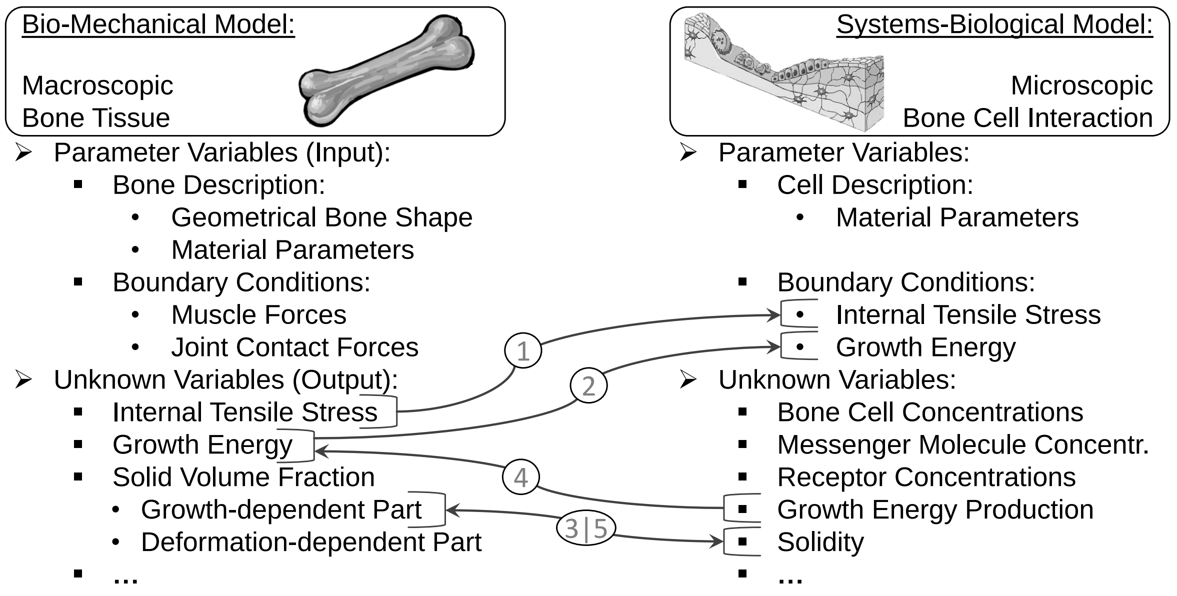


Figure 2.1: Coupled bio-mechanical and systems-biological simulation models to investigate structure changes in bones, as well as an extract of mathematical variables that are important for data provisioning and data exchange.

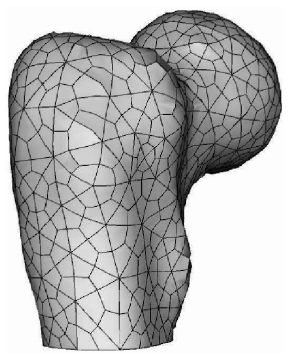
2.1.1 Simulation Models and their Implementation

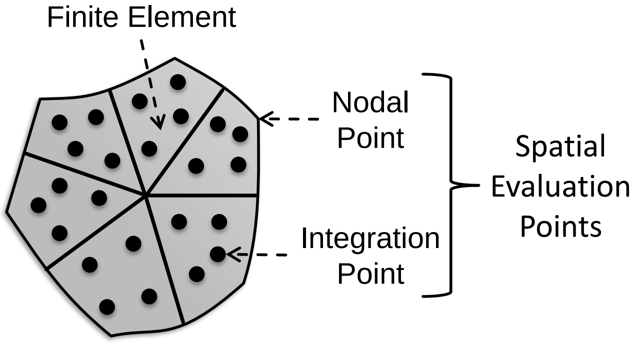
Figure 2.1 shows how the bio-mechanical and the systems-biological simulation models are coupled in order to realize the running example of a bone simulation illustrated in Section 1.1 [Kra14]. Furthermore, the figure respectively classifies important mathematical variables of these simulation models. These variables either serve as input or are produced as output of the underlying systems of differential equations. They are thus important for the data provisioning for a single simulation model, as well as for the data exchange between both models. *Parameter variables* serve as input of a model, whereas *unknown variables* represent its output and are typically calculated for several time steps. In some examples, a set of initial conditions, i. e., values of specific unknown variables for the first time step of the simulation, has to be provided as further input.

The bio-mechanical simulation model is based on the Theory of Porous Media and governed by PDEs [Ehl09]. Its main focus is on the macroscopic mechanical behavior of a bone, especially on the exchange of mass and momentum between porous solids of the bone tissue and therein embedded fluids [Kra14]. The bio-mechanical parameter variables include a static *description* of the *bone* to be simulated, i. e., the geometrical bone shape and material parameters. In addition, the *boundary conditions* determine the time-dependent external load on the bone, e. g., caused by muscle forces and by joint contact forces of adjacent bones. The simulation converts such an external load situation into the *internal tensile stress* within the bone tissue. Another resulting unknown variable is the *growth energy* that summarizes the chemical energy available for cell metabolism, e. g., offered by glucose. The *solid volume fraction* represents the bone density and is subdivided into multiplicative growth-dependent and deformation-dependent parts. Note that it is not necessary to provide the bio-mechanical simulation with explicit initial conditions. The reason is that this simulation may start with arbitrary initial conditions and then calculates its unknown variables in an iterative way until they converge to a steady state [Kra14].

The bio-mechanical model does not consider any cellular reactions within the bone tissue. However, both the growth energy and the growth-dependent part of the solid volume fraction depend on cellular reactions as well, which may result in an inaccurate calculation of these unknown variables [Kra14]. This is where the systems-biological model comes into play, which uses ODEs to describe the microscopic formation or resorption of the bone tissue as result of the stress-regulated interaction between cells. Its first parameter variable is a *cell description*, i. e., some cell-specific material parameters. The bio-mechanical simulation provides it with its boundary conditions, namely the internal tensile stress (step 1 in Figure 2.1) and the growth energy (step 2). In addition, an interpolated value of the growth-dependent solid volume fraction corresponds to the initial condition of the systems-biological *solidity* (step 3). Besides the solidity, the major systems-biological unknown variables encompass *concentrations* of *bone cells*, *messenger molecules*, and *receptors*, as well as the cellular *growth energy production*. To close the loop, the systems-biological and more precise growth energy production and solidity are finally used to update the growth energy and the growth-dependent solid volume fraction of the bio-mechanical model (steps 4 and 5). More details about this process are given in Section 3.2.2.

One of the most common solutions to numerically discretize a system of PDEs is the FEM [GRS07, ZTZ13]. This method is also applied to the bio-mechanical simulation model [Kra14]. Thereby, the whole and complex problem domain, i. e., the bone in this example, is spatially subdivided into several simpler parts: the finite elements. Figure 2.2a shows an FEM grid consisting of individual finite elements as spatial discretization of a human femur. This FEM grid and its finite elements are described by the parameter variable representing the geometrical


(a) FEM grid of a femur.



(b) Part of the grid and spatial evaluation points.

Figure 2.2: FEM-based spatial discretization of a human femur.

bone shape (see Figure 2.1). For each finite element, specific base functions as well as the values of material parameters, initial conditions, and boundary conditions are inserted into the PDEs to convert them into a system of discrete linear or non-linear equations. This system of equations allows for calculating discrete and locally approximate solutions of the unknown variables at individual spatial evaluation points of the FEM grid. As indicated in Figure 2.2b, one kind of spatial evaluation points are *nodal points*, i. e., the points where different finite elements coincide. According to the collocation method [dBS73, AP97], each nodal point is associated with a collocated *integration point* within a finite element. Each bio-mechanical unknown variable is thereby calculated at nodal points, at integration points, or even at both [Kra14]. Finally, the systems of equations of all finite elements are composed to a global system of equations that allows for calculating the unknown variables for the whole FEM grid. The time discretization of the bio-mechanical simulation subdivides the whole continuous time interval into several equidistant numerical time steps [Alb96, Kra14]. The above-mentioned global system of equations is then consecutively solved for each of these numerical time steps. In summary, the bio-mechanical simulation calculates one discrete value (1) for each degree of freedom in the mathematical simulation model (the unknown variables), (2) for each relevant spatial evaluation point of the FEM grid, and (3) for each numerical time step.

The ODEs describing the systems-biological simulation model get their initial and boundary conditions – including some unknown variables of the bio-mechanical model – at the integration points within the finite elements. They are then likewise converted into a system of discrete equations, but using a less complex numerical discretization [Kra14]. While the bio-mechanical calculations at different spatial evaluation points heavily depend on each other, the systems-biological calculations at individual points are completely independent. So, the systems-biological unknown variables may be calculated locally at each integration point [Kra14]. It is thereby sufficient to directly solve the smaller local systems of discrete equations at these points instead of composing them to a more complex global system. Note that this spatial independence of calculations is also the reason why the systems-biological model does not require a geometry description as part of its parameter variables (see Figure 2.1). The individual local systems of equations are again consecutively solved for a number of numerical time steps. The states of cells within the bone tissue change rather infrequently compared to the mechanical properties described by the bio-mechanical model. So, the systems-biological simulation increases its time scale, i. e., numerical time step sizes change from about ten milliseconds in the bio-mechanical calculation to several seconds or even minutes at the systems-biological scale.

In general, simulations that are coupled among different scientific domains not only require different scales in temporal, but also in spatial discretizations [GZC14]. For instance, the bio-mechanical simulation model may be coupled with a more coarse-grained approach to even better understand the holistic behavior of greater parts of the human skeleton. This coarse-grained approach is a multi-body simulation describing the mechanical behavior of all relevant bones and skeletal muscles, as well as of the interfaces between these bones and muscles [RKH$^+$10]. Its simulation model is coupled with the bio-mechanical model in a similar way as the bio-mechanical and systems-biological models depicted in Figure 2.1. More precisely, the unknown variables calculated by the multi-body simulation cover the muscle forces and joint contact forces that are used as boundary conditions of the bio-mechanical model. In analogy, the systems-biological simulation may be coupled with fine-grained and thus more accurate molecular-dynamical or even quantum-mechanical simulations [RK11]. All these differences in spatial scales of simulation calculations may result in likewise varying temporal scales, i. e., numerical time step sizes may vary between hours and nanoseconds. Simulations that correspondingly differ in spatial and/or temporal scales of their calculations are called *multi-scale simulations* [GZC14]. Due to the differences in spatial and temporal discretizations, the data exchanged between several simulation models of multi-scale simulations must be interpolated or extrapolated accordingly [Gat14]. More details about the resulting complexity of data exchange are given in Section 3.2.2.

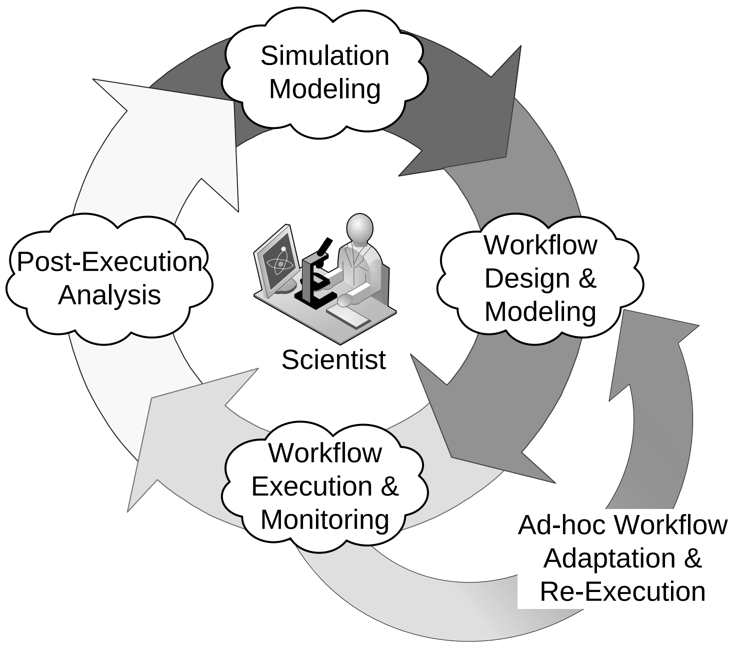

Simulation Modeling
Workflow Design & Modeling
Post-Execution Analysis
Scientist
Workflow Execution & Monitoring
Ad-hoc Workflow Adaptation & Re-Execution

Figure 2.3: Lifecycle of computer-based simulations and simulation workflows.

2.1.2 Typical Phases of Simulation Workflows

As indicated in Figure 2.3, scientists are involved in all phases of the typical lifecycle of computer-based simulations and simulation workflows [LWMB09, SK10]. In the first phase *simulation modeling*, they define mathematical simulation models describing the usually time-dependent behavior of the systems to be investigated, e. g., in terms of ODEs or PDEs as illustrated in Section 2.1.1. The next phase *workflow design and modeling* comprises tasks to construct workflow models that realize the simulation models and that produce the simulation outcome. During *workflow execution and monitoring*, scientists observe and control the workflow execution. They should furthermore be able to make ad-hoc changes to workflow models at their runtime and to re-execute affected parts of them [SK10]. Finally, they analyze and interpret the simulation outcome in the phase *post-execution analysis*, e. g., based on a visualization of result data. This may lead to the decision to adapt the used simulation models or to couple these simulation models with other ones and finally to repeat the whole lifecycle.

Accessing and providing huge amounts of heterogeneous input data as well as generating huge intermediate and final data sets are some of the major challenges of scientific workflows and likewise of simulation workflows [RLSR+06, GDE+07, DC08, CBL11]. This is also highlighted by the common structure of a simulation

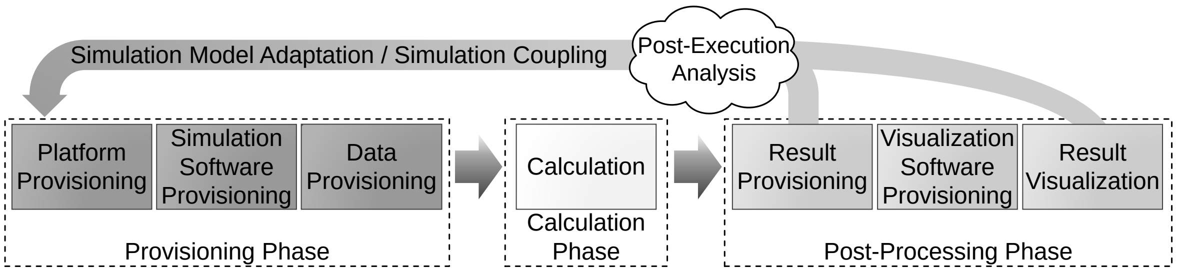


Figure 2.4: Common structure and phases of a simulation workflow.

workflow as shown in Figure 2.4. The *provisioning* phase covers pre-processing activities that are necessary before the actual simulation calculations are executed. Possible tasks in the sub-phase *platform provisioning* are to deploy necessary middleware components, e. g., application servers [VHKL13]. Other tasks include the creation of data containers e. g., some directory structures in file systems that may be used to store input and output data of simulation calculations. In the sub-phase *simulation software provisioning*, the workflow deploys and configures all software packages needed in the calculation phase [GSK$^+$11, VHKL13]. The activities of the third sub-phase *data provisioning* provide the simulation software with necessary input data. This is the phase where the workflow has to carry out many complex tasks that filter input data from various heterogeneous data sources and transform these data according to the formats and granularity the simulation software requires [RLSR$^+$06, RRS$^+$11, RS14].

In the *calculation* phase, the workflow uses the previously provisioned platform, software, and input data to perform the actual simulation, i. e., to compute and store the simulation outcome. The activities in the *post-processing* phase prepare the post-execution analysis (see Figure 2.3). This is commonly based on a visualization of the simulation outcome in the last sub-phase *result visualization*. The sub-phases *result provisioning* and *visualization software provisioning* prepare the data and software needed for this visualization in a similar way as the corresponding sub-phases in the provisioning phase. In case scientists decide to adapt the simulation models, the post-execution analysis may entail the re-execution of the same workflow or of an altered version of it. As another option, the simulation outcome may also be used as input of another simulation workflow to couple the current simulation model with a different one. Especially in multi-scale simulations, this simulation coupling entails the execution of additional and even more complex workflows that perform the data exchange between the underlying simulation models and simulation workflows [RSM14b].

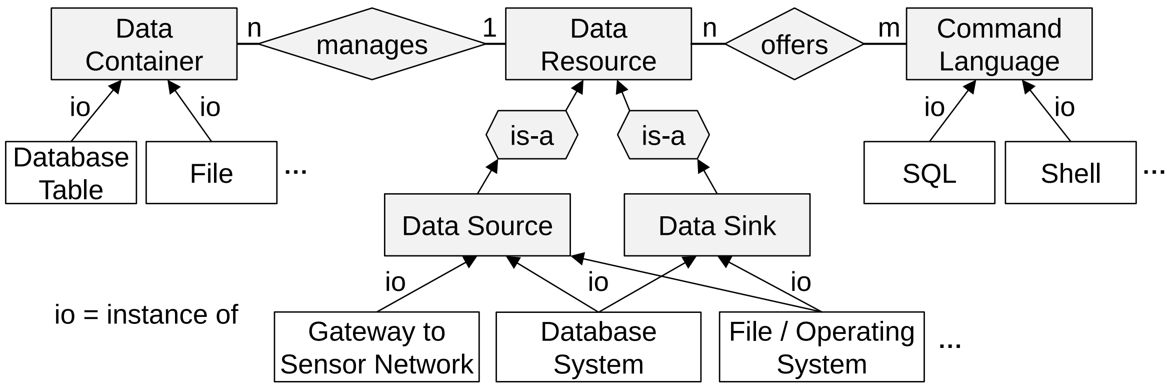


Figure 2.5: Relevant terms and concepts regarding data resources; cf. [RS13b].

2.2 Data Resources and Command Languages

Figure 2.5 defines relevant terms and concepts regarding *data resources* that are frequently used to manage the various input and output data of computer-based simulations [RS13b]. A data resource may be a *data source* that is able to provide clients with data and/or a *data sink* in which clients may store data. For instance, gateways to sensor networks generally do not allow to store data and thus may often only act as data sources [DP10]. Database systems and file or operating systems usually may act both as data sources and as data sinks. A data resource manages several *data containers*. Each data container is an identifiable collection of data, e. g., a table in a database system or a file in a file or operating system. Furthermore, a data resource offers specific *command* or query *languages*, e. g., SQL or shell languages. It receives and executes data management commands that are described in the offered languages and that may be used to create, read, update, and delete data containers and their data.

The following sections illustrate concrete examples of data containers and command languages for different kinds of data resources that are most relevant for computer-based simulations. This includes a discussion how flexible the respective data resources are regarding data organization and data structures. Other aspects are the capabilities or the power of command or query languages in terms of the data management operations they support. Thereby, the main focus is on giving an overview of the native capabilities of single commands described in the offered languages, but possibilities to extend the capabilities are discussed as well. Table 2.1 summarizes the results of these discussions.

Table 2.1: Data resources used in computer-based simulations.

Kind of data resource	**Typical data containers**	**Flexibility w.r.t. data structures**	**Typical command language**	**Power of command language**
File / operating systems	file or directory	very high	shell	moderate
Relational database systems	database table or schema	moderate	SQL	high
XML database systems	XML document or collection	high	XQuery / XPath	high
Sensor networks and gateways	proprietary	usually low	proprietary	usually moderate

2.2.1 File / Operating Systems

The data containers managed by file or operating systems are individual *files* or *directories*, which may contain several further files or sub-directories. Using files and directories is the most common way to manage data in computer-based simulations. The main reason is a very *high flexibility* offered by file or operating systems with respect to data organization and data structures. For instance, files and directories may be flexibly organized and grouped in hierarchical directory structures. In fact, scientists often adopt application-specific conventions for file names, directory names, and directory structures in order to ensure a well-documented and reproducible data organization [MBBL15]. Another benefit is that files support virtually any data format and data structure. Unstructured ASCII text or binary formats allow for arbitrary file contents or for user-defined and customized data structures within files. Other options are structured or semi-structured formats that allow for the definition of specific data schemata as long as they comply with a certain base structure. Examples are CSV-based files or XML documents that allow for tabular or hierarchical representations of file contents, respectively. It is in general even possible to combine different structured or unstructured formats within one single file. The flexibility with respect to data formats even makes files the default option for exporting data from or importing them into other data resources. As an example, relational database systems usually allow for exporting or importing database tables to or from CSV-based files, or even whole databases to or from text or binary files.

The high flexibility of file or operating systems regarding data structures however leads to only a *moderate power* of the offered command language. Corresponding *shell languages* are rather intended for automating tasks of system administrators than for executing complex data management operations. The capabilities operating systems support via native shell commands include basic operations for data definition, e. g., to create, delete, move, or copy files or directories. Native shell commands may often also be used to initially set the content of files while they are created, or to append additional contents to existing files. Furthermore, most operating systems offer system utilities to search for files or to filter or manipulate parts of the content of a single file. For instance, the grep utility (Globally search a Regular Expression and Print)[2] allows for specifying regular expressions that define a corresponding search or manipulation pattern. However, such regular expressions need to be customized to the respectively used data formats, e. g., ASCII text files require completely different regular expressions than XML documents. This is also one reason why the toolbox offered by operating systems is increasingly extended by system utilities that natively support specific operations for certain kinds of data formats. For instance, the Windows PowerShell supports some basic operations to read and manipulate the content of XML documents, including XPath-based filters [Sch14].

In order to support more complex operations to be executed on files, the utilities of operating systems have to be extended by external libraries or tools. Common external libraries or tools however even lack support of highly complex operations that work on more than one file at once. This for instance concerns set-oriented operations, such as intersections, unions, or complements. Such operations usually have to be implemented by writing additional shell scripts. However, this is often associated with a remarkable implementation overhead.

2.2.2 Relational Database Systems

In single simulation applications, relational database systems are used as alternatives or complements to a solely file-based data management. For instance, Heber et al. propose and evaluate a relational database back-end to support Finite Element Analyses (FEA) [HG05, HG06, HPD$^+$05]. An FEA encompasses all phases of a FEM-based simulation process: Developing a simulation model, generating an FEM grid, building and solving a system of discrete equations,

[2]GNU Grep: `http://www.gnu.org/software/grep/`

```sql
SELECT base_function_equation
FROM FEM_parameters
WHERE model = 'boneTissue' AND dim = 3 AND nodes_per_element = 20
```

Listing 2.1: Exemplary SQL query to retrieve a base function equation from the database tables storing FEM parameters depicted in Figure 1.1.

pre-processing result data, and interpreting the final outcome [CMPW07]. In engineering, several of such FEA-based processes are conducted consecutively or in parallel in order to predict the behavior of different variants of a product under specific circumstances. This results in highly data-intensive simulation applications. Here, relational database systems may be attractive for their efficient and robust capabilities to process big and complex amounts of data [HG05].

The data managed by relational database systems is organized according to the relational model, i. e., the data consists of a set of relations [Cod70]. A relation is an instance of a relation schema $R(A_1, ..., A_n)$ with its attributes A_1 to A_n. So, a relation $r(R)$ is a subset of the cross product of the attribute domains of the relation schema R, i. e., $r(R) \subseteq dom(R) = dom(A_1) \times ... \times dom(A_n)$. In other words, a relation is a set of tuples, where each tuple is a list of n values $(v_1, ..., v_n)$ with $v_i \in dom(A_i)$. In a similar way as CSV-based files, this relational model imposes a tabular base structure for the major data containers managed by relational database systems, i. e., for *database tables*. Each database table may be structured according to an arbitrary list of n columns (the attributes A_i) with virtually arbitrary column types (the attribute domains $dom(A_i)$). The rows of a database table then correspond to the tuples of the relation. This restriction to the relational model and thus to a tabular base structure entails only a *moderate flexibility* with respect to data structures when compared to the high flexibility offered by file or operating systems. Other kinds of data containers are *database schemata* that summarize one or more database tables and specify their tabular structure, i. e., the underlying relation schema.

On the other hand, the restriction to the relational model leads to a *high expressive power* of the command or query languages offered by relational database systems. The de-facto standard for such command or query languages is the Structured Query Language (SQL) [ISO11b, ISO11d]. Listing 2.1 shows an exemplary SQL query that is used in the workflow depicted in Figure 1.1 to access the database table storing FEM parameters (`FROM` in Listing 2.1). The query delivers a text-based encoding of a base function equation (`SELECT`) that may later be used to convert the PDEs of the simulation model into a system of discrete equations (see Section 2.1.1). Thereby, it filters all base functions summarized in the database table according to (1) the simulation model, (2) the number of spatial dimensions, and (3) the number of nodal points per finite element in the FEM grid (`WHERE` in Listing 2.1). SQL is based on the relational algebra, which has the same expressive power as first-order predicate calculus [Cod70]. It supports all set-oriented operations, e. g., union, intersection, set difference, or the Cartesian product of several database tables. The major contribution to the power of SQL is offered by table operations, such as projection, selection, division, and different variants of relational joins [Cod70]. Further features of SQL encompass nested sub-queries, recursive queries, as well as many built-in functions to aggregate, group, and sort the output of a query. Besides this powerful query capability, SQL offers many possibilities to define tabular data structures (data definition), as well as to insert, update, and delete data (data manipulation). Moreover, it includes mechanisms to specify assertions or triggers that ensure the semantic integrity of data. Thereby, the relational model itself offers built-in integrity constraints, e. g., to ensure the uniqueness of tuples via primary or candidate keys and the referential integrity between several relations using foreign keys [Cod70]. In addition, SQL allows for defining custom integrity constraints, e. g., value-based constraints between several table columns.

Over and above, SQL encompasses many additional parts that extend the pure standard and even increase its power. For instance, Persistent Stored Modules (SQL/PSM) add support for procedural extensions [ISO11e]. This includes the declaration and implementation of so-called stored procedures and user-defined functions, as well as control flow constructs, variable assignments, and mechanisms to handle exceptions. This procedural SQL extension offers comparable features as scripting technologies provided by shell languages of operating systems. Another example is an XML extension (SQL/XML) that allows for storing, querying, and manipulating native XML data within relational database tables and in conjunction with pure relational data [ISO11c]. Note that this extension not only increases the expressive power of SQL, but also the flexibility with respect to data structures offered by database systems (see Section 2.2.3). It is even possible to manage spatial data in some relational database systems, i. e., to consider spatial issues such as geometry and location [ISO11a].

```
<materialParameters>
  <person id="007">
    <bone name="leftFemur">
      <stiffness>3.0</stiffness>
      <solidDensity>2.1e−6</solidDensity>
      <hydraulicConductivity>3.0e−2</hydraulicConductivity>
      ...
    </bone>
    <bone name="rightFemur"> ... </bone>
    ...
  </person>
  <person id="008"> ... </person>
  ...
</materialParameters>
```

Listing 2.2: Exemplary excerpt of an XML document storing material parameters accessed by the workflow shown in Figure 1.1. Parameter values are written in scientific E notation and assume standard physical units.

2.2.3 XML Database Systems

XML database systems, which are able to natively process XML data, are used in some, but rather few simulation applications. One use case is the material parameters accessed by the workflow shown in Figure 1.1. Material parameters usually differ between individual bones of individual persons to be simulated. So, a large XML document or even multiple documents may be needed to describe multiple sets of such material parameters. Depending on the number of these sets, a database system offering an efficient and robust data management may be a better choice than relying only on XML documents stored in a file system.

The major data containers managed by XML database systems are (1) individual *XML documents*, as well as (2) *document collections*, which comprise several XML documents. Listing 2.2 exemplifies an XML document storing the material parameters accessed by the simulation workflow shown in Figure 1.1. This document represents the material parameters of several bones and of several persons. XML imposes a hierarchical base structure to describe data via nested tag-based elements `<elementName>elementData</elementName>` [W3C15]. The hierarchical nesting of such tag-based elements also corresponds to a tree-based data structure. In addition, each element may contain a list of attributes written

```
FOR $bone IN fn:doc("materialParameters.xml")//person[@id="007"]/bone
LET $bone_name = fn:data($bone/@name)
WHERE $bone_name = "leftFemur"
ORDER BY $bone/*/name() ASCENDING
RETURN <params>{$bone/*}</params>
```

Listing 2.3: Exemplary XQuery FLWOR statement to retrieve an ordered sequence of all material parameters of person '007' and bone 'leftFemur' from the XML document 'materialParameters.xml' depicted in Listing 2.2.

as `attributeName=attributeValue`. In the document shown in Listing 2.2, for instance, such attributes are used as keys to uniquely identify a particular person or bone, respectively. The restriction to a hierarchical base structure again entails a less flexibility with respect to data structures than offered by file or operating systems. Nevertheless, this *flexibility* is still *higher* than in case of relational database systems. While a database table may be represented via a simple sequence of equally structured XML elements, XML additionally allows for embedding virtually arbitrary elements in a likewise arbitrary hierarchy.

XML database systems usually support several standardized command or query languages, while each supported language is tailored to a specific purpose. The purpose of the XML Path Language (XPath) is to define path expressions that enable the navigation through the hierarchical structure of an XML document [W3C14a]. As result, such an XPath expression delivers specific elements or fragments of the whole document that qualify for the path expression. The XML Query Language (XQuery) and corresponding FLWOR statements (For, Let, Where, Order by, Return) provide more sophisticated query capabilities [W3C14b]. Listing 2.3 shows an XQuery FLWOR statement that delivers a sequence of some material parameters stored in the XML document depicted in Listing 2.2. The `FOR` clause iterates over the set of bones of the person with `id "007"` using an XPath expression to select proper XML elements. The `LET` clause then binds the name of the current bone to the variable `bone_name`, while the `WHERE` clause specifies a filter according to bones that are called `"leftFemur"`. `ORDER BY` is used in this example to sort the output according to the name of the material parameters in ascending order, e. g., `hydraulicConductivity` followed by `solidDensity` and `stiffness`. Finally, the `RETURN` clause specifies how to construct the output of the query.

In addition to this query capability, the XML Schema Definition (XSD) language may be used for data definition, i. e., for defining an XML-based data structure certain XML documents need to comply with [W3C04b, W3C04c, W3C04d]. The XQuery Update Facility is used to insert, update, and delete data [W3C11]. A complementary solution for data manipulation that may be used both in isolation and together with XQuery Update are XSLT scripts (Extensible Stylesheet Language Transformations) [W3C07c]. Furthermore and similar as the SQL extension PSM, XQuery facilitates the declaration and implementation of additional functions that may be used to extend the features of the pure language.

In summary, all these languages used for XML database systems provide a similarly full range of features as SQL and its extension parts offer for relational database systems. Likewise, much research is conducted to enhance the theoretical and algebraic basis of XQuery and related languages. For instance, Ré et al. propose an algebra and an algebraic compiler for XQuery statements [RSF06]. They combine and extend ideas of (1) a tuple-based algebra that bears resemblance to the relational algebra [MHM04] and of (2) a tree-based algebra that is tailored to XML [CJLP03]. This results in a very expressive algebra that may be used to implement XQuery statements. Grust et al. discuss how XQuery statements may be expressed only via the standard relational algebra and executed using a purely relational XQuery processor [GT04]. In other words, they show that the relational algebra is sufficient to express and implement the majority of the constructs used in XQuery statements. In conclusion, the relational algebra and the major XQuery-specific algebras – and thus also the languages SQL and XQuery – have a similarly *high expressive power*.

2.2.4 Sensor Networks and Gateways to them

The major computer-based simulation applications that use sensor networks as data resources are motivated by problems investigated in the area of environmental science [GGCFEl07, Ace12, BBBD12, Hun12]. Probably the most prominent examples are earthquake simulations [KTJT03, HI08, COJ$^+$10], as well as weather predictions or climate system modeling [Lyn08, Tre10]. Here, sensor networks usually provide real-time input data for the respective mathematical simulation models. These models then allow for predicting if hazardous earthquakes or weather conditions, e. g., hurricanes, may occur in near future and how such phenomenons might geographically spread or move.

A sensor network consists of several geographically distributed sensor nodes, i. e., each sensor node is installed at a distinct location [DP10, OR11]. Adjacent sensor nodes usually have wireless connections between each other enabling communication and data exchange within the sensor network. Each sensor node embeds one or more sensors that each captures one relevant measure, e. g., seismic signals, temperature, or wind force. Further components that are relevant for data management are memory for temporary data storage, receivers and transmitters for communication between sensor nodes, as well as processing units, e. g., CPUs, for data aggregation or pre-analysis [OR11]. In a similar way as done for numerical simulation calculations, the measures of sensors are temporally discretized, i. e., each sensor samples its measure for consecutive, discrete time steps. Note that the geographic distribution of sensor nodes at distinct locations likewise corresponds to a spatial discretization of the measures.

Individual sensor nodes and/or the whole sensor network are managed by a corresponding operation system, e. g., TinyOS[3] [LMP+05]. Such operating systems may typically be accessed via application programming interfaces (APIs) or command line interfaces (CLIs). These APIs or CLIs offer basic operations to configure a sensor node, to read individual measured values from it, to transmit these values between several connected sensor nodes, or to wait for certain events, e. g., until a measured value exceeds a threshold. More sophisticated features need to be additionally implemented and embedded into sensor nodes, e. g., TinyOS allows for writing embedded C code. Gateways to whole sensor networks provide an alternative that reduces the implementation overhead for such sophisticated features. Typically, this is achieved by more expressive command or query languages and corresponding query processing engines. For instance, TinyDB[4] offers a SQL-like query language to extract data from a whole sensor network, where each sensor node is individually managed by TinyOS [MFHH05].

The kinds of data containers that are managed by different gateways to sensor networks are usually very proprietary. The default data container managed by TinyDB, for instance, is a table called `sensor` that summarizes all sampled measures of all sensor nodes in the relevant network [MFHH05]. Note that this is only a virtual database table, i. e., its tuples are only materialized as soon as a query is issued against TinyDB, and only those tuples are materialized that qualify for the query. Furthermore, the tuples are usually deleted again

[3]TinyOS: `http://www.tinyos.net/`
[4]TinyDB: `http://telegraph.cs.berkeley.edu/tinydb/`

```
SELECT AVG(fine_dust), urban_district
FROM sensors
WHERE city = 'Stuttgart'
GROUP BY urban_district
HAVING AVG(fine_dust) > 50
SAMPLE PERIOD 1h FOR 7d
```

Listing 2.4: Exemplary query to a TinyDB gateway delivering average values of fine dust that exceed the permitted maximum of 50 $\mu g/m^3$ in certain urban districts of the city Stuttgart in Germany.

after a short period of time, e. g., as soon as the query results are delivered to the client application. Each row of the table `sensor` represents the measured values of a particular sensor node and of a particular time step for which the sensor node samples its measures. Some of the table columns represent the (composite) key of a row, e. g., a sensor node id and a time stamp. Other columns, amongst others, correspond to the individual measures sampled by the sensor nodes, e. g., temperature or humidity. Hence, TinyDB imposes a specific tabular structure with specific column types for its single default data container. This leads to only a *low flexibility* regarding data structures, especially compared to relational database systems that allow for arbitrary tabular data structures. Alternative data containers of TinyDB are so-called materialized storage points. They represent a buffer in which results of a TinyDB query may be stored for a longer period of time and then be re-used by other queries.

The command or query languages offered by gateways to sensor networks are typically proprietary as well. In general, a corresponding query specifies (1) which attributes or measures shall be retrieved (2) from which spatial region and in which spatial resolution, as well as (3) for which period of time and for which sampling intervals, e. g., time steps [BKR11]. Listing 2.4 shows an exemplary SQL-like query issued against the default table `sensor` of a TinyDB gateway (FROM). This query delivers the measure fine dust (`SELECT`) in the spatial region of the city Stuttgart in Germany (`WHERE`). Thereby, the spatial resolution of the query corresponds to individual urban districts in Stuttgart, i. e., the query calculates average fine dust values for each urban district (`SELECT` and `GROUP BY`). In other words, each urban district may have several geographically distributed sensor nodes measuring fine dust, and the query interpolates the values of all sensor nodes within each individual urban district. It also includes a filter to deliver only the values of those urban districts that exceed the permitted maximum of 50 $\mu g/m^3$ (`HAVING`). Finally, the query specifies that all values shall be measured and delivered with a temporal sampling interval of one hour and for a total period of seven days (`SAMPLE PERIOD`). The set of further built-in functions that come with TinyDB includes temporal aggregations of measures or event-based triggering of query execution [MFHH05]. It is also possible to integrate user-defined functions within TinyDB queries, e. g., to signal events or to initiate other physical actions as part of a query result.

Accompanied by the restriction to query data solely from sensor networks, the *expressive power* of command or query languages offered by corresponding gateways is *usually only moderate.* So, these languages are less expressive than SQL or XQuery for database systems (see Sections 2.2.2 and 2.2.3). TinyDB, for instance, does not allow for sorting the results of a query that accesses the table `sensor`. The reason is that this table is an unbounded data stream of values and thus does not allow for such blocking operations [MFHH05]. Furthermore, relational joins are only possible in case at least one of the tables to be joined is a materialized storage point, i. e., self-joins of the table `sensor` are not feasible. Likewise, TinyDB forbids the usage of sub-queries that are directly nested within another query accessing this table `sensor`. The data manipulation and data definition features are limited as well, especially because sensor networks may usually act as data source only, but not as data sink (see Figure 2.5). It is only possible to temporarily store a query result into a materialized storage point. However, such buffered data must not be altered or re-structured later on.

2.3 Workflows and Workflow Languages

Workflows have a long track record supporting the automation and acceleration of business processes [LR00, Wes12]. A *process model* specifies the individual tasks of a real-world business process and the order in which these tasks need to be executed to achieve the corresponding process goal. Concrete executions of such a process model are called *process instances* [LR00]. The parts of a process model that are automatically coordinated by a computer system are captured using a *workflow model.* A *workflow (modeling) language* encompasses means and constructs how to specify or to design particular workflow models. Such workflow languages usually allow for designing workflow models as compositions of relevant tasks that are arranged according to causal and/or data dependencies. The workflow models may then be executed by a language-specific workflow management system (WfMS), where individual workflow executions are analogously called *workflow instances* [LR00].

Recently, the general concept of workflows has also found application in the area of computational science, and the term *scientific workflow* has been coined [TDG07]. Different application areas imply different requirements for workflows, workflow languages, and workflow systems. Section 2.3.1 therefore encompasses a classification of different kinds of workflows, i. e., business workflows, scientific and simulation workflows, as well as other related kinds. Afterwards, a demarcation of available types of workflow languages and workflow systems is given in Section 2.3.2. Finally, Section 2.3.3 describes features of a de-facto standard workflow language, namely the OASIS standard Web Services Business Process Execution Language (WS-BPEL or BPEL for short) [OAS07].

2.3.1 Classification of Workflows

Figure 2.6 shows a classification of different kinds of workflows that are relevant in the context of either business or scientific applications [RSM11]. The main classes are orchestration workflows and data-intensive workflows, and the figure associates these main classes with respective sub-classes. Furthermore, it shows which kinds of workflows are usually included in the set of scientific workflows. These are workflows that are essentially motivated by scientific applications, i. e., simulation workflows, data analysis workflows, and data curation workflows.

The main focus of *orchestration workflows* is to compose or integrate different and heterogeneous applications, as well as to define their execution order and the way how they interact with each other [LR00, Ley05]. These orchestration workflows originate from the area of business applications, where *business workflows* or production workflows realize and automate business processes [LR00, Wes12]. Another sub-class are *service provisioning workflows*, e. g., to provision or deploy certain applications in cloud environments [OAS13, VHKL13, BBK$^+$14]. Examples of such cloud-based applications are virtualized infrastructure, platform, or software services [MG11]. The most common solutions to integrate various applications in orchestration workflows are the Service-oriented Architecture (SOA) and especially the Web Services technology [Ley03, WCL$^+$05]. Hence, such workflows are oftentimes also called service orchestrations [Ley05].

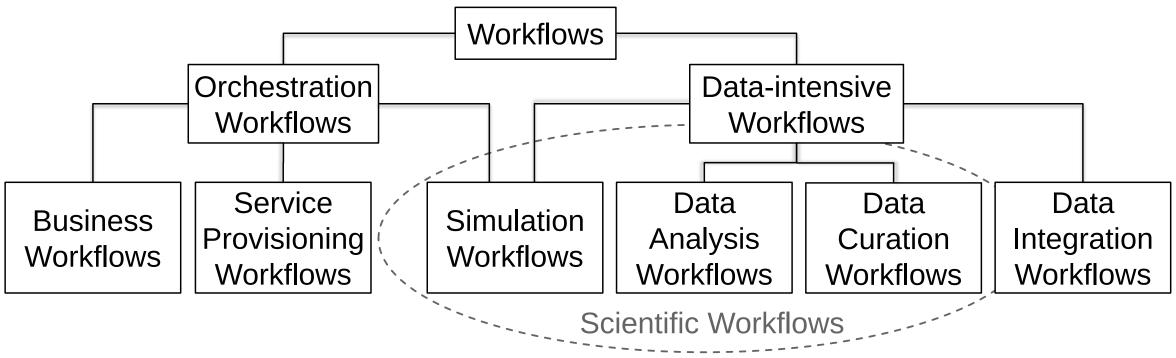


Figure 2.6: Coarse-grained classification of workflows that are relevant for business or scientific applications; cf. [RSM11].

In contrast to orchestration workflows, *data-intensive workflows* treat data and their processing as first-class citizens, instead of considering applications and services as the major artifacts. Such data-intensive workflows typically carry out a multiplicity of complex data processing steps that access huge amounts of possibly distributed and heterogeneous data. The main sub-class of them are *data analysis workflows* or pipelines [TDG07, SR09]. Their goal is to provide new insights from existing data in order to facilitate and accelerate data-intensive scientific discovery [HTT09]. Typical operations are object identification, feature discovery, and pattern matching or recognition. As an example, consider a workflow using similarity search and classification to detect patterns among chemical compounds [KWK$^+$09]. Other major examples aim at searching for certain structures or properties in proteins or genomes [BTS07, DCBS$^+$10]. Nevertheless, workflows of this kind may also employ techniques for data visualization, instead of only relying on purely analytical approaches [FSC$^+$06].

The second sub-class of data-intensive workflows encompasses *data curation workflows* [DCM$^+$12, Son15]. Their goal is a long-time preservation of massive amounts of data, especially ensuring that data is always fit for contemporary purpose and that it may be discovered and re-used over time [LMLG04]. This mainly includes activities to assess and improve data quality [Sad13], e. g., based on corresponding data cleaning techniques [RD00]. The main goal of *data integration workflows* is to provide data integration or provisioning processes for superordinate applications or workflows. This encompasses, amongst others, sophisticated operations for loading or retrieving a bulk of data, filtering a data set, or merging two data sets [MMLW05]. Furthermore, such workflows may also carry out data cleaning techniques, i. e., to improve data quality in a similar way as

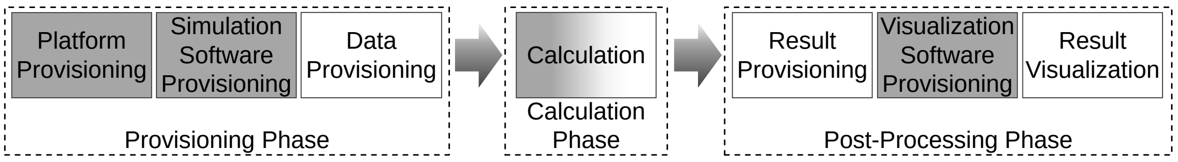


Figure 2.7: The common phases of a simulation workflow shown in Figure 2.4 and whether individual sub-phases may rather be implemented via orchestration workflows (grey background) or data-intensive workflows (white background). Depending on the concrete simulation example, the calculation phase may either be treated as orchestration workflow or data-intensive workflow.

done by data curation workflows [RD00]. Prominent examples of data integration workflows are ETL processes, e. g., to upload data into a data warehouse [VZ14] or to collect and pre-process scientific or simulation data [RLSR+06, RRS+11, RS14]. The set of examples additionally includes data mashups that facilitate flexible and ad-hoc data integration scenarios [DM14, HRWM15].

Simulation workflows, e. g., the running example depicted in Figure 1.1, are the last relevant sub-class of workflows and the main focus of this thesis. As indicated in Figure 2.6, simulation workflows cannot be clearly associated with either the class of orchestration workflows or the class of data-intensive workflows. Figure 2.7 depicts which of the common sub-phases of a simulation workflow shown in Figure 2.4 are usually treated as orchestration workflow or as data-intensive workflow. The main purpose of simulation workflows is to execute a set of long-running numerical calculations in their calculation phase. So, they need to coordinate the execution order of and the interaction between different simulation software that implement these numerical calculations. This focus on execution order and interaction of applications makes the calculation phase of a simulation workflow a good candidate to be treated as orchestration workflow [GSK+11]. This also holds for the sub-phases platform provisioning, simulation software provisioning, and visualization software provisioning that may be implemented via variants of service provisioning and thus orchestration workflows [VHKL13].

Nevertheless, some phases of a simulation workflow may also be implemented via data-intensive workflows. In particular, the sub-phases data provisioning and result provisioning are common examples of data integration workflows, as they provision and prepare the input data of other phases. Even more data-intensive data integration workflows have to be executed in multi-scale simulation couplings. Here, complex operations for data filters, data format conversions, and especially interpolations and extrapolations are required to overcome the differences of different simulation tools and of varying scales in numerical calculations. Finally, the sub-phase result visualization may also be seen as a special variant of a data analysis workflow. Albeit it is often more appropriate to treat the calculation phase of a simulation workflow as orchestration workflow, a few examples of simulation calculations are rather tailor-made for the data-intensive counterparts. For instance, this concerns the examples originating from environmental science illustrated in Section 2.2.4, since these examples usually exhibit a continuous and real-time nature of processing massive amounts of sensor data.

2.3.2 Workflow Languages and Workflow Systems

The different classes of workflows depicted in Figure 2.6 imply different requirements for workflow languages and workflow systems. The major types of workflow languages may be distinguished between dataflow-oriented and control-flow-oriented languages [LWMB09, RSM11]. Table 2.2 summarizes the main differences between these two concepts of workflow languages. In particular, it indicates (1) the key artifacts considered by relevant languages, (2) the major benefits typically provided by corresponding workflow systems, and (3) the workflow classes shown in Figure 2.6 to which the languages are usually tailored.

A *dataflow* forms a directed graph, where the nodes are the individual tasks of a workflow and the edges define data dependencies between these tasks [JHM04]. Each task is associated with a set of named ports for receiving or sending data, i. e., one or more input ports and one or more output ports. The data dependencies of a dataflow then connect output ports and input ports of different tasks. Thereby, they correspond to unidirectional channels over which data streams or individual data items are sent between the tasks [Mor11]. All tasks of a dataflow are active at the same time and wait for a certain number or combination of data items that arrive at their input ports. Then, each task processes arrived data items according to its functional definition. For example, it transforms data items to another format or executes more sophisticated analytical operations as illustrated for data analysis workflows in Section 2.3.1. Afterwards, a task forwards the resulting data items over its output ports and over the outgoing data channels to either the input ports of other tasks succeeding in the dataflow or to the final output of the workflow. The previous task may then concurrently process further data items that have meanwhile arrived at its own input ports.

Table 2.2: Demarcation and main characteristics of dataflow-oriented and control-flow-oriented workflow languages and corresponding workflow systems.

Language concept	**Key artifacts**	**Major benefits of corresponding systems**	**Supported workflows**
Dataflow	Data dependencies and data processing	Efficient execution of single workflow instances; Capturing / reconstructing data provenance	Data-intensive workflows
Control flow	Causal dependencies and decisions	Robust execution and high throughput of multiple workflow instances; Modular workflow design and many expressive modeling constructs; Standard-based solutions	Orchestration workflows, but increasingly data-intensive workflows

This data-centric focus makes dataflow-oriented workflow languages the intuitive approach to model and implement most kinds of data-intensive workflows shown in Figure 2.6. The dataflow concept enables various kinds of optimization techniques to ensure the efficient execution of single instances of these data-intensive workflows. This encompasses data parallelism within individual workflow tasks or pipeline parallelism between several tasks [PA06, LAB+09]. In addition, some solutions allow for seamlessly employing scalable data processing infrastructures, such as high performance computing (HPC) environments or MapReduce [COdO+10, ZBKL10]. Another benefit is that a dataflow allows for linking the conceptual data specifications in the workflow model with the actually processed or generated data after a workflow has been executed. This significantly helps to capture, reconstruct, and understand the provenance of data, facilitating the reproducibility of workflow execution [MBBL15]. Many products exist that show proprietary solutions in terms of modeling languages, optimization techniques, execution engines, and provenance support. For instance, this includes data stream processing platforms, e. g., IBM Streams[5] or NexusDS [CEB+09]. Moreover, most scientific workflow systems exploit a dataflow-oriented approach. Examples are Kepler [LAB+06], Pegasus [DSS+05], Taverna [OGA+06], Triana [CGH+06], and VisTrails [FSC+06].

[5]IBM Streams: `http://www-03.ibm.com/software/products/en/ibm-streams`

A *control flow* likewise constitutes a directed graph with workflow tasks being the nodes of the graph. However, the edges are not data dependencies, but causal dependencies [LR00, Wes12]. These causal dependencies define a strict execution order among all workflow tasks. Thereby, a workflow instance starts with one of the tasks that constitute the initial nodes of the graph, i. e., that do not have any incoming edges. Afterwards, the execution of any other task may only start after all preceding tasks within the control flow graph have successfully and completely finished their execution. Workflow execution ends when one of the final tasks, which do not have any outgoing edges, has been executed successfully. Most workflow languages also allow for expressing imperative or procedural elements. For instance, this includes gateway tasks supporting the conditional execution of a subset of several outgoing control flow branches. Other examples are loop structures that embed another control flow of tasks as sub-workflow and that define how to repeat the execution of this sub-workflow [LR00, OAS07, Wes12].

The major use cases of control-flow-oriented workflow languages are orchestration workflows. Hence, this includes business workflows, service provisioning workflows, and many parts of simulation workflows as shown in Figure 2.7. Especially business workflows often require the simultaneous execution of a large number of workflow instances [LR00]. Most of the workflow systems relying on control-flow-oriented languages support this high workload via transactional features and solutions to workflow recovery [Ley95, EL96]. So, these systems ensure the robust execution and a high throughput of multiple workflow instances, instead of focusing on single instances only. Other benefits are the possibilities for a modular workflow design and the support of expressive modeling constructs. Examples are sophisticated control flow branches, many features to interact with other workflows or users, as well as fault, compensation, and event handling capabilities at the workflow level [LR00, OAS07, Wes12]. Furthermore, the existence of the OASIS standard WS-BPEL [OAS07] leads to standard-based solutions for workflow systems. This in turn entails an extensive tool support and a good portability of workflow models between different workflow systems [GSK$^+$11].

Nevertheless, control-flow-oriented workflow languages are increasingly adopted for data-intensive workflows as well [MMLW05, BHW$^+$07, Slo07, GHCM09, SSH$^+$10, GSK$^+$11]. This is in part due to the benefits offered by these languages and corresponding workflow systems as described above. Transaction and recovery concepts are important to ensure the robust and reliable execution of long-running data-intensive workflows [AMA06]. The offered expressive modeling constructs may simplify modeling complex workflows, e. g., when these workflows require some degree of control flow or certain fault handling capabilities [SSH+10]. Furthermore, the standard-based solutions, e. g., as provided by WS-BPEL, lead to common, unified modeling languages for both data-intensive workflows and related orchestration workflows. This is not only interesting for single simulation workflows, which anyway consist of data-intensive and orchestration (sub-)workflows, as shown in Figure 2.7. It moreover allows for a generic approach to model and execute different workflows of various scientific domains, i. e., in multi-scale simulation couplings. It even facilitates the combination of scientific and business processes, e. g., assume a crash test simulation embedded in a vehicle development and testing process [JMG11]. Resulting detailed benefits of this generic approach include a seamless modeling environment, which avoids the overhead of getting accustomed to many different tools or technologies [CWGN11]. In addition, it enables a holistic workflow optimization across different kinds of data-intensive and orchestration workflows [VSS+07].

Another complementary motivation to use control-flow-oriented workflow languages for data-intensive workflows is given by approaches to recover dataflow information from a particular control flow of workflow tasks [KKL08a, KKL08b]. This additionally offers some of the major benefits that are otherwise provided by dataflow-oriented workflow languages and corresponding workflow systems. In other words, it facilitates the optimized execution of single workflow instances [BHW+07, VSS+07]. Furthermore, it helps to reconstruct data provenance information for reproducibility purposes [MSK+15, MBBL15].

In summary, control-flow-oriented workflow languages provide data-intensive workflows with many additional benefits, while they may still offer some of the major features that are provided by dataflow-oriented languages. The question whether to use either control-flow-oriented or dataflow-oriented languages of course cannot be universally answered for all examples of data-intensive workflows. Nevertheless, the above discussion emphasizes that control-flow-oriented languages may be good candidates to model and execute simulation workflows. This is especially underpinned because (1) simulation workflows are anyway combinations of both data-intensive and orchestration (sub-)workflows and because (2) it leads to a generic, standard-based approach facilitating multi-scale simulation couplings. Due to these reasons, this thesis treats control flow as the main concept for workflow languages realizing simulation workflows. Furthermore, it discusses some extensions to control-flow-oriented workflow languages that make these languages even more suitable for data-intensive workflows (see especially Section 1.2.2). Likewise, the de-facto standard WS-BPEL is used as basis to develop all prototypical implementations of workflows, as well as of extensions to workflow languages and workflow systems. However, note that this thesis still discusses whether and how all proposed concepts and extensions may also be applied to dataflow-oriented workflow languages.

2.3.3 Web Services Business Process Execution Language

WS-BPEL (or BPEL for short) is the de-facto standard language to model and execute workflows based on the control-flow-oriented orchestration of service interactions [OAS07]. It fosters the concept of a Service-oriented Architecture (SOA) and especially the Web Services technology [Ley03, WCL$^+$05]. Hence, the services orchestrated in a BPEL workflow are provided as Web Services. The first main part of the Web Services technology is the Simple Object Access Protocol (SOAP) [W3C07a]. It is a lightweight protocol for exchanging structured information – encoded in XML-based messages – in a distributed environment. In the context of BPEL, SOAP is used to exchange messages between a BPEL workflow and the Web Services the workflow orchestrates. The interfaces of Web Services are described using the likewise XML-based Web Services Description Language (WSDL) [W3C01, W3C07b]. Note that the interface of a BPEL workflow itself is described via WSDL, too. So, BPEL workflows may seamlessly call other BPEL workflows, enabling a hierarchical workflow design [OAS07].

BPEL employs WSDL version 1.1 [W3C01, OAS07]. Listing 2.5 shows an excerpt of exemplary WSDL definitions that describe the abstract interface of a simple Web Service delivering stock quotes [W3C01]. Firstly, this WSDL document contains definitions of data *types*, e. g., using XSD. These data types are used to encode certain parts of the subsequently defined *messages* the service may receive or send. The WSDL document shown in Listing 2.5 defines two messages `getLastTradePriceInput` and `getLastTradePriceOutput`. Each of these messages has a part named `body` of the previously defined types `tradePriceRequest` and `tradePrice`, respectively. Thereupon, *port type* definitions declare a set of named *operations* that correspond to the actions supported by the service. The declaration of an operation specifies its input and output messages, as well as possible fault messages. The example WSDL document shown in Listing 2.5 defines one port type with one operation. The first message `getLastTradePriceInput`

```
<definitions name="stockQuote"
             targetNamespace="http://example.com/stockquote/definitions"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
    <xsd:schema>
      <xsd:element name="tradePriceRequest" type="xsd:string" />
      <xsd:element name="tradePrice" type="xsd:float" />
    </xsd:schema>
  </types>

  <message name="getLastTradePriceInput">
    <part name="body" element="tradePriceRequest"/>
  </message>
  <message name="getLastTradePriceOutput">
    <part name="body" element="tradePrice"/>
  </message>

  <portType name="stockQuotePortType">
    <operation name="getLastTradePrice">
      <input message="getLastTradePriceInput"/>
      <output message="getLastTradePriceOutput"/>
    </operation>
  </portType>
  ...
</definitions>
```

Listing 2.5: Excerpt of an exemplary WSDL document defining the abstract interface of a service delivering stock quotes; cf. [W3C01].

described above constitutes the input of this operation, whereas it delivers the message `getLastTradePriceOutput` as output.

Additional WSDL constructs not shown in Listing 2.5 allow for defining specific service bindings and a set of service ports. A service binding is associated with a WSDL port type and defines the concrete transport protocol and physical message formats to be used for communicating with the Web Service. A service port can be seen as a concrete instance of a port type, i. e., it defines an endpoint of a particular service deployment that supports the relevant port type. Such an endpoint definition corresponds to a combination of an appropriate service binding and of an endpoint address, e. g., a Uniform Resource Locator (URL).

Listing 2.6 exemplifies key elements of the workflow language BPEL in that it specifies a BPEL workflow implementing the Web Service interface defined in Listing 2.5. Therefore, the BPEL workflow specification firstly imports the corresponding WSDL document (line 5 in Listing 2.6). The remainder of the workflow is subdivided into global declarations and the actual execution logic.

The *global declarations* of a BPEL workflow usually start with the definition of a set of *partner links*. Each partner link is associated with a *partner link type* as shown in Listing 2.6. Partner link types are BPEL-specific WSDL extensions [OAS07]. They define feasible conversational relationships between the port types of two Web Services. Furthermore, they associate these port types with abstract roles each of the services may play in a conversation. A partner link definition in a BPEL workflow then concretizes whether a particular role is played by the BPEL workflow itself (`myRole` as shown in Listing 2.6) or by another partner Web Service (`partnerRole`). This likewise indicates whether the port type associated with this role is provided by the BPEL workflow or by a partner Web Service. Moreover, such partner links or their roles act as containers for concrete endpoint references to corresponding workflow or service deployments, e. g., using Web Services Addressing (WS-Addressing) [W3C04a].

In addition, BPEL makes use of *variables* to represent the state of a particular workflow instance [OAS07]. Listing 2.6 exemplifies the declaration of two variables `request` and `response`. The attribute `messageType` within these declarations indicates that the variables act as containers for storing the messages received or sent by the BPEL workflow. Furthermore, such an attribute points to the concrete WSDL message definition specifying the kind and structure of a corresponding message. In this example, `request` constitutes the input message and `response` the output message of the BPEL workflow shown in Listing 2.6 and of the underlying Web Service defined in Listing 2.5. Nevertheless, the state of a BPEL workflow not only consists of its messages, but also of intermediate data that may be used in the control flow logic or to compose certain parts of messages. For that purpose, BPEL allows for declaring further kinds of variables using XML Schema types or XML Schema elements [W3C04d, OAS07].

BPEL features additional kinds of global declarations that are not included in the BPEL workflow shown in Listing 2.6 [OAS07]. For instance, *fault handlers* enable the definition of activities that must be triggered when certain faults happen during workflow execution. This includes standard or user-defined faults that may be caused by the BPEL workflow itself or by the Web Services this

```
<process name="stockQuoteWorkflow"
          targetNamespace="http://example.com/stockquote/workflow"
          xmlns:defs="http://example.com/stockquote/definitions"
          xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable">
  <import namespace="http://example.com/stockquote/definitions"
      location="http://example.com/stockquote/stockquote.wsdl"/>

  <!-- Global declarations -->
  <partnerLinks>
    <partnerLink name="stockQuoteLink"
      partnerLinkType="defs:stockQuoteLinkType" myRole="stockQuoteService"/>
  </partnerLinks>

  <variables>
    <variable name="request" messageType="defs:getLastTradePriceInput"/>
    <variable name="response" messageType="defs:getLastTradePriceOutput"/>
  </variables>

  <!-- Execution logic -->
  <sequence>
    <receive partnerLink="stockQuoteLink" portType="defs:stockQuotePortType"
      operation="getLastTradePrice" variable="request" createInstance="yes"/>

    <assign>
      <copy>
        <from>
          <literal>
            <defs:tradePrice>10</defs:tradePrice>
          </literal>
        </from>
        <to variable="response" part="body"/>
      </copy>
    </assign>

    <reply partnerLink="stockQuoteLink" portType="defs:stockQuotePortType"
      operation="getLastTradePrice" variable="response"/>
  </sequence>

</process>
```

Listing 2.6: Example of a BPEL workflow implementing the Web Service interface defined in Listing 2.5.

workflow calls. In a similar way, *event handlers* run concurrently to the ordinary workflow execution and wait for certain events. Such events may correspond to (1) messages that are exceptionally sent to the relevant workflow instances or to (2) user-defined alarms that, e. g., are triggered after a specific time period has elapsed. Each event handler then again defines a set of activities that must be executed in the occurrence of the relevant event.

To define the actual *execution logic* of a BPEL workflow, i. e., a control flow of several workflow tasks, BPEL offers various types of activities [OAS07]. The set of activity types is subdivided into basic activities and structured activities. Basic activities represent elementary actions, such as sending or receiving a message. In contrast, structured activities embed other basic or again structured activities and define the control flow among these embedded activities. Thereby, the execution logic of a BPEL workflow is specified by one main activity [OAS07]. Since a typical workflow does not consist of only one elementary workflow task, this main workflow activity is commonly a structured activity.

The example workflow shown in Listing 2.6 employs one structured `sequence` activity that embeds three basic activities `receive`, `assign`, and `reply`. The `sequence` activity indicates that these embedded activities are to be executed sequentially in lexical order. The `receive` activity waits for a certain message to arrive. Its major attributes point to a previously defined partner link indicating the communication partners of this message exchange, as well as to the WSDL port type and operation to be implemented by the `receive` activity. The attribute `variable` refers to the message variable storing the arrived message. Furthermore, `createInstance` may be used to specify if the arrival of a message in a `receive` activity triggers a new BPEL workflow instance or not. The subsequent `assign` activity copies an XML literal value to the message part `body` of the message variable `response`. The final `reply` activity sends the content of this message variable back to the client of the BPEL workflow by using the same partner link, port type, and operation as the previous `receive` activity.

The `assign` activity of BPEL facilitates much more complex variable modifications than illustrated by the simple example shown in Listing 2.6 [OAS07]. Firstly, it may encompass multiple `copy` statements that all are executed in one single transaction. Each `copy` statement may embed sophisticated `from` and `to` specifications that respectively determine which source data shall be copied to which target. This includes built-in functions, e. g., for copying individual WSDL message parts or the endpoint references stored in partner links. The expressive power may even be increased by using expression or query languages within `from` and `to` specifications. By default, BPEL supports XPath version 1.0 [W3C99] and XSLT [W3C07c]. Modern workflow systems even extend the BPEL standard to support newer and more powerful expression or query languages [KGK+11]. For instance, the Apache Orchestration Director Engine (Apache ODE) supports XPath version 2.0 [W3C10a] and XQuery version 1.0 [W3C10b][6].

Another major basic activity is the `invoke` activity that allows for calling a one-way or request-response WSDL operation of another partner Web Service or BPEL workflow [OAS07]. User-defined faults may be thrown from within a BPEL workflow by using the `throw` activity. The `wait` activity facilitates waiting for a specific time period or until a certain point in time has been reached. The counterpart of the structured `sequence` activity is the `flow` activity that enables the concurrent execution of its embedded activities. Additional `link` declarations may be used to specify explicit control flow dependencies between these embedded activities. Conditional control flow branches based on data expressions are provided by either these `link` declarations or by the `if` activity. The `pick` activity allows for an event-based selection of specific branches. Various kinds of loop structures to define the repeated execution of nested activities are offered by the `while`, `repeatUntil`, and `forEach` activities. The `scope` activity represents a nested scope in the BPEL workflow with its own declarations for variables, partner links, fault handlers, event handlers, and other modeling constructs. More information about all these or other activities, as well as about further BPEL constructs can be found in the BPEL specification [OAS07].

All these various kinds of activities and modeling constructs make BPEL a highly expressive control-flow-oriented workflow language. BPEL even allows for extending the standard language and this way for further increasing its expressive power. In particular, the possibilities for variable modifications offered by the `assign` activity may be extended by so-called *extension assign operations*. Other explicit extension constructs are *extension activities* that facilitate completely new and customized activity types. Prominent examples enable the coordinated execution of sub-processes that may be re-used across multiple BPEL workflows [KKL+05], as well as the integration of human interactions within individual workflows [OAS10]. Especially this extensibility makes BPEL a good choice as base workflow language that may be extended by new data-aware activity types as proposed by this thesis (see for instance Section 1.2.2).

[6]Apache ODE BPEL extensions: `http://ode.apache.org/extensions/`

Chapter 3

Data Management in Simulation Workflows

The previous chapter provides background information with respect to three separate topics that are relevant for this thesis: (1) computer-based simulations and mathematical simulation modeling, (2) most commonly used data resources and the command languages these resources offer, as well as (3) workflow technology, workflow languages, and workflow systems. This chapter now combines these three topics in that it circumscribes the most relevant aspects with respect to data management in simulation workflows. Section 3.2 details common data characteristics of computer-based simulations in terms of typical kinds of data, their data formats, and data sizes. Afterwards, Section 3.3 illustrates basic data management patterns describing common operational aspects that frequently occur in simulation workflows. These common data characteristics and data management patterns have been derived by investigating the same use cases for simulations as described for the identification of research challenges in Section 1.2. So, this includes several academic use cases described in literature [Har96, GZC14] and especially a set of real-world simulations or related applications [KSR+11, KAK+13, Kra14, FDP15, FE11, RK11, Wag10]. Section 3.1 therefore gives a brief overview of the real-world simulations or related applications that are considered as main use cases in this thesis. Finally, Section 3.4 summarizes the major aspects of all these discussions and illustrations. Parts of this chapter correspond to revised and composite versions of excerpts of previous author publications that are cited at affected locations [RSM11, RS13b, RSM14a, RSM14b, RWWS14].

3.1 Considered Example Simulations

The following list summarizes the real-world example simulations or related applications that are considered as main use cases in this thesis:

- The first main use case is the *simulation of structure changes in bones*, i. e., the running example illustrated in Sections 1.1 and 2.1.1 [KSR$^+$11, KAK$^+$13, Kra14]. As described in these sections, this multi-scale simulation couples simulation models and methods from the scientific domains bio-mechanics and systems-biology.
- *Simulations of catalysis reactions* may be used to assess whether certain chemical compounds can serve as catalysts for certain chemical reactions. Rommel et al. investigate how the enzyme glutamate mutase catalyzes the conversion of glutamate into methyl aspartate [RK11]. This chemical reaction is mainly relevant for the metabolism of carbon-based life forms. The authors propose a multi-scale approach coupling different simulation models and methods of varying levels of granularity. This includes molecular-dynamic, molecular-mechanical, and quantum-mechanical calculations.
- Franzelin et al. study the *damage and crack propagation in a plate* that has been hit by a high-speed projectile [FDP15]. They use a macroscopic simulation method based on peridynamics [SA05]. Furthermore, they describe the overall setting as a data-intensive *uncertainty quantification problem* in order to investigate the sensitivity of the peridynamic method [IH88, LMK10].
- Elastic multi-body simulations, e. g., product tests in engineering or the holistic investigation of the human skeleton [Lar01, RKH$^+$10], require to solve highly complex numerical simulation models. *Model reductions*, e. g., as proposed by Fehr et al. [FE11], may be used to reduce the number of degrees of freedom in such complex numerical models. The major goal is to speed up subsequent simulation calculations, but without losing too much precision in the computation. So, model reductions are actually not simulations, but they prepare subsequent simulation calculations. Nevertheless, the workflows realizing model reduction processes have a very similar common structure as shown for simulation workflows in Figure 2.4 [Rem11]. So, they also imply similar challenges with respect to data provisioning.
- All the use cases listed above correspond to variants of simulation workflows according to the classification of workflows depicted in Figure 2.6. The

last use case considered in this thesis is however a data analysis workflow. The main intention behind including such a more data-intensive scientific workflow is to be able to better evaluate the generality of all approaches discussed in this thesis and to potentially broaden their scope of application. The use case is a *protein modeling workflow* identifying or investigating proteins that can solve specific chemical or biological problems [Wag10]. The methods used in this workflow hence originate from the scientific domains life sciences and bio-chemistry [BTS07]. The workflow firstly extracts a list of protein sequences from a protein database, such as GenBank [BKML$^+$10]. It then iterates over this list and uses pattern matching to find important regions within individual protein sequences. For instance, it searches for amino acids that are needed for certain chemical reactions.

Altogether, the examples listed above constitute a wide set of applications covering different simulation models and methods of a multiplicity of important scientific domains. Hence, they represent a good base for both developing and evaluating the approaches discussed in this thesis. The use case listed first, i. e., the running example of a bone simulation, is the largest and most complex scenario of all considered examples. So, it is used as primary use case to discuss the most important aspects throughout the whole thesis. Furthermore, the protein modeling workflow is used at special locations where its data-intensive nature is more appropriate for the relevant illustrations, e. g., in Chapter 7. Nevertheless note that all other examples listed above are still used occasionally, e. g., to discuss evaluation results regarding the generality of proposed approaches.

3.2 Characteristics of Simulation Data

As discussed in Sections 1.1 and 2.1.1, the most challenging scenarios of providing data for computer-based simulations can be classified into (1) providing and preparing heterogeneous input data for single simulation models and (2) exchanging data between different coupled simulation models. Sections 3.2.1 and 3.2.2 respectively deal with these two kinds of scenarios and describe the most relevant characteristics of simulation data and of their processing. Thereby and as discussed above, the bone simulation described by Krause et al. [KSR$^+$11, KAK$^+$13, Kra14] is used to exemplify the general discussion about common data characteristics.

Table 3.1: Common characteristics of simulation data.

Kind of data	**Most common data formats**	**Typical data size**
Geometry	Proprietary text or binary files	100 KBs to 10 GBs
Material parameters	Proprietary text files; XML documents or XML database systems	1 KB to 1 MB
Boundary conditions	CSV-based files; seldom SQL database systems	10 KBs to 1 GB
Initial conditions	CSV-based files; sometimes SQL database systems	100 KBs to 10 GBs
Method parameters	Proprietary text files; seldom SQL database systems	1 KB to 1 MB
Simulation output	CSV-based files; SQL database systems; proprietary text or binary files	1 MB to 100 TBs

3.2.1 Provisioning of Input Data for Single Simulations

The main part of the running example of a bone simulation is the bio-mechanical simulation model, as illustrated in Sections 1.1 and 2.1.1. This simulation model is numerically realized by the Pandas software[1], which is based on the Finite Element Method (FEM) [ZTZ13]. The FEM and related numerical methods are the most popular solutions to discretize and solve the majority of real-world dynamic simulation models [GRS07]. The primary kinds of input data for such numerical methods describe (1) a geometry, (2) material parameters, (3) boundary conditions, (4) initial conditions, and (5) method-specific parameters (see also Section 2.1.1, as well as Figures 1.1 and 2.1). The major output data of a simulation correspond to the unknown variables of the relevant simulation model. Table 3.1 summarizes the most relevant data formats that are commonly used to represent these kinds of input and output data. Furthermore, it indicates the magnitudes of typical data sizes as they are read or written by one particular simulation run. Figure 3.1 exemplifies the data formats that are specifically required by the Pandas software realizing the bio-mechanical bone simulation. Furthermore, it indicates how the input data formats shown in Figure 1.1 are transformed by the underlying workflow so that Pandas can properly read them.

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

The *geometry* describes the shape or topology of the object to be simulated, e. g., the geometrical shape of a bone (see Figure 2.2a). In the FEM or related numerical methods, this oftentimes corresponds to a description of the finite element grid. As shown in Figure 3.1, the Pandas software realizing the bone simulation requires three *ASCII text files* as input representing the geometry. One file contains spatial coordinates of individual nodal points of the finite element grid. The second file describes the topology of the grid, i. e., how several nodal points are arranged to individual finite elements. The last file indicates which of the nodal points and finite elements are located at the outer surface of the whole grid. Different simulation tools however require likewise different data formats. In fact, there exists a vast range of *proprietary text or even binary file formats*, especially for representing 3D geometries [MB08]. The size of such geometry files depends on the size and complexity of the object to be simulated, as well as on the resolution and accuracy to represent this object. Typically, this is related to the number of elements and nodal points in the finite element grid. For instance, the three files describing the geometry in the bone simulation have a total size of about 2 MBs and describe 3676 finite elements as well as 17130 nodal points. In bigger and more complex simulation applications, especially in engineering, this data size may reach *several 100 MBs or even multiple GBs* [HG05].

Material parameters describe further properties of the object to be simulated, e. g., the stiffness or hydraulic conductivity of a bone tissue [Kra14]. Usually, each parameter is represented by a name, by its value, and by an optional unit. The most common data formats are again *proprietary ASCII text files.* For instance, Pandas expects such a text file, where each relevant material parameter is stored in a separate row of the form `parameter name = parameter value`. Alternatives are *XML documents or XML database systems* [BS00, RS14] (see Listing 2.2 and Figure 3.1). Compared to geometry data, this kind of input data features a smaller data volume, as it only covers a limited set of parameters and their values. The magnitude of the typical data size is *between 1 KB and 1 MB*.

Boundary conditions represent parts of the solution a simulation model needs to have at the boundary or outer surface of the problem domain. As discussed in Section 2.1.1, the bone simulation considers the time-dependent external load on the bone, i. e., caused by muscle forces and by joint contact forces of adjacent bones [Kra14]. Pandas expects one *CSV-based file* (comma-separated values) for each relevant muscle and adjacent bone. Each of these files stores one corresponding force vector for each numerical time step of the simulation. Thereby,

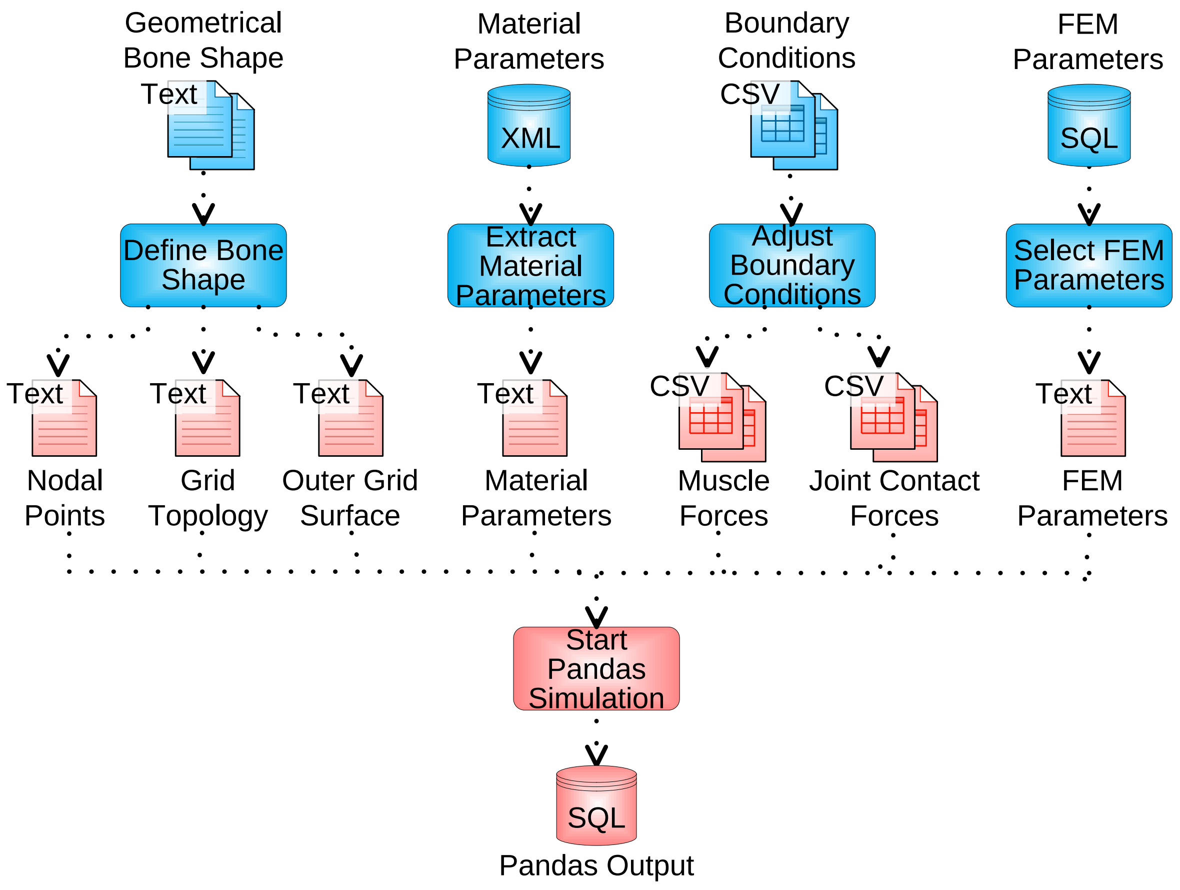


Figure 3.1: Data formats required by the Pandas software and corresponding input transformations carried out by the workflow depicted in Figure 1.1.

one column of a CSV-based file identifies the numerical time step, and three further columns represent the values of the relevant force vector. Simulation tools commonly impose such a tabular base structure on data formats for boundary conditions, because they bear much resemblance to (low-)dimensional data. This also makes *SQL database systems* possible candidates, but such systems are used rather seldom to store boundary conditions. One reason is that the data size is rather limited. For instance, the boundary conditions of the bone simulation usually have a size *between 10 KBs and 50 MBs*. This data size mainly depends on the number of time steps and on the number of different kinds of boundary values, such as different muscle or contact forces. Other applications may exhibit larger sets of data, but typically *not much more than 1 GB*.

Initial conditions are a solution of most of the unknown variables of a simulation model, but only for the first time step of the simulation. They usually encompass values for each relevant unknown variable and for each spatial evaluation point of the finite element grid, e. g., for each of its nodal points and/or integration points (see Figure 2.2b). This again bears much resemblance to dimensional data and thus argues for a tabular base structure, i. e., in *CSV-based files or sometimes in SQL database systems.* Depending on the numbers of relevant spatial evaluation points and unknown variables, the typical data size ranges *between 100 KBs and 10 GBs.* Note that the bio-mechanical simulation does not require explicit initial conditions as input data, as discussed in Section 2.1.1.

The last kind of input data do not describe aspects of a simulation model, but *specific parameters of the numerical method* used to discretize a model. Here, the Pandas software requires a *proprietary text file* as input. This file for instance specifies the base functions used to spatially discretize the calculation of the bio-mechanical unknown variables according to the FEM (see Section 2.1.1). Other FEM-specific parameters define the time discretization, e. g., the sizes of numerical time steps. *Proprietary text files* are the most common data formats, as the types of such method-specific parameters are likewise proprietary. In some scenarios, a *SQL database system* or other kinds of database systems might be used as well. For instance and as shown in Figure 3.1, the FEM-specific parameters Pandas requires may originate from a SQL database system that manages multiple method-specific parameters for several simulation applications.

The *simulation output*, i. e., the unknown variables determine how initial conditions evolve over time, given the geometry, material parameters, boundary conditions, and method-specific parameters as input. As illustrated in Section 1.1, Pandas calculates up to 20 of such unknown variables for each numerical time step and for each spatial evaluation point of the finite element grid. It then stores the resulting values in a *SQL database* [KAK$^+$13]. In general, the set of data formats used to store simulation outputs features the highest degree of heterogeneity. Relational *SQL database systems* may be attractive for their efficient and robust capabilities to manage big and complex amounts of data [HG05]. Nevertheless, many simulation applications still rely on *CSV-based files*, which are a bit more flexible with respect to data organization. In some use cases, even *proprietary text or binary files* are the solution of choice – usually when proprietary needs call for tailored low-level data structures. The typical data size of the bio-mechanical simulation outcome ranges between 100 MBs and several GBs [RSM14a]. These lower and upper bounds may significantly vary in other applications, e. g., large-scale simulations may produce *up to 10 or even 100 TBs* of result data [HG05].

As Table 3.1 shows, the most common way to manage data in simulations is based on files, especially on structured CSV- or XML-based files and unstructured text or binary files. The main reason is that files offer a high flexibility with respect to data organization and data structures, as discussed in Section 2.2.1. In particular, files may be flexibly organized and grouped in hierarchical and well-documented directory structures [MBBL15]. Furthermore, text or binary files allow for virtually any low-level data structure, which may often be tailored to specific applications. Most of the simulation examples listed in Section 3.1 adopt a file-based data management as well. For instance, both the simulation of catalysis reactions described by Rommel et al. [RK11] and the model reduction example proposed by Fehr et al. [FE11] solely rely on proprietary text files. The simulation example studied by Franzelin et al. mainly uses proprietary low-level data structures that are compressed in gzip files[2] [FDP15]. The protein modeling workflow discussed by Wagner firstly accesses a protein database and then manages its data as XML documents [Wag10]. As discussed in Section 2.2.4, some rare examples, e. g., from the area of environmental sciences, even use sensor networks as parts of their input data resources.

However, the flexibility with respect to data organization offered by files leads to a high degree of heterogeneity with respect to different kinds of proprietary data formats. Furthermore, the input data of a simulation oftentimes originate from other data sources or software tools that manage their data in different formats than needed by the used simulation tools [RLSR$^+$06, RS14]. Hence, simulation workflows have to carry out many complex data provisioning tasks to overcome the intrinsic heterogeneity of data formats. This is also highlighted by Figure 3.1 that indicates the data transformations required by the bio-mechanical bone simulation. For instance, the first data provisioning step preparing the geometry files may have to carry out more or less sophisticated coordinate transformations [Gat14]. This may be necessary in order to adapt the nodal points and the topology of the finite element grid to the coordinate system expected by the Pandas software. In engineering applications, CAD tools may even deliver a CAD model of a geometry that firstly needs to be discretized and transformed into a description of a finite element grid [HG05]. The second data provisioning step shown in Figure 3.1 needs to filter proper material parameters from an XML database system, e. g., using the XQuery statement shown in Listing 2.3. Then, it needs to store the filtered parameters in the Pandas-specific text file format. The next

[2]gzip: `http://www.gzip.org/`

data provisioning step adjusting the boundary conditions has to select proper CSV-based files containing the force vectors of muscles and adjacent bones that are connected to the bone to be simulated. Finally, the last data provisioning step has to select relevant FEM-specific parameters from the underlying SQL database system and to export them into text files. This may be based on a SQL statement that is similar to the one shown in Listing 2.1.

3.2.2 Data Exchange between Different Simulations

The complexity of data provisioning in simulation workflows is even multiplied in case data needs to be exchanged between different simulation models and simulation tools. As discussed in Section 2.1.1, the CSV-based files storing boundary conditions of the bio-mechanical bone simulation may originate from a multi-body simulation of a whole human skeleton [RKH$^+$10]. However, this multi-body simulation and the bio-mechanical simulation based on Pandas feature significant variances with respect to low-level tabular structures in the CSV-based files. In particular, the multi-body simulation outputs one file containing the force vectors of all relevant muscles, as well as one file for all adjacent bones. Pandas however requires exactly one input file for each muscle and adjacent bone. So, a workflow realizing this data exchange has to filter proper rows from the input CSV-based files and has to split their columns to multiple output CSV-based files. Furthermore, coordinate transformations are again required in a similar way as described for geometry files in Section 3.2.1 [Gat14, Kra14].

The increased complexity of data provisioning and data exchange between different simulation models is especially challenging for multi-scale simulations. They not only entail an increased heterogeneity of data formats, but also bigger and more varying data sizes as well as more sophisticated data transformations. In particular, different levels of granularity in both simulation calculations and data representations make it necessary to additionally aggregate, interpolate, or extrapolate data [Gat14]. Figure 3.2 shows the process that couples the multi-scale bio-mechanical and systems-biological simulation models depicted in Figure 2.1 [RSM14a, RWWS14]. Thereby, the figure highlights ETL processes that are necessary to exchange the data between these two models [RSM14b].

The boundary conditions of the bio-mechanical simulation, i. e., the external load on the bone, mainly depend on the way the relevant person and his or her bones move. This in turn depends on the daily routines of the person. In the

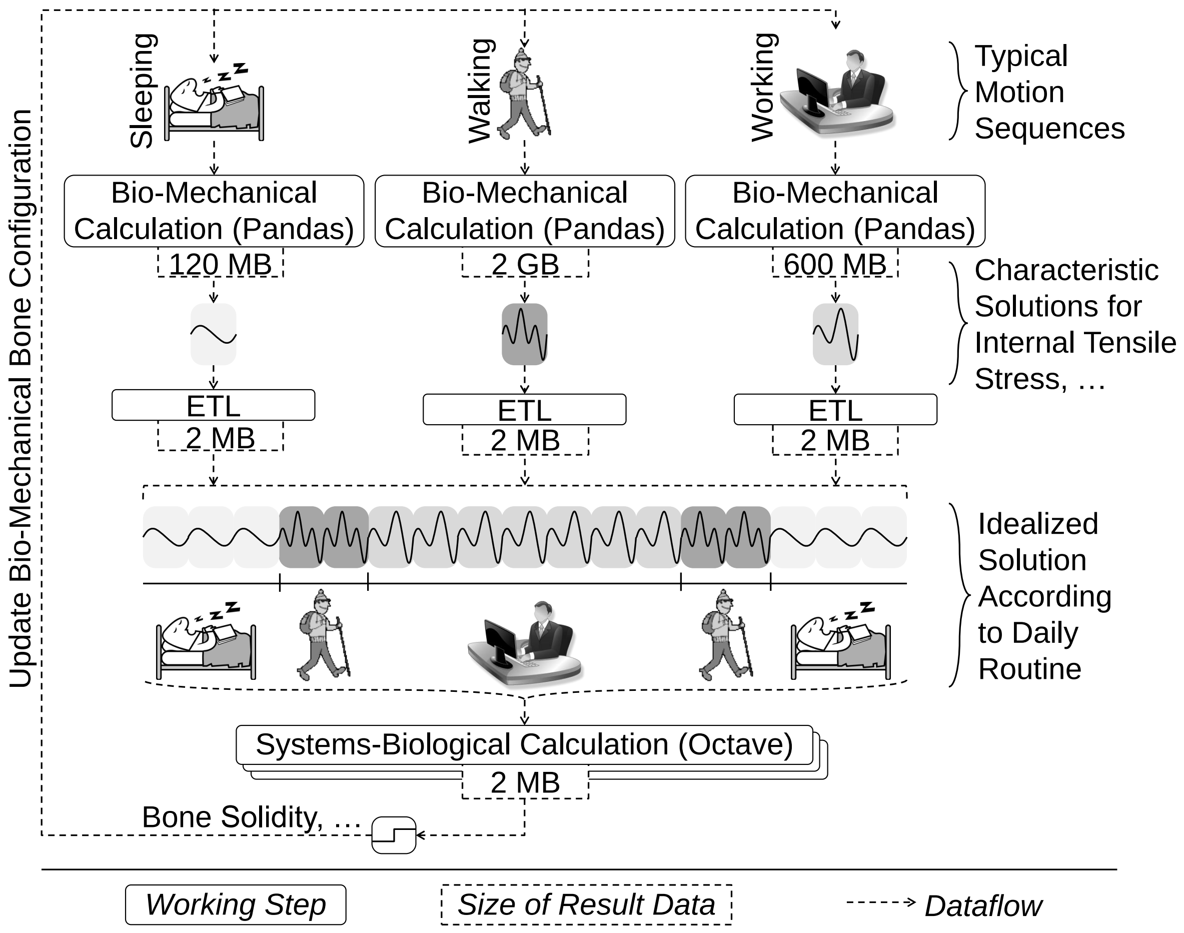

Sleeping
Walking
Working
Typical Motion Sequences
Bio-Mechanical Calculation (Pandas)
Bio-Mechanical Calculation (Pandas)
Bio-Mechanical Calculation (Pandas)
120 MB
2 GB
600 MB
Characteristic Solutions for Internal Tensile Stress, …
ETL
ETL
ETL
2 MB
2 MB
2 MB
Idealized Solution According to Daily Routine
Systems-Biological Calculation (Octave)
2 MB
Bone Solidity, …
Update Bio-Mechanical Bone Configuration
Working Step
Size of Result Data
Dataflow

Figure 3.2: Coupling process of the multi-scale simulation of structure changes in bones combining bio-mechanical and systems-biological calculations [RSM14a].

simulation, each relevant daily routine is approximated by a composition of representative motion sequences, e. g., for sleeping, walking, or working. For each motion sequence, Pandas converts the associated external load into a characteristic solution of the bio-mechanical unknown variables, e. g., the internal tensile stress. Since especially the number of numerical time steps varies between the calculations regarding individual motion sequences, the respective sizes of result data vary as well (see Figure 3.2). More details about this calculation and about the resulting simulation output are given in Sections 1.1, 2.1.1, and 3.2.1.

The systems-biological simulation is not implemented by Pandas, but by GNU Octave[3]. It gets an idealized solution as input that is composed of the characteristic solutions of Pandas according to the approximated daily routine of

[3]GNU Octave: `http://www.gnu.org/software/octave/`

the relevant person. While Pandas stores its simulation output in a SQL database, Octave expects CSV-based files for both its input and output data. The systems-biological calculation in Octave only needs a subset of the 20 mathematical unknown variables Pandas stores in its database. In addition, Octave needs the values of these variables to be aggregated among all numerical time steps, e. g., by calculating average values. Since the systems-biological calculations at individual spatial evaluation points are completely independent from each other (see Section 2.1.1), they may be massively parallelized. This makes it necessary to partition the output data of Pandas among multiple instances of Octave. The ETL processes shown in Figure 3.2 perform the corresponding format conversion, filter, aggregation, and partitioning of the data. Thereby, especially the filter and aggregation operations reduce the data size from several GBs in the database of Pandas to a few MBs in the CSV-based files of Octave.

Subsequently, each Octave instance uses the resulting CSV-based files as input and calculates its output, e. g., the precise bone solidity, until the end of one daily routine. It then stores this result in another CSV-based file having a data size of again a few MBs. The output files of all concurrently executed Octave instances are then imported into the database of Pandas. This makes the bone configuration of the bio-mechanical simulation model more precise and prepares Pandas for further calculations. The whole process is repeated for the next daily routines, i. e., until a sufficient number of days has been considered [RSM14b, RSM14a].

Figure 3.3 shows an abstract view on a workflow realizing the coupling process shown in Figure 3.2 [RS13b]. This workflow firstly accesses a repository to load a list of relevant daily routines including their motion sequences. Afterwards, it iterates over this list and sequentially executes the bio-mechanical and systems-biological simulations for each daily routine. As depicted in Figure 3.2, the bio-mechanical calculation may be executed independently for each motion sequence. So, it is parallelized among several computers. The next workflow step shown in Figure 3.3 loads a list of available Pandas computers from another repository. Thereupon, each motion sequence of the current daily routine is assigned to one of the computers that has to process the relevant motion sequence. The workflow then starts a Pandas simulation in parallel for each motion sequence.

The preparation of the data exchange and of the systems-biological simulation begins by loading a list of available Octave computers. Subsequently, the next workflow step determines how to partition the output data of Pandas among these computers. Therefore, it accesses the SQL database to determine the

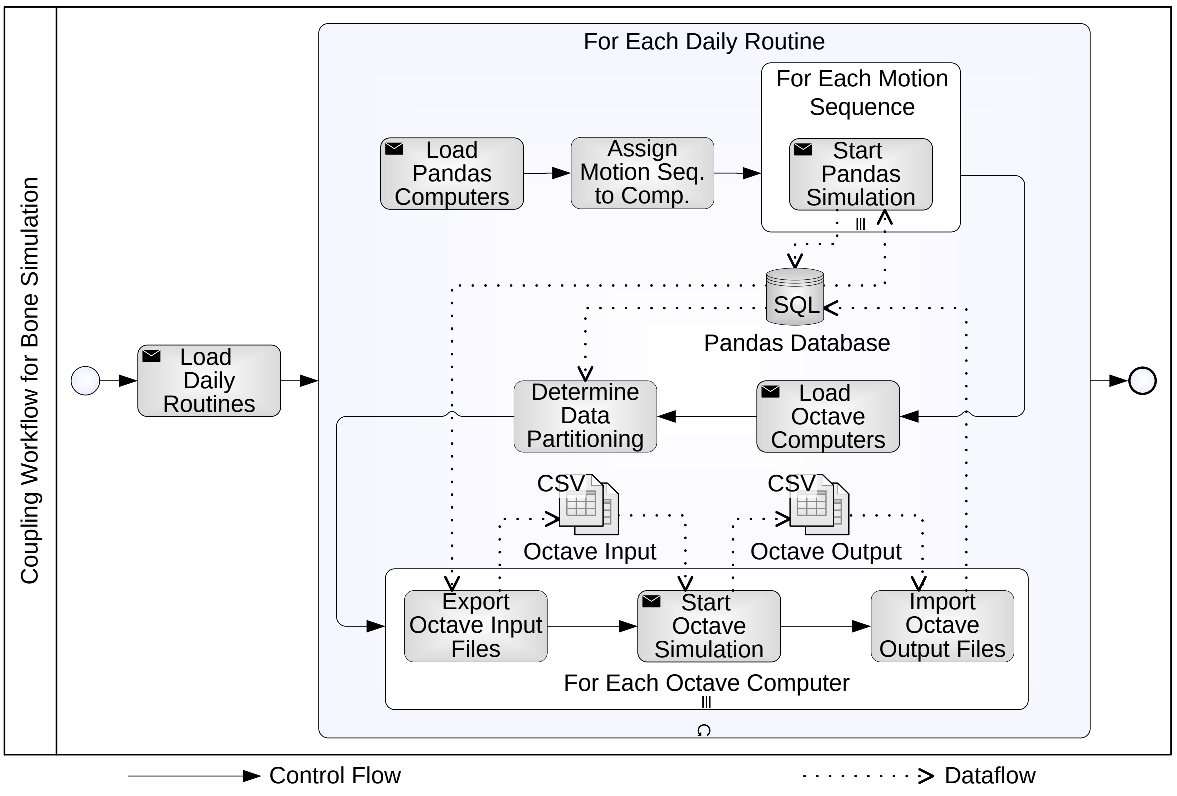


Figure 3.3: Abstract view on a workflow realizing the coupling process shown in Figure 3.2; cf. [RS13b].

number of data items Pandas has previously stored in this database, as well as the number of data items each Octave computer has to process.

The following parallel loop iterates over the list of Octave computers. The first workflow step in this loop implements the actual data exchange between Pandas and Octave, i. e., most parts of the ETL processes shown in Figure 3.2. So, it filters appropriate data from the SQL database of Pandas, performs the necessary data aggregations and partitioning, and exports the results into CSV-based files. Afterwards, the workflow starts a new instance of the systems-biological Octave simulation. After this Octave instance has finished its calculation, the last activity in the loop imports its output files back into the SQL database of Pandas. This updates the bio-mechanical bone configuration. The workflow then repeats its outermost loop until all relevant daily routines have been processed.

A concrete workflow implementing the abstract workflow depicted in Figure 3.3 consists of more than 15 workflow tasks [RSM14b]. To design such a workflow, scientists need to specify a multiplicity of complex low-level details of data management and data exchange. In particular, around half of the workflow tasks require the specification of several shell commands or even of sophisticated SQL, XPath, and XQuery statements. The most complex statement is the SQL statement embedded in the workflow task *Export Octave Input Files* shown in Figure 3.3. It is a `SELECT` query including three sub-queries to filter the proper mathematical variables stored in the database of Pandas (see also Listing 5.1 on page 155) [Ges14]. Each of these sub-queries specifies several selection predicates, as well as an average function and a `GROUP BY` clause to carry out the necessary data aggregations. The individual results of all sub-queries are combined via corresponding join predicates. Finally, the data partitioning or data split among available Octave instances is realized via an `ORDER BY` clause and associated `LIMIT` and `OFFSET` clauses.

Scientists are often not familiar with languages such as SQL, XPath, or XQuery. In particular, they usually do not have the necessary skills to specify the mentioned low-level operations that filter, aggregate, group, join, and partition data. All this significantly increases the complexity of data transformations to be designed and executed in simulation workflows for exchanging data between different simulation models [Gat14, RSM14b]. It constitutes a common challenge for multi-scale simulations that are coupled across different scientific domains [RSM14a]. For instance, this also concerns the multi-scale simulation of catalysis reactions described by Rommel et al. [RK11]. Here, coupling the molecular-dynamic, molecular-mechanical, and quantum-mechanical calculations likewise requires the specification of low-level operations to perform the necessary data exchange between these different calculations [Mül10, Pie12].

As a consequence, the increased complexity of data exchange between several simulation models often hinders scientists to design corresponding workflows at all. This was actually the case for the workflow shown in Figure 3.3, which could only be designed in collaboration between the scientists conducting this simulation and several experts in workflow and data engineering [Mül10, Dor11, Pie12, RS13b, RSM14b]. Note that such an interdisciplinary collaboration between different persons having different skills is getting more and more indispensable to couple several simulation models [RSM14a]. It is especially important to conduct multi-scale simulations working with different levels of granularity in both numerical calculations and data representations. So, it is crucial to facilitate novel and sophisticated simulation applications that allow for producing precise simulation results and for making simulations more realistic [GZC14].

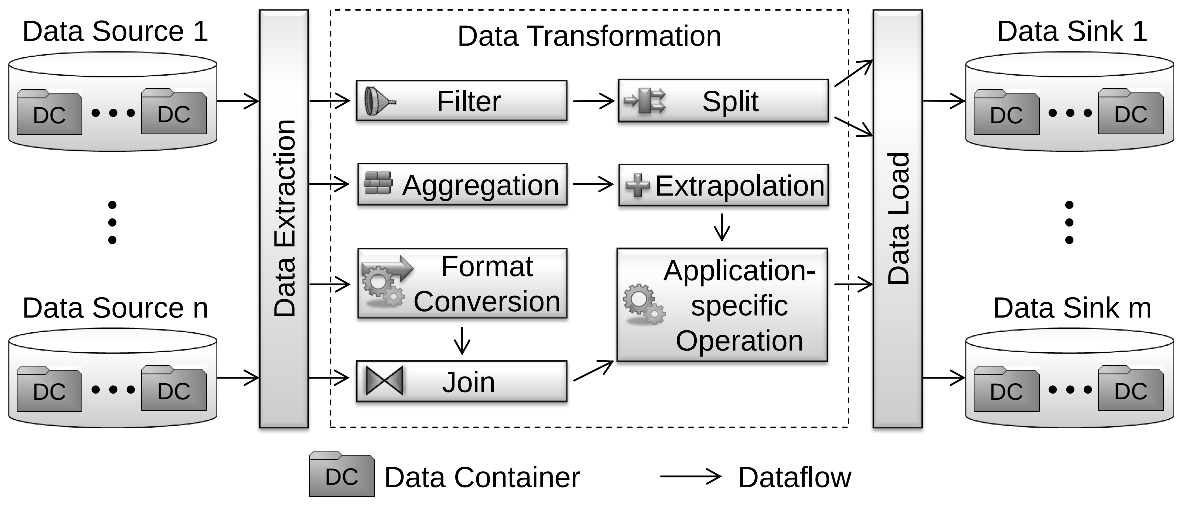


Figure 3.4: General Data Transfer and Transformation Pattern; cf. [RS13b].

3.3 Basic Data Management Patterns

The real-world simulation examples or related applications listed in Section 3.1 cover a wide base of use cases spanning various scientific domains. Hence, these examples and the resulting characteristics of simulation data and of their processing discussed in Section 3.2 are well-suited to derive basic data management patterns describing common operational aspects in simulation workflows. The most frequent data management patterns observed in these simulation examples may be classified into two different groups: *Data Transfer and Transformation Patterns* and *Data Iteration Patterns* [RS13b]. The following two subsections circumscribe the definitions and examples of these two groups of patterns.

3.3.1 Data Transfer and Transformation Patterns

As depicted in Figure 3.4, *Data Transfer and Transformation Patterns* describe a process to transfer data between several data resources, i. e., from one or more data sources to one or more data sinks [RS13b]. For both the data sources and the data sinks, this process may access an arbitrary number of data containers e. g., tables in a database system or files in a file system. In addition, the patterns typically cover ETL operations, i. e., 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. These ETL operations cope with the intrinsic heterogeneity of involved data resources and data formats.

Such Data Transfer and Transformation Patterns occur frequently in the workflows of any simulation example, e. g., they especially occur in any example listed in Section 3.1. This holds both for workflows providing input data for single simulation models illustrated in Section 3.2.1 and for workflows exchanging data between different simulation models as discussed in Section 3.2.2. For instance, the data provisioning tasks of the bio-mechanical simulation workflow (blue workflow steps shown in Figures 1.1 and 3.1) each constitute such a pattern. Other, even more complex examples are the ETL processes shown in Figure 3.2 that exchange data between the bio-mechanical and systems-biological simulations.

In workflows providing data for single simulation models (see Section 3.2.1), frequent ETL operations are *filtering* a data (sub-)set and complex *format conversions* [Mül10, Pie12]. The format conversions thereby have to cope with various kinds of data formats, e. g., different kinds of database systems, CSV-based files, or even proprietary text or binary files [RS14]. Some simulation applications also require more or less sophisticated *application-specific operations*. Most common examples are coordinate transformations in geometry descriptions or boundary conditions [Gat14], which may also be required for providing the geometry data of the bio-mechanical simulation model.

Workflows exchanging data between different simulation models (see Section 3.2.2) extend the set of necessary ETL operations by even more complex ones. For instance, the ETL processes of the coupled bone simulation shown in Figure 3.2 not only carry out data filters and format conversions. In addition, they also perform *aggregations* among several data values, i. e., they calculate average values of the bio-mechanical unknown variables among all numerical time steps [Mül10, Ges14]. Furthermore, they *split* one data set into several ones in order to partition the output data of Pandas among several Octave instances [RSM14b, RSM14a]. Another example is a *join* operation that combines the results of sub-queries of the SQL statement embedded in the workflow task *Export Octave Input Files* shown in Figure 3.3. Finally, multi-scale simulations commonly make it necessary to *extrapolate* data [Gat14].

The general Data Transfer and Transformation Pattern shown in Figure 3.4 describes a data provisioning process from n to m data containers with $n, m \geq 1$. Individual variants of this pattern may also restrict the numbers of input and output data containers. The first variant, the Container-to-Container Pattern, accesses exactly one data container in one data source and one data container in one data sink. Examples of this pattern variant are the workflow tasks *Extract*

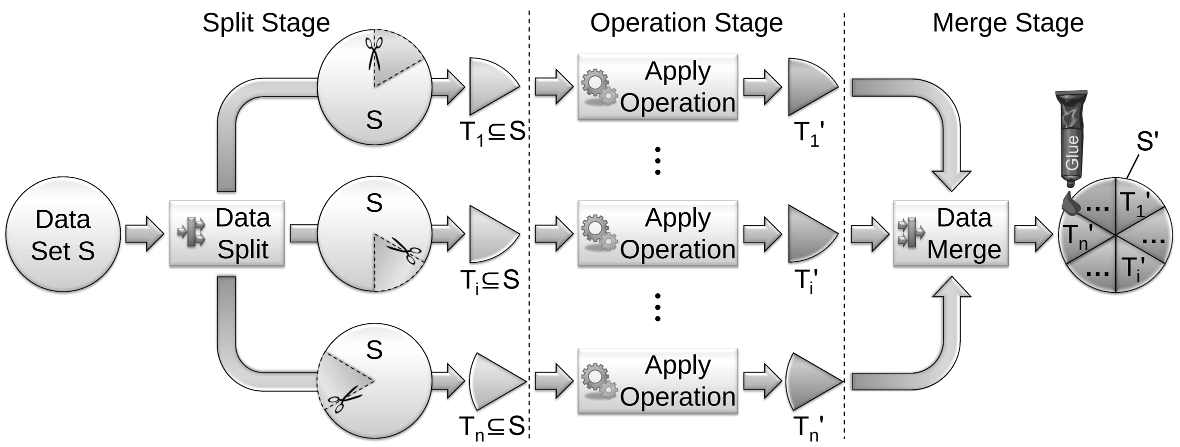


Figure 3.5: Parallel Data Iteration Pattern; cf. [RS13b].

Material Parameters and *Select FEM Parameters* shown in Figure 3.1. A Data Split Pattern extracts a data set from one data container and splits this data set into an arbitrary number of target containers. In the opposite way round, a Data Merge Pattern describes a process to merge data of multiple data containers into exactly one container. Examples of the two last-mentioned pattern variants are part of the coupling workflow depicted in Figure 3.3. This concerns the workflow steps *Export Octave Input Files* and *Import Octave Output Files* that partition and subsequently merge the data in the database of Pandas. Most of the ETL operations depicted in Figure 3.4 are applicable to each of these three pattern variants. Restrictions may occur, e. g., for the split operation, as this operation is only feasible for patterns considering multiple target data containers.

3.3.2 Data Iteration Patterns

The principle of the *Data Iteration Patterns* is the iteration over a data set S and the execution of an operation, where this data set or subsets of it serve as input [RS13b]. This iteration and thus the pattern itself may occur in two variants: (1) a parallel one (Figure 3.5) and (2) a sequential one (Figure 3.6).

The *Parallel Data Iteration Pattern* comprises a split stage, an operation stage, and a merge stage. Its focus is on the parallelization of an operation among multiple resources. The *split stage* encompasses tasks to partition the set S into n subsets $T_i \subseteq S$ and to distribute these subsets among the resources. This split stage may also be represented by a Data Split Pattern introduced in Section 3.3.1. In the *operation stage*, the subsets T_i serve as input to apply the

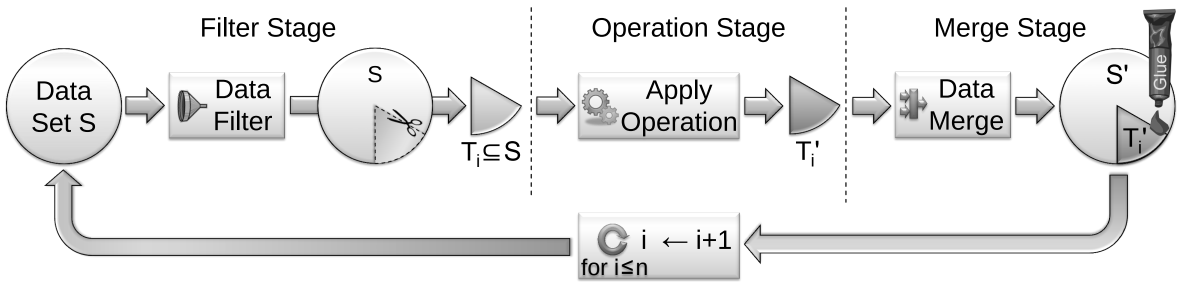


Figure 3.6: Sequential Data Iteration Pattern.

parallel operations, which each delivers the corresponding $T_i{}'$ as result. The *merge stage* then comprises tasks, e. g., represented by a Data Merge Pattern, to integrate all changed subsets $T_1{}'$ to $T_n{}'$ into the overall output set S'.

Especially multi-scale simulations usually entail extremely time-consuming numerical calculations for at least some of the coupled simulation models [Gat14, GZC14, UGM14]. Hence, they often require their iterative and/or parallel computation and thus a partitioning of input data among several computers. The Parallel Data Iteration Pattern represents this common scenario of parallel calculations and of a data partitioning. This pattern for instance finds application in the coupling workflow depicted in Figure 3.3 [RSM14a]. Thereby, the data in the database of Pandas corresponds to the data sets S and S'. The Octave simulation constitutes the operation, while the CSV-based input and output files represent the subsets T_i and T_i'. Pietranek shows that the pattern may even be used to describe parts of the multi-scale simulation of catalysis reactions listed in Section 3.1 [Pie12]. Furthermore, the workflow realizing the crack propagation simulation studied by Franzelin et al. bears much resemblance to the coupling workflow depicted in Figure 3.3 regarding the data partitioning [FDP15]. So, it is also a good candidate to be expressed via the Parallel Data Iteration Pattern.

The *Sequential Data Iteration Pattern* neither considers a parallelization of an operation nor the partitioning of the data set S. Instead, all iteration cycles are executed one after another, and the split stage changes to a *filter stage*. This filter stage selects a subset $T_i \subseteq S$, which is again changed to $T_i{}'$ in the operation stage. Subsequently, $T_i{}'$ is again merged into the overall result S'. This process is repeated as long as the total number of iterations reaches a certain count n.

This kind of pattern especially occurs in simulations, where individual iterative calculations depend on each other and are thus not suited for massive parallelization [UGM14]. For instance, this concerns the outermost loop of the coupling workflow shown in Figure 3.3. This loop sequentially iterates over the previously loaded list of daily routines and executes the embedded bio-mechanical and systems-biological calculations for one day after another [RSM14a]. The sequential variant of this pattern may also occur in the simulation of catalysis reactions listed in Section 3.1. Pietranek discusses how it may serve as complement or even as alternative to the Parallel Data Iteration Pattern described above [Pie12]. A previous author publication discusses how the Sequential Data Iteration Pattern may be applied to the model reduction example proposed by Fehr et al. [FE11, RSM14a]. Finally, such a pattern may express nearly the whole protein modeling workflow illustrated by Wagner [Wag10]. This workflow sequentially iterates over a list of protein sequences and applies a pattern matching operation to each individual sequence [RSM11].

3.4 Summary and Future Work

Computer-based simulations feature a highly heterogeneous data environment, especially with respect to various kinds of proprietary data formats as well as big and varying data sizes. Hence, simulation workflows have to carry out many complex data provisioning tasks to overcome this intrinsic data heterogeneity. This already holds for workflows providing data for single simulation models, as illustrated in Section 3.2.1. The complexity of providing data is even increased in case data needs to be exchanged between several simulation models that are coupled across different scientific domains (see Section 3.2.2). This is particularly true for multi-scale simulations that involve various levels of granularity in both numerical calculations and data representations. Due to this high complexity of data provisioning and especially of data exchange, scientist often are not able to design corresponding simulation workflows at all. This calls for an abstraction support that is especially tailored to the needs of scientists.

In spite of this heterogeneity and complexity, it is nevertheless possible to derive basic data management patterns that describe common operational aspects in simulation workflows. The resulting patterns illustrated in Section 3.3 are used in the following chapters to discuss detailed functional requirements for individual proposed approaches. Furthermore, they are used as basis for the pattern-based abstraction support for simulation workflow design introduced in Chapter 6. Thereby, this set of patterns is even extended by more abstract ones that make simulation workflow design especially tailor-made for scientists.

Chapter 4

Data Provisioning Techniques for Simulation Workflows

Most of current workflow systems or other simulation tools provide some means to realize data access and data provisioning in workflows. The complex and heterogeneous data environment for computer-based simulations highlighted in Chapter 3 however not only leads to a high complexity of data provisioning tasks in simulation workflows. It also tends to increase the diversity of data provisioning techniques that are offered by workflow systems or simulation tools. As discussed in Section 1.2.1, these systems or tools use specialized and thus heterogeneous solutions, e. g., application-specific services or custom workflow extension activities. Individual systems even offer multiple of these specialized solutions in order to support as many applications and scientific domains as possible [CBL11].

Scientists designing simulation workflows have to choose proper data provisioning techniques from this large set of available techniques in order to realize the various data provisioning tasks in their workflows. The diversity and complexity of available techniques however complicates the scientists' decision on appropriate techniques. This raises a new problem to scientists that induces them not to leverage existing techniques at all [CBL11]. Instead, they still implement the necessary data provisioning on their own, leading again to a significantly high effort to be spent on workflow design.

This thesis addresses the mentioned issues associated with the diversity and complexity of data provisioning techniques by a systematic comparison of available techniques. The major goal of this comparison is to derive guidelines that help scientists to choose appropriate data provisioning techniques for given workflow tasks (see Section 1.3.1). This includes the following detailed contributions:

- The high diversity of available low-level data provisioning techniques likewise complicates their systematic comparison. Therefore, Section 4.1 discusses how to conquer this diversity by classifying existing data provisioning techniques into three generic concepts that are representative for the techniques offered by a multitude of workflow systems. Furthermore, the chosen classification scheme and the resulting concepts are assessed regarding their suitability for a systematic comparison of underlying techniques.
- Section 4.2 deals with the actual comparison of available data provisioning techniques. It first discusses comparison criteria that are especially relevant to evaluate the main differences of the previously classified data provisioning concepts or their underlying techniques. Functional criteria are related to the basic data management patterns formalized in Section 3.3. The analysis regarding non-functional criteria is based on the challenges discussed in Section 1.2. The resulting criteria are then used to systematically compare the data provisioning concepts.
- The comparison results are used in Section 4.3 to derive guidelines that assist scientists in choosing appropriate data provisioning techniques for their workflows. Furthermore, this section shows how scientists may effectively apply these guidelines. It discusses the results of evaluating the guidelines, which has been based on applying them to the real-world simulation examples listed in Section 3.1. Thereby, the main focus is on how the guidelines facilitate the design of simulation workflows. In particular, they enable scientists to focus on the most appropriate data provisioning concept, instead of being overstrained by a multitude of low-level techniques.
- As another contribution, Section 4.4 uses both the comparison results and the guidelines to derive essential features simulation workflow systems have to offer. In particular, these features are necessary to effectively apply the guidelines in practice. By matching these mandatory features with those offered by available workflow systems, missing features of these systems are identified. This thesis additionally introduces an extended simulation workflow system that offers all these missing features in a holistic way.

Finally, Section 4.5 summarizes the major aspects and lists possible future work.

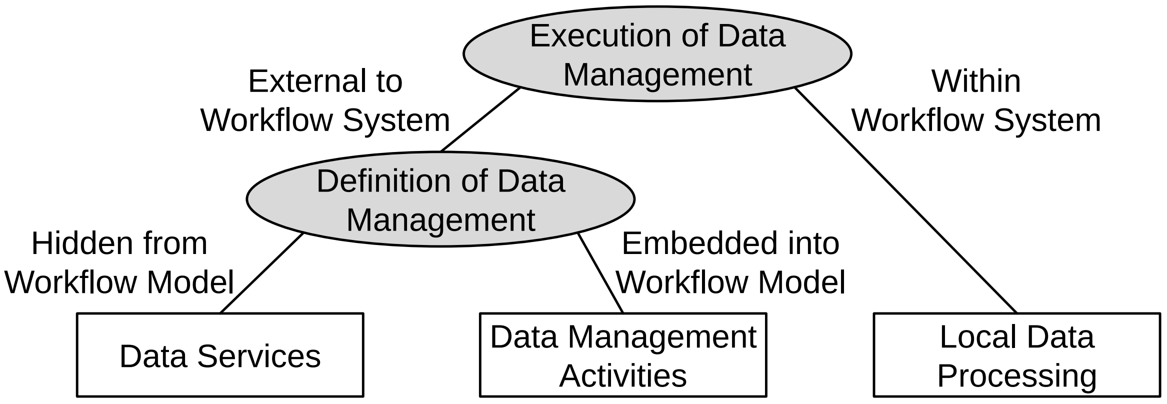

Execution of Data Management
External to Workflow System
Within Workflow System
Definition of Data Management
Hidden from Workflow Model
Embedded into Workflow Model
Data Services
Data Management Activities
Local Data Processing

Figure 4.1: Decision tree to classify data provisioning techniques into representative and comparable concepts of such techniques. Decisions are shown in circles with grey background, whereas the resulting concepts are depicted as rectangles with white background.

4.1 Classification of Data Provisioning Techniques

As discussed above, a systematic comparison of all available low-level data provisioning techniques is not reasonably practicable due to their high diversity. For that purpose, this section discusses how to classify existing data provisioning techniques into a smaller set of representative concepts of such techniques. This then facilitates the systematic comparison of the concepts, and it allows for drawing conclusions about the corresponding low-level techniques. Section 4.1.1 presents the classification scheme proposed in this thesis, as well as the resulting concepts of data provisioning techniques. Furthermore, it illustrates how these concepts are supported by existing workflow systems. Afterwards, the classification scheme and the resulting concepts are assessed in Section 4.1.2, especially why they are suited for systematically comparing the underlying low-level techniques.

4.1.1 Concepts of Data Provisioning Techniques

In the course of working on this thesis, the classification scheme represented as decision tree in Figure 4.1 has been derived. This has been based on an analysis of several workflow languages and of the data provisioning techniques supported by a multitude of workflow systems. This classification scheme characterizes data provisioning techniques along two decisions, which are related to the two general aspects of workflow management: (1) *execution* and (2) *definition or modeling* of (the data management in) workflows [LR00, Wes12]. Thereby, it identifies three relevant concepts of data provisioning techniques. These concepts are depicted in Figure 4.2 and illustrated in the following subsections.

4.1.1.1 Data Services

The first concept, called *data services*, originates from settings of business applications, where the Service-oriented Architecture (SOA) and the service orchestration facilities of workflows realize business processes [WCL$^+$05, Wes12]. On this note, services may encapsulate access to one or more data resources and provide operations on this data, e. g., for data extraction or data transformation. In other words, such data services hide data resources and data processing from the workflow that invokes the services. Data management operations are not *executed* within the workflow system, but *by external services or data resources* (see Figure 4.1). Likewise, the concrete semantics of a data management operation, i. e., *its algorithmic and/or detailed declarative definition*, is not directly modeled in the workflow. Instead, it is usually part of an external service implementation and thus *hidden from the workflow model.*

As discussed in Section 2.3.3, the Web Services technology is a common solution to service calls from workflows, especially in the workflow language BPEL [WCL$^+$05, OAS07]. The REST protocol (Representational State Transfer) is a complementary approach to stateful (data) services [PZL08]. Other examples that are quite common in the workflow or service domains are specific file transfer services, e. g., based on FTP solutions or on the Java Secure Channel API[1]. Scientific applications – including simulation workflows – recently adopted the SOA paradigm and the Web Services technology as well [TDG07, GSK$^+$11]. An example solution to service-based data management in such applications is the Open Grid Services Architecture - Data Access and Integration framework (OGSA-DAI)[2]. Most of today's scientific workflow systems and their proprietary workflow languages offer some kind of actors or activities to invoke services and may thus use services for data management. For instance, this includes the workflow system proposed by Görlach et al. [GSK$^+$11], as well as Kepler, Taverna, Triana, and Microsoft Trident [LAB$^+$06, OGA$^+$06, CGH$^+$06, BJA$^+$08].

[1]Java Secure Channel API: `http://www.jcraft.com/jsch/`
[2]OGSA-DAI: `http://www.ogsadai.org.uk/index.php`

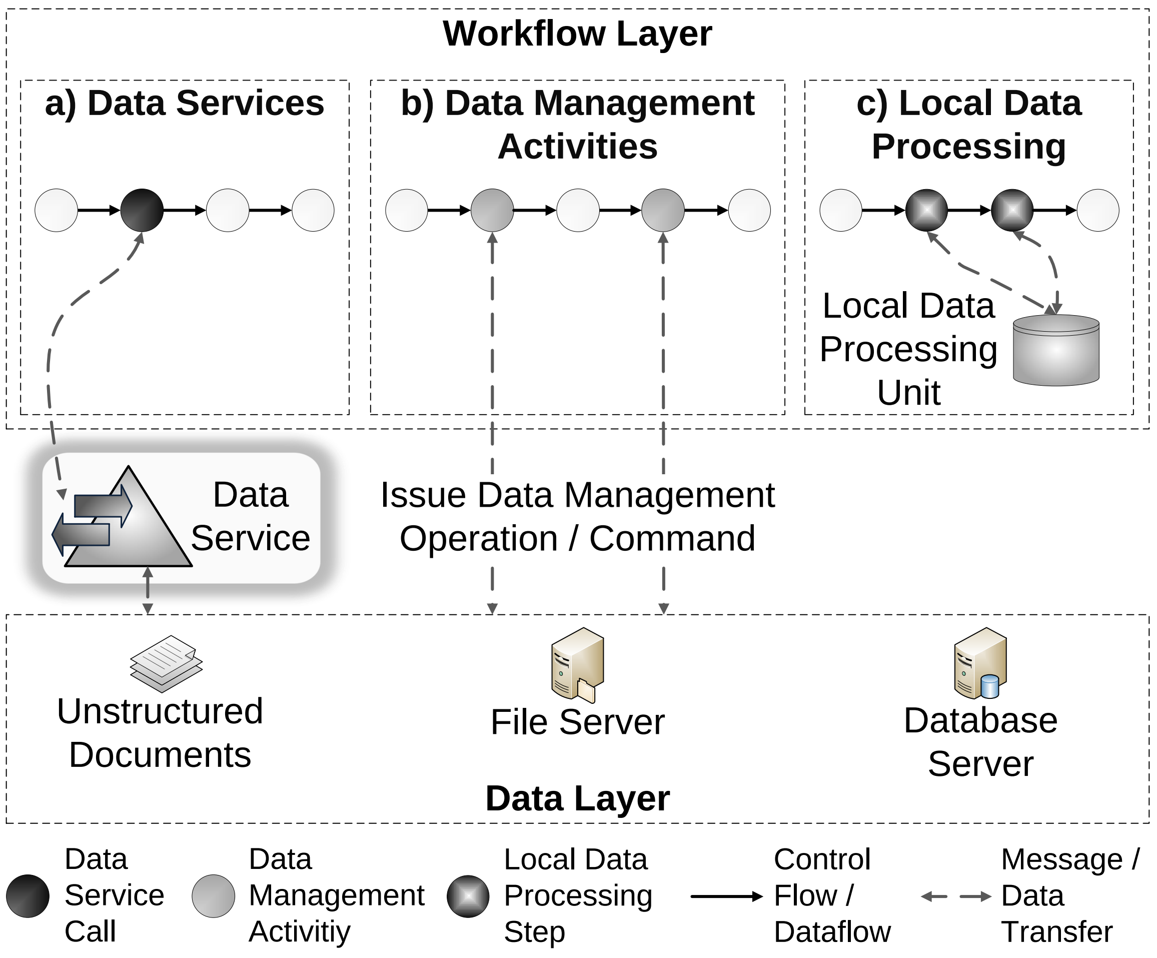


Figure 4.2: Concepts of data provisioning techniques in simulation workflows.

4.1.1.2 Data Management Activities

The second concept is an approach using special *data management activities* in workflows. Typically, such an activity is associated with an external data resource and embeds a command in the command language of this resource, e. g., a SQL statement or a shell command [VSRM08, RRS+11]. During its execution, the activity seamlessly issues this command against the associated resource that then carries out the corresponding data management operation. So, data management operations are *executed* by external data resources, i. e., *external to the workflow system* as in case of data services. Nevertheless, the above-mentioned command corresponds to a declarative definition of an operation, and this *declarative definition* is thus directly *embedded into an activity of the workflow model*. In some cases, data management activities may even embed an algorithmic definition of an operation, e. g., Node-RED allows for dynamically deploying and executing custom code snippets on external resources[3].

The usage of data management activities mainly goes back to business workflow systems, e. g., the workflow products of IBM, Microsoft, and Oracle allow for integrating SQL statements into BPEL workflows [VSRM08]. Nevertheless, the same also holds for the scientific workflow systems Kepler and Microsoft Trident [LAB+06, BJA+08]. Kepler even offers proprietary actors to access file systems and sensor networks [BAJ+10]. The data management activities proposed in Chapter 5 of this thesis offer a generic approach to embed any data management command for any kind of data resource directly within workflows [RRS+11].

4.1.1.3 Local Data Processing

In contrast to the other concepts, the last concept reflects a local *execution of data management operations within workflow systems*. This concept is therefore called *local data processing*. Here, data and workflow processing are integrated together. Data is stored in the process context of the relevant workflow, e. g., in workflow variables in case of workflow languages such as BPEL. Workflow tasks may embed or otherwise specify data management operations that are executed locally in the workflow environment, e. g., based on variable assignments. The workflow environment may integrate a database system or other kinds of local data processing units that manage the data as well as its processing by the workflow [RSM11]. The analysis of workflow languages and workflow systems performed while working on this thesis revealed that any available workflow system offers specific opportunities for this local data processing.

4.1.2 Assessment of the Proposed Classification Scheme

In summary, each of today's workflow systems supports different solutions of a sub-set of the data provisioning concepts classified above. However, all these systems and related literature leave the decision on appropriate data provisioning techniques to the workflow developers, i. e., to scientists [CBL11]. The diversity and complexity of the techniques offered by different systems may then overburden scientists and induce them not to use the techniques at all. The systematic classification of data provisioning techniques into representative concepts, as

[3]Node-RED: `http://nodered.org/`

discussed in this section, is a first step towards conquering this diversity and complexity. Scientists may use the classification scheme depicted in Figure 4.1 to correspondingly classify the techniques that are supported by relevant workflow systems. This then already helps them to focus on the choice of appropriate concepts, instead of being overstrained by a multitude of low-level techniques. In order to further assist scientists in this choice and to derive a set of corresponding guidelines, the resulting concepts nevertheless need to be compared to each other. The classification scheme and the concepts proposed in Section 4.1.1 are well-suited for achieving this goal and for drawing conclusions about the underlying low-level techniques, because they show the following properties:

- Firstly, the **number of concepts should be as little as possible**. This is mandatory to effectively conquer the diversity of existing data provisioning techniques and thus to facilitate their systematic comparison. The classification scheme depicted in Figure 4.1 shows this property, as it covers only two decisions with two options each. So, the theoretically maximum number of resulting concepts is four. In practice, the classification scheme even results in only three concepts as shown in Figure 4.1.
- A reasonable comparison requires the **concepts to be explicitly distinguishable** from each other. The classification scheme depicted in Figure 4.1 shows this property, since the two options of each decision are mutually exclusive. In practice, any data management operation is either executed within the workflow system or externally to it. Likewise, the algorithmic or declarative definition of an operation is either embedded into the workflow model or it is not part of this model and thus hidden from it. Moreover, if one of the decisions was left out, some concepts would coincide although being completely different. In case the decision where to execute a data management operation was left out, it would not be possible to distinguish between the concept of local data processing and the other two concepts anymore. Neglecting the decision where to define an operation would similarly lead to the coincidence of data services and data management activities.
- The resulting **concepts should be representative**, i. e., they should summarize all relevant low-level data provisioning techniques. The proposed classification scheme shows this property as a direct implication of the argumentation regarding the second property: For each of the decisions, either the one or the other option holds (exclusive or). So, the two options of each decision are not only mutually exclusive, but also collectively

exhaustive. This means that at least one of the options holds for any kind of existing data provisioning technique. Furthermore, the two decisions of the classification scheme cover the two general and thus all major aspects of workflow management, i. e., execution and modeling [LR00, Wes12]. These two aspects and thus the chosen classification scheme and the resulting concepts remain valid even in case underlying technologies, e. g., data resources or workflow systems, change over time.

The development of the classification scheme shown in Figure 4.1 has been underpinned by inspecting alternative ways for classifying data provisioning techniques. However, no other scheme has been found that shows all three essential properties listed above. For instance, questions about data quality, e. g., the accuracy or timeliness of data, would require introducing specific thresholds for these time-variant data characteristics to make resulting concepts explicitly distinguishable. Proper thresholds for data quality are however application-specific, leading to data provisioning concepts that are not representative, i. e., not valid for every application or domain [Sad13]. Nevertheless note that future work may use the classification scheme shown in Figure 4.1 as basis and elaborate on more fine-grained decisions and corresponding options. This would also lead to more fine-grained concepts of data provisioning techniques. It would enable the comparison of these concepts, which may later result in likewise more detailed guidelines helping scientists to choose proper techniques. However and as discussed above, a systematic comparison is only possible in case the diversity of available data provisioning techniques has previously been conquered by classifying all low-level techniques into a smaller set of representative concepts. So, the classification scheme shown in Figure 4.1 can be seen as a good starting point and as a framework for future, more detailed classifications and comparisons.

4.2 Comparison of Data Provisioning Techniques

This section summarizes the main results of comparing the data provisioning concepts illustrated above. Section 4.2.1 discusses comparison criteria that are especially relevant to achieve the major goal of this comparison, i. e., deriving guidelines for choosing appropriate data provisioning techniques for a workflow. Sections 4.2.2 and 4.2.3 then present the corresponding comparison results.

4.2.1 Discussion of Relevant Comparison Criteria

A systematic comparison has to be based on a set of comparison criteria that reflect the most relevant requirements for the considered scope of application. The most relevant requirements for data provisioning in simulation workflows are related to (1) the challenges discussed in Section 1.2 and to (2) the common data characteristics and operational aspects illustrated in Chapter 3. A corresponding requirements analysis performed while working on this thesis has led to a multiplicity of comparison criteria. The data provisioning concepts illustrated in Section 4.1.1 have then been intensively evaluated with respect to these criteria. The illustrations in the following sections however focus on only a sub-set of the criteria for the sake of clarity. Thereby, especially those criteria are neglected that did not lead to significant insights with respect to the major goal of the comparison, i. e., using it to derive guidelines for choosing appropriate data provisioning techniques (see Section 4.3). For instance, some of these neglected criteria are related to the scientists' requirement of being able to make ad-hoc changes to workflow models at runtime [SK10]. Regardless of the used data provisioning concept, such dynamic changes to workflow models are generally possible thanks to approaches to concurrent workflow evolution and ad-hoc workflow instance migration [SK11, SK13]. The following list summarizes the major classes of comparison criteria that finally led to new and significant insights with respect to comparing the data provisioning concepts:

- **Generic support of common data management patterns.** This class of comparison criteria deals with functional issues. These functional issues cover the question to what extent the data provisioning concepts *support common data management patterns* that frequently occur in simulation workflows. Thereby, a special focus is on patterns to overcome the intrinsic heterogeneity of data resources and data formats, e. g., the patterns discussed in Section 3.3. This is especially important for multi-scale simulations that are coupled across different scientific domains, because such coupled simulations show the highest degree of data heterogeneity. To enable a seamless simulation coupling, the data provisioning concepts have to support common patterns in a sufficiently *generic* way, i. e., they have to cope with all different data resources and data formats used in the respective domains. Section 4.2.2 summarizes detailed comparison criteria and associated comparison results with respect to these functional issues.

- **Non-functional issues:** Section 4.2.3 correspondingly covers major comparison results with respect to relevant non-functional issues. Associated comparison criteria that led to new and significant insights may be subdivided into the following two aspects:
  - **Information on data management:** As depicted in Table 3.1, the total size of the data involved in particular simulation runs ranges from a few 100 KBs to multiple TBs. This may even include a dynamically changing data volume, in particular for simulations coupled across different scientific domains (see Section 3.2.2). The amounts of these data influence the execution times of simulation workflows. These issues call for suitable optimization techniques that increase the efficiency of data processing in such workflows [OdOV+11, Vhr11]. To make proper optimization decisions, an optimizer component *needs enough information on the data management of a workflow model.* For instance, optimization techniques based on workflow re-structuring require detailed and combined information about control flow and dataflow dependencies, as well as about data sizes or cardinalities [BHW+07, VSS+07].

    Another relevant aspect is to ensure the reproducibility of a simulation and of its outcome [HTT09]. This has led to many technologies to collect and subsequently query information about data provenance or data quality [ABJF06, DF08, FKSS08, CVDK+12, Sad13, RBKK14]. Here, it is crucial to properly reconstruct the correlation between (1) the collected and queried provenance or quality information and (2) the affected parts of a workflow model [KSB+10, RSM14a, MBBL15]. This again calls for enough and sufficiently accurate information on the data management of the relevant workflow model.
  - **Abstraction support:** Developers of simulation workflows, i. e., scientists, have much knowledge in their simulation domain, but rather limited skills regarding workflow or data management. The heterogeneous data environment and the resulting high complexity of data provisioning in simulation workflows, as illustrated in Section 3.2, may thus overwhelm scientists during workflow design [RLSR+06, RS14, RSM14b]. Hence and as discussed in Section 1.2.2, a further essential non-functional issue is whether data provisioning concepts or their underlying techniques offer a *suitable abstraction support* for this complex data provisioning in simulation workflows.

4.2.2 Comparison Regarding Support of Data Management Patterns

Russel et al. and van der Aalst et al. discuss common workflow patterns for control flow, data, resource, and exception handling [vdAtHKB03, RtHvdAM06, RtHEvdA05a, RtHEvdA05b, RvdAtH06]. These patterns represent basic workflow design primitives that workflow languages and workflow systems in the business process domain should support. For instance, some of the data patterns discussed by Russel et al. refer to the question whether workflow tasks or workflow instances may exchange data by value or by reference [RtHEvdA05a]. Shiroor et al. and Migliorini et al. discuss whether and how scientific workflow systems support these common workflow patterns as well [SSH$^+$10, MGLRtH11]. Moreover, Yildiz et al. and Migliorini et al. introduce a small set of further patterns that scientific workflow systems should additionally support [YGN09, MGLRtH11]. These further patterns mainly consider low-level dataflow dependencies between workflow tasks, including their interplay with basic control flow structures. Examples concern the number of data tokens consumed from input data channels or forwarded to output data channels of workflow tasks. Pautasso et al. classify patterns covering different kinds of parallelism that should be expressible in grid workflows, e. g., different kinds of data and pipeline parallelism [PA06]. All these patterns are well-suited to evaluate and compare the implementation details of or the basic features offered by different workflow languages and workflow systems. However, none of them explicitly cover operational aspects that are essential for data provisioning in simulation workflows, i. e., to overcome the intrinsic heterogeneity of data resources and data formats. Hence, they are not tailored to directly compare corresponding data provisioning techniques or concepts.

The data management patterns discussed in Section 3.3 explicitly cover the most essential operational aspects of data provisioning in simulation workflows, e. g., in terms of ETL operations. Furthermore and in contrast to all other workflow patterns, these data management patterns are situated at a similar abstraction level as the data provisioning concepts classified in Section 4.1.1. Due to these reasons, the patterns discussed in Section 3.3 are used in this thesis to compare the data provisioning concepts. The following subsections deal with the two main groups of these patterns, i. e., Data Transfer and Transformation Patterns and Data Iteration Patterns. The discussion is split into four comparison criteria. These criteria and their evaluation results are summarized in Table 4.1.

Table 4.1: Support of common data management patterns. The filling degrees of the pie charts depict how much a data provisioning concept supports the respective criteria.

Criteria	**Data services**	**Data management activities**	**Local data processing**
Data Resources Support Control over selection of data resources	◑	●	◔
Data Operation Support Range of supported data management operations	◑	◕	◔
Data Transfer Support Data transfer independent from other concepts	◕	◕	◔
Data Iteration Support Data iteration independent from other concepts	◕	◑	◔

4.2.2.1 Support of Data Transfer and Transformation Patterns

Data Transfer and Transformation Patterns can logically be subdivided into data access parts, data transformation parts, and data transfer parts. This reflects the fact that data provisioning techniques (1) have to be able to access multiple kinds of data resources, (2) have to support a wide range of data management or data transformation operations, and (3) have to facilitate corresponding data transfers between resources. These issues are especially important for coupling simulations of different scientific domains in a generic way. They correspond to the first three criteria presented in Table 4.1.

The focus of criterion **Data Resources Support** is to what extent workflow developers may control the selection of data resources their workflows shall access. *Data services* typically provide access to only a fixed set or fixed types of data resources. This may result in no or a limited control over the selection of resources. In contrast, *data management activities* offer full control over this selection, e. g., via reference variables that may be associated with any data resource of any type [VSRM08, RRS+11]. In case of *local data processing*, often only access to local storage in workflow systems is possible. Some workflow systems may also integrate external data resources. However, the set or types of such resources are typically even more restricted than in case of data services. For instance, the workflow engine Apache ODE is restricted to tuple-oriented retrievals and manipulations of data from relational database systems[4].

Supported options for specifying data management operations, in particular ETL operations, are in the focus of criterion **Data Operation Support**. Data provisioning via *data services* enables workflow developers to use only those data management operations that are offered by available services. Typically, this is again a limited set of proprietary functions. *Data management activities* better support this criterion as they enable all data management commands the respective data resources offer. In case of database systems, for instance, the offered command languages are usually highly expressive and support various ETL operations (see Section 2.2). Nevertheless, command languages of file or operating systems, e. g., shell languages, are typically not intended for complex or proprietary operations. So, data management activities might exhibit only a few disadvantages for file systems and/or when proprietary operations have to be supported. Data management operations that are supported by the local data processing unit of the workflow system are in the toolbox for workflow developers that rely on *local data processing*. However, such a local data processing unit is oftentimes most inappropriate for ETL operations in terms of limited, proprietary functions and/or restricted workflow-internal data structures.

Criterion **Data Transfer Support** refers to the question to what extent a data provisioning concept supports the design and execution of tasks for data transfers between data resources. The main focus is on whether a concept offers sufficient means to support data transfers on its own or whether it depends on other concepts. Figure 4.3 illustrates the ways how the individual data provisioning concepts or their techniques may implement data transfers.

A service may (1) temporarily load the data into the service context, e. g., into variables of the used programming language, or it may (2) initiate a direct data transfer between the involved data resources. Data management activities support only the second way. For both data provisioning concepts, such a direct data transfer is only possible if it can be defined as data management command of the relevant data source or data sink. For instance, file systems usually offer commands for locally copying files in one computer, and in many cases even for remote file transfers between several computers via SSH tunnels. In contrast,

[4]Apache ODE external variables: `http://ode.apache.org/external-variables.html`

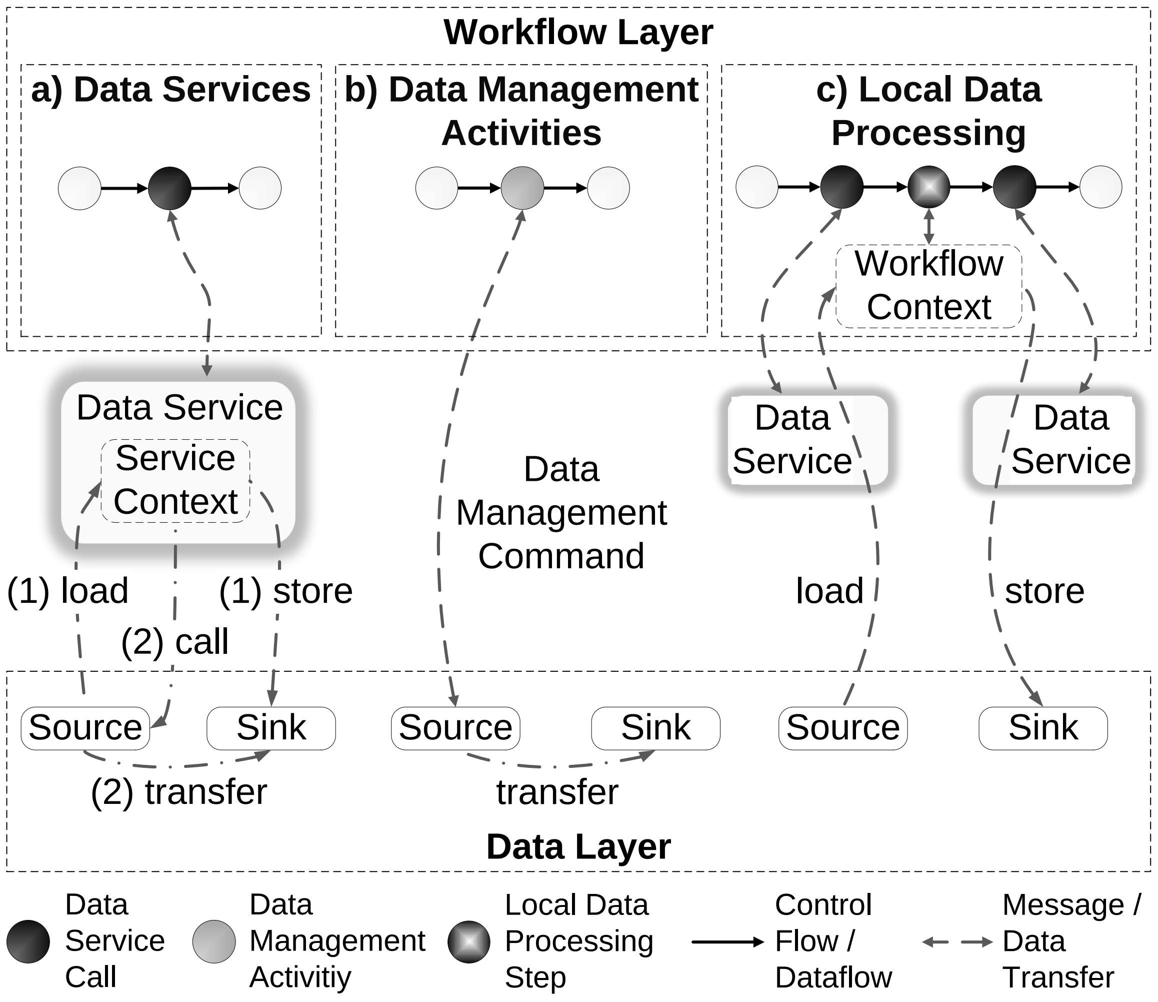


Figure 4.3: Data transfer support of the individual data provisioning concepts depicted in Figure 4.2.

database systems often do not support direct data transfers to other database systems, but only export or import functions to or from locally accessible files.

Whenever a direct data transfer is possible at all, *data management activities* may have an advantage compared to data services, as they allow workflow developers to use all commands the respective data resources offer. *Data services* may be more restrictive due to a possibly limited set of available service functions. When data resources do not support a direct transfer, data management activities cannot be used, while data services may exist that additionally enable a data transfer via the service context. In summary, data services and data management activities both show minor advantages and disadvantages and thus support criterion Data Transfer Support to nearly the same degree.

Only when both data services and data management activities do not provide data transfers at all, *local data processing* in workflow systems is indispensable. As depicted in Figure 4.3, the workflow context may then serve as temporary data storage. However, local data processing always depends on the other concepts, as data services or data management activities are needed to load data from the data sources into the workflow context, and to store it to the data sinks. Hence, this concept shows the worst support of criterion Data Transfer Support.

4.2.2.2 Support of Data Iteration Patterns

Criterion **Data Iteration Support** (fourth criterion shown in Table 4.1) aims at discussing whether a data provisioning concept offers sufficient means to support the Data Iteration Patterns on its own or whether it depends on other concepts. The comparison mainly depends on whether the relevant workflow inevitably has to load some external data into the workflow context in order to retain control on the data iteration. In particular, a workflow might need to load some data to properly determine the partitioning of the whole data set S into subsets $T_i \subseteq S$ (see Figures 3.5 and 3.6). In the following, such loaded data are called *loop data*. An example is a list of available Octave computers among which the coupling workflow shown in Figure 3.3 partitions the output data of Pandas.

In case the workflow inevitably needs to load loop data into the workflow context, the data provisioning concepts heavily depend on each other. Typically, data services or data management activities are used to initially load the loop data into the workflow context (see Figure 4.3). Local data processing in the workflow system is then used to further process this loop data in order to control the data iteration. So, local data processing is indispensable in such scenarios. In other words, both *data services* and *data management activities* are not sufficient to provide the full functionality of a data iteration and thus do not completely support this criterion (see Table 4.1).

If the workflow does not need to load loop data into the workflow context, data services or data management activities might be used to push down the whole data iteration to external resources or services [VSS$^+$07]. *Data services* may be implemented via typical programming languages, which usually provide means for loops or native commands for data partitioning and thus also for data iterations. However, there must be an appropriate data service available for a particular data iteration scenario. *Data management activities* can only be used if the data resources that are responsible for controlling the data iteration offer commands to completely execute the iteration in these resources. Examples may be SQL Cursors or other set-oriented operations of relational database systems. However, the default command languages offered by data resources usually do not provide the same range of opportunities as programming languages implementing services. So, criterion Data Iteration Support is the first one, where data management activities show a few disadvantages compared to data services (see Table 4.1).

As discussed above, *local data processing* always depends on the other concepts in case data services or data management activities are used to load loop data into the workflow context. If the workflow does not inevitably need to load such loop data, local data processing might even not be necessary at all. This is particularly true if the other two concepts offer enough means to completely execute a data iteration in external resources or services. As conclusion, local data processing again shows the worst support of criterion Data Iteration Support.

4.2.3 Comparison Regarding Non-Functional Issues

The following subsections deal with the two main comparison criteria regarding non-functional issues illustrated in Section 4.2.1, i. e., (1) the information on the data management of a workflow model and (2) the abstraction support offered by data provisioning techniques. Table 4.2 summarizes the results of this discussion. Note that, for this part of the comparison, it is assumed that the data provisioning concepts or their underlying techniques offer the needed functionality, as such functional aspects are already covered by Section 4.2.2.

4.2.3.1 Information on Data Management

Criterion **Information on Data Management** deals with the question how much information on the data management of a workflow is known or can be derived by certain components associated with a workflow system. On the one hand, this is crucial for an optimizer component in order to make proper optimization decisions increasing the efficiency of data processing in workflows [BHW+07, VSS+07]. On the other hand, it is important for provenance components in order to properly correlate collected provenance information with affected parts of a workflow model [KSB+10, RSM14a, MBBL15].

Table 4.2: Support of non-functional issues. The filling degrees of the pie charts depict how much a data provisioning concept supports the respective criteria.

Criteria	**Data services**	**Data management activities**	**Local data processing**
Offered *Information on Data Management*	◔	●	◑
Offered *Abstraction Support*	●	◔	◑

The abstraction layer introduced by *data services* hides most of the information on the data management of workflows. So, it is very difficult for a workflow system or associated components to get much of this information. Usually, this is restricted to single input parameters of a service call, which do not give detailed insights about data management operations. The data provisioning concepts of *local data processing* and *data management activities* better support this criterion, because they treat descriptions of data management operations as integral parts of workflow models. Hence, the workflow system may analyze these workflow models to get much more information it needs for a holistic provenance support and for a consolidated optimization of the data processing [VSS$^+$07, RRS$^+$11]. While data management activities may support various optimization techniques [BHW$^+$07, VSS$^+$07, RRS$^+$11, Kal15], local data processing is however restricted to those that are tailor-made for the workflow-internal data processing [RSM11]. For instance, a distinguishing drawback of local data processing is that only access to the local data processing unit in the workflow system is possible. So, this data provisioning concept oftentimes prohibits the distributed execution of data management operations, which would be necessary for task and data parallelism [PA06]. As conclusion, the concept of local data processing supports criterion Information on Data Management to a less degree than the concept of data management activities (see Table 4.2).

4.2.3.2 Abstraction Support

The purpose behind criterion **Abstraction Support** is to evaluate the suitability of a data provisioning concept to provide scientists with an adequate abstraction support reducing the complexity of simulation workflow design. The main issue is whether scientists have to acquire some technical knowledge that is necessary for successfully applying a particular data provisioning technique. For acceptability purposes, the effort scientists spend on this knowledge acquisition should be minimal, which corresponds to a good abstraction support as depicted in Table 4.2. Otherwise, scientists would waste time they actually want to spend on their core issues, i. e., simulation modeling and post-execution analysis (see Figure 2.3).

For data provisioning based on *local data processing*, scientists have to get familiar with languages to describe workflow-local data processing steps. Examples are copy statements and corresponding XPath expressions of a BPEL `assign` activity [OAS07] (see Section 2.3.3). Scientists typically do not perceive such languages as integral parts of their domain-specific methodology. So, scientists are often not familiar with these languages and may have to spend a considerable effort to learn them. In a similar way, *data management activities* require scientists to get familiar with the command languages the involved data resources offer, e. g., shell languages or even SQL or XQuery. The diversity of the respective languages is usually higher than for workflow-local data processing. So, the effort scientists have to spend on applying them may be higher as well. In fact, this effort even increases with the complexity of involved data management operations. Altogether, this leads to a worse support of criterion Abstraction Support for data management activities. The abstraction layer introduced by *data services* hides most of the complexity of the involved data management technology and operations. Scientists only need to specify the input parameters of the relevant service call, which can typically be done in an easy way. Furthermore, the wide acceptance of service technology in the scientific domain entails that many scientists are quite familiar with service calls [TDG07]. So, the effort they have to spend is minimal for this data provisioning concept, i. e., data services show the best support of criterion Abstraction Support.

4.3 Guidelines for Choosing Appropriate Data Provisioning Techniques

The first focus of this section is to use the results of comparing data provisioning concepts in order to derive guidelines for simulation workflow design. These guidelines assist scientists in choosing appropriate data provisioning techniques for certain data provisioning tasks of their simulation workflows. Figure 4.4

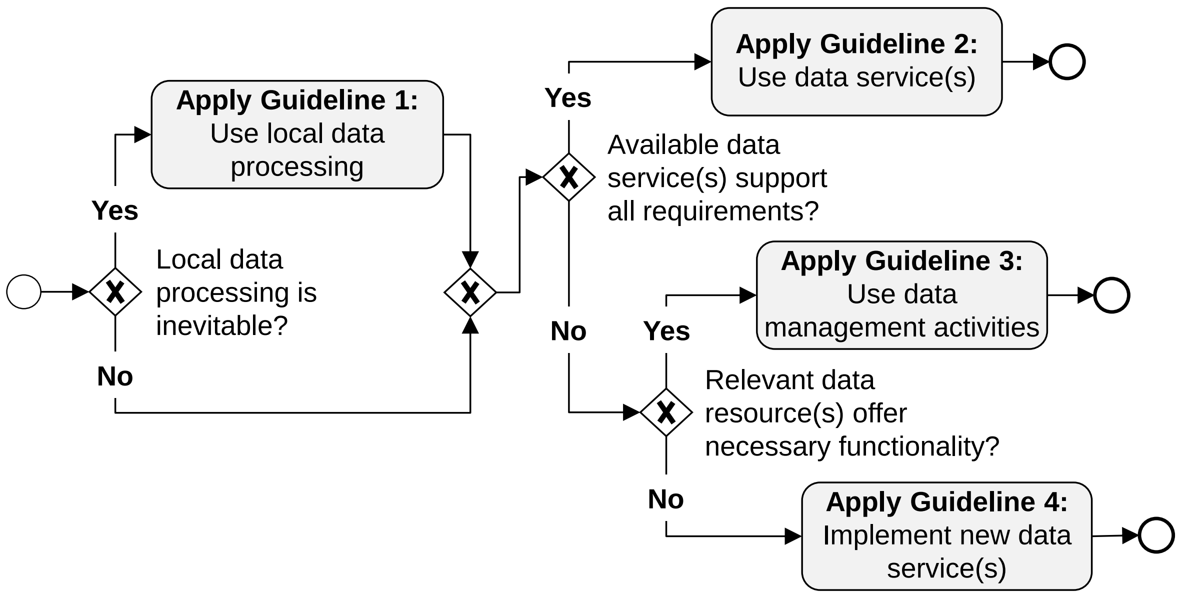

Apply Guideline 1:
Use local data processing
Yes
Local data processing is inevitable?
No
Yes
Available data service(s) support all requirements?
No
Apply Guideline 2:
Use data service(s)
Yes
Apply Guideline 3:
Use data management activities
Relevant data resource(s) offer necessary functionality?
No
Apply Guideline 4:
Implement new data service(s)

Figure 4.4: Sequence of decisions to apply the guidelines for choosing appropriate data provisioning techniques.

shows the sequence in which scientists may apply the guidelines. In addition, the figure depicts the decisions scientists have to make to check whether a guideline may be applied or not. Moreover, the guidelines have been evaluated by successfully applying them to the real-world simulation examples listed in Section 3.1. Section 4.3.2 illustrates the major results of this evaluation.

4.3.1 Discussion of Guidelines

An overview over the comparison results summarized in Tables 4.1 and 4.2 leads to the assumption that local data processing offers the worst solution for data provisioning in simulation workflows and thus should be avoided as often as possible. In addition, data services and data management activities compete against each other with alternating success. The following subsections discuss these two aspects in more detail.

4.3.1.1 Local Data Processing as Solution with Many Drawbacks

The discussion in Section 4.2.2 reveals that local data processing in workflow systems shows the worst support of data management patterns among all three data provisioning concepts. This holds for each individual comparison criteria depicted in Table 4.1. For instance, local data processing significantly limits the range of supported data management operations, which oftentimes makes this concept inappropriate for data provisioning in simulation workflows. This impression is even reinforced by the comparison regarding non-functional issues summarized in Table 4.2. As discussed in Section 4.2.3.1, local data processing offers lots of information about the data management in workflows. However, this fact may become insignificant, since this concept restricts the set of possible optimization techniques that might make use of this information. Finally, it only offers a low abstraction support, especially when compared to data services.

Guideline 1 *Local data processing in workflow systems should only be used in simulation workflows when this is inevitable or when proprietary needs call for it.*

The Data Iteration Patterns shown in Figures 3.5 and 3.6 constitute the major scenario where local data processing may be inevitable. For instance, the coupling workflow depicted in Figure 3.3 needs to load a list of available Octave computers. This workflow then processes the list locally in the workflow context to properly partition the output data of Pandas among the computers. Note that even if scientists choose the concept of local data processing, they still need to check which of the other guidelines has to be applied (see Figure 4.4). This is because local data processing seldom offers the full functionality of any data provisioning task. For instance, data services or data management activities are needed to load the above-mentioned list of Octave computers into the workflow context.

4.3.1.2 When to Use Data Services or Data Management Activities

Now, the question is under which circumstances scientists should prefer data services to data management activities or vice versa. Table 4.1 shows that, in contrast to data services, data management activities offer full control over the selection of data resources. Furthermore, they support the full range of data management operations offered by the involved external data resources via their command languages. Data services, however, only provide a limited set of proprietary functions, which may even restrict the available functionality offered by data resources. So, data management activities show a better support of data management patterns, but only if the involved data resources offer the necessary functionality at all. When this is not the case, this functionality of data resources needs to be enhanced via additional data services. This may be necessary, e. g., when operating systems are involved, whose command languages are typically not intended for highly complex data provisioning tasks in simulation workflows.

The discussion regarding non-functional issues summarized in Table 4.2 likewise does not clearly argue for data services or data management activities. The main benefit of data management activities is that they offer much information about data management operations in workflow models. This is a prerequisite for a holistic provenance support [KSB+10, RSM14a, MBBL15], and it facilitates making proper optimization decisions to increase the efficiency of data processing in workflows [BHW+07, VSS+07, RRS+11]. On the contrary, data services offer a much better abstraction support that reduces the effort scientists have to spend to successfully apply corresponding data provisioning techniques.

None of these arguments nominates a distinct winner among data services and data management activities. This shows in conclusion that the proper choice between these two data provisioning concepts depends on the concrete scenario. Nevertheless, the comparison results may be used to derive the following guidelines for choosing either data services or data management activities.

Guideline 2 *Whenever a data service exists that supports all desired functional and the most relevant non-functional requirements, this service should be used. This constitutes the most convenient solution for scientists, since data services offer the best abstraction support of all data provisioning concepts.*

Guideline 3 *When a suitable data service does not exist, but the relevant data resources offer the necessary functionality, data management activities are a good solution. This is mainly due to the functional flexibility offered by data management activities, and since the external resources typically support more or less efficient means to execute various data management operations. However, note that data management activities may require scientists to get familiar with previously unknown command languages of some of the data resources.*

Guideline 4 *The last case is when neither existing data services nor command languages of data resources and thus data management activities offer the necessary functionality. So, scientists need to implement new proprietary data services that enhance existing functionality of data resources. Then, scientists not only need to get familiar with command languages of data resources, but also with scripting or programming languages to implement data services. Hence, this option should only be used if it is absolutely inevitable.*

4.3.2 Evaluation of the Proposed Guidelines

In the course of working on this thesis, the proposed generic guidelines have been evaluated via a comprehensive case study. Thereby, they have been successfully applied to the real-world simulation examples listed in Section 3.1. This has been backed up by collaborations with research partners conducting the simulation examples. Before these research partners knew about the guidelines, they were confused about the large and complex set of available data provisioning techniques, as illustrated in Section 1.2.1. This diversity and complexity of techniques induced them not to leverage the techniques at all. Instead, they still implemented or even manually performed the data provisioning on their own. Finally, the systematic approach offered by the proposed guidelines significantly helped them to choose appropriate data provisioning techniques and to use them in their workflows.

During this evaluation process, it turned out that the running example of a bone simulation depicted in Section 1.1 is well-suited to illustrate all major results of this evaluation. This is mainly because this bone simulation is the largest and most complex scenario of all considered simulation examples, as discussed in Section 3.1. The most important implications and conclusions that have been derived for other examples are hence likewise covered by the bone simulation. In the following two subsections, this discussion is split according to the two kinds of the most challenging scenarios also considered in Section 3.2: (1) providing and preparing heterogeneous input data for single simulation models and (2) exchanging data between different coupled simulation models. Finally, Section 4.3.2.3 summarizes the main conclusions of this evaluation.

4.3.2.1 Provisioning of Input Data for Single Simulations

Figure 4.5 illustrates how the proposed guidelines map to a prototypical implementation of the bio-mechanical simulation workflow depicted in Figure 1.1. This prototypical implementation of the workflow is based on the workflow language BPEL and on the workflow engine Apache ODE[5] [Dor11, Pie12, Boh14]. No step in this simulation workflow inevitably requires workflow-local data processing. Hence and in line with guideline 1, the workflow completely neglects this data provisioning concept. Figure 4.5 therefore only depicts the usage of either data services or data management activities for relevant data provisioning tasks.

[5]Apache ODE: `http://ode.apache.org/`

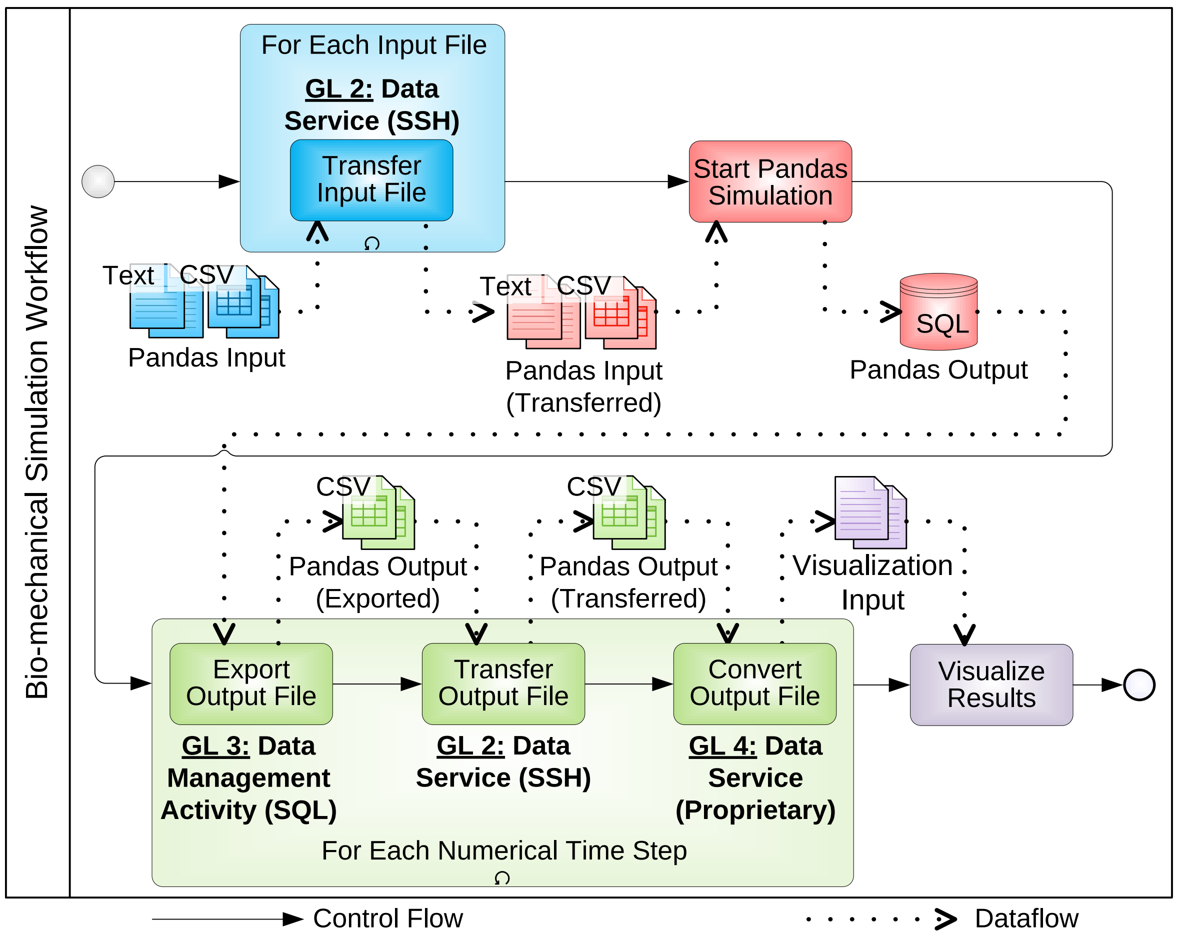


Figure 4.5: Usage of data provisioning concepts in the bio-mechanical simulation workflow shown in Figure 1.1. Data provisioning tasks and the respectively used concepts are highlighted via bold labels. “GL *i*” means “Guideline no. *i*”.

In its prototypical implementation, the data provisioning phase of the workflow (blue steps shown in Figure 4.5) is a simplified version of the one depicted in Figure 1.1. More precisely, it does not access the XML or SQL database systems storing material parameters or FEM parameters. Instead, it copies the necessary text- or CSV-based input files of Pandas from a set of source computers to the computer where Pandas is installed. As illustrated in Section 4.1.1.1, corresponding file transfer services, e. g., using SSH tunnels, are common in the workflow or service domains. So, it is likely that a data service exists realizing an appropriate file transfer. According to guideline 2, this data service is the choice to realize this task. Thereby, a BPEL `forEach` activity iterates over a list of relevant input files and initiates one instance of the data service for each file.

After Pandas has stored the simulation outcome in its SQL database, the workflow prepares this result data for the subsequent visualization (green and purple steps shown in Figure 4.5). The tool used for this final result visualization expects one input file for each numerical time step calculated by Pandas. For each of these time steps, the workflow firstly exports the relevant data from the Pandas database and stores it in a CSV-based file. The underlying database system – a PostgreSQL[6] version 9.2 – supports these data exports via SQL statements, and its optimization mechanisms enable the efficient execution of these statements. These aspects, as well as the fact that often no data service offers such a specific functionality, recommend following guideline 3. So, the functional flexibility of a data management activity embedding an appropriate SQL statement is used as realization of the data export. Subsequently and according to guideline 2, a data service again uses an SSH tunnel to copy the previously exported file from the Pandas computer to the computer where the visualization tool resides. The next activity converts this CSV-based file into the format suitable for the visualization tool. In this case, neither existing data services nor the file or operating system of the visualization computer offer sufficient means for this proprietary data format conversion. So, the only option is guideline 4, i. e., the offered functionality needs to be enhanced via a new proprietary data service.

4.3.2.2 Data Exchange between Different Simulations

Figure 4.6 depicts how to apply the proposed guidelines to the more complex scenario of coupling the bio-mechanical simulation model with a systems-biological model discussed in Section 3.2.2. The figure mainly indicates the choices among the data provisioning concepts for a BPEL-based prototypical implementation of the coupling workflow shown in Figure 3.3 [Dor11, Pie12, Ges14, Rie14, vS15b].

The repository accessed by the first workflow step offers a suitable service interface to load the list of daily routines including their motion sequences. So, guideline 2 is applicable for this workflow step, i. e., it uses the corresponding service. This also holds for the next step that loads a list of available Pandas computers from another repository. Afterwards, the workflow assigns each motion sequence of the current daily routine to a Pandas computer calculating the simulation outcome for this motion sequence. Therefore, the corresponding workflow step has to process the previously loaded list of daily routines and

[6]PostgreSQL: `http://www.postgresql.org/`

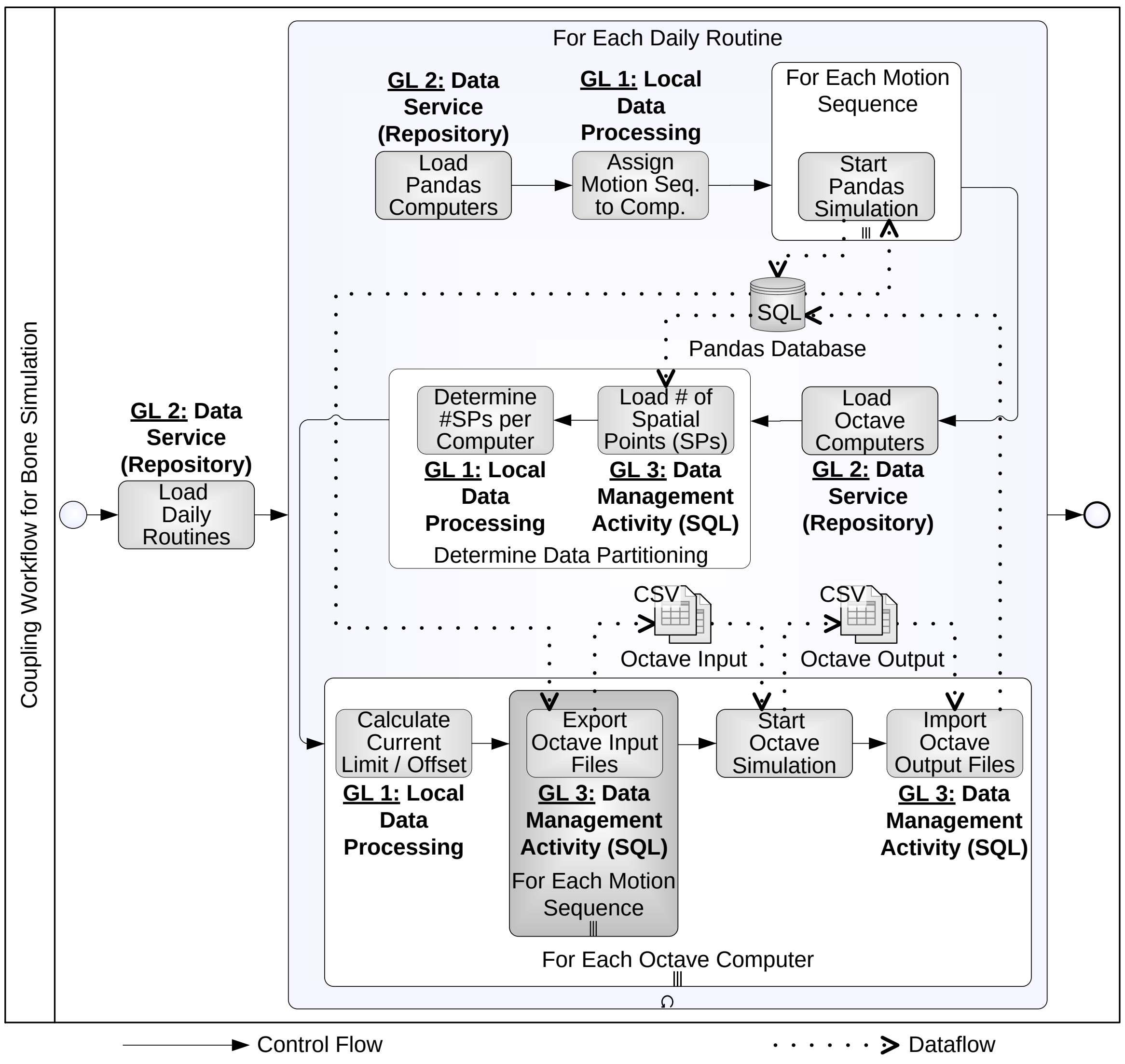

Coupling Workflow for Bone Simulation
For Each Daily Routine
GL 2: Data Service (Repository)
Load Daily Routines
GL 2: Data Service (Repository)
Load Pandas Computers
GL 1: Local Data Processing
Assign Motion Seq. to Comp.
For Each Motion Sequence
Start Pandas Simulation
SQL
Pandas Database
Determine #SPs per Computer
GL 1: Local Data Processing
Load # of Spatial Points (SPs)
GL 3: Data Management Activity (SQL)
Load Octave Computers
GL 2: Data Service (Repository)
Determine Data Partitioning
CSV
Octave Input
CSV
Octave Output
Calculate Current Limit / Offset
GL 1: Local Data Processing
Export Octave Input Files
GL 3: Data Management Activity (SQL)
For Each Motion Sequence
Start Octave Simulation
Import Octave Output Files
GL 3: Data Management Activity (SQL)
For Each Octave Computer
Control Flow
Dataflow

Figure 4.6: Usage of data provisioning concepts in the coupling workflow shown in Figure 3.3. Data provisioning tasks and the respectively used concepts are highlighted via bold labels. "GL i" means "Guideline no. i"; cf. [RSM14b].

list of Pandas computers within the workflow context. Hence, this is the first part of the considered workflows where local data processing is inevitable and guideline 1 has to be applied. Subsequently, each Pandas simulation is started by calling a similar workflow as the one depicted in Figure 4.5.

Loading a list of available Octave computers leads to the same arguments as for the other two repository services described above, i. e., guideline 2 may be applied here. The following two workflow steps prepare the later partitioning of the output data of Pandas among these Octave computers. As discussed in Section 2.1.1, Pandas calculates its outcome for a set of spatial evaluation points of the finite element grid. The second of the two steps determining the data partitioning calculates the number of spatial evaluation points each available Octave computer has to process. This step therefore needs access to the previously loaded list of Octave computers. So, the workflow again inevitably has to process some data locally in the workflow context, i. e., guideline 1 has to be applied. As depicted in Figure 4.4, it is still necessary to check whether other data provisioning concepts are needed to previously load some external data into the workflow context. In fact, the prior workflow step needs to load the total number of spatial evaluation points for which Pandas has stored the simulation outcome in its database. Again, the underlying PostgreSQL database system offers the efficient execution of an appropriate SQL statement, but there is no data service available providing this functionality. According to guideline 3, this workflow step is thus realized by a data management activity.

Subsequently, the actual parallel data iteration for each Octave computer starts. The first workflow step calculates the limit and offset numbers of the spatial evaluation points to be processed by the current Octave computer. Here, again guideline 1 needs to be applied, as it is inevitable for this workflow step to locally process the previously loaded data values. The next step realizes the data export, i. e., the format conversion, filter, aggregation, and partitioning of the data from the database of Pandas to the relevant CSV-based files. For this step, guideline 3 is applicable and a data management activity is chosen due to the same reasons as described above when accessing the PostgreSQL database system. After Octave has finished its calculation in the next workflow step, the last step of the coupling workflow imports the resulting CSV-based output files back into the database of Pandas. This step may again use a data management activity that issues an appropriate SQL statement against the PostgreSQL database system.

4.3.2.3 Main Conclusions

The guidelines derived in Section 4.3.1 helped to choose a realization for each individual data provisioning task in the workflows depicted in Figures 4.5 and 4.6. This completeness of the guidelines has been confirmed when also testing them on the other example simulations considered for the evaluation [Mül10, Wag10, FE11, RK11, Rem11, FDP15]. In addition, it turned out that each individual guideline is relevant for simulation workflows. Furthermore, the guidelines and the order in which Figure 4.4 proposes to apply them always led to the right decisions on realizations of data provisioning tasks. For instance, the application of guideline 1 was essential for the coupling workflow shown in Figure 4.6. In particular, this workflow inevitably needs to load a portion of external data to control the data partitioning, which is a common requirement for the Parallel Data Iteration Pattern depicted in Figure 3.5 [RSM14b]. Figure 4.4 arranges the other three guidelines in an order that aims at reducing the effort scientists have to spend on simulation workflow design. Guideline 2 constitutes the most convenient solution for scientists, since data services abstract from many low-level details of data provisioning. If this guideline was neglected or postponed in Figure 4.4, scientists would need to specify many complex data management operations in data management activities according to guideline 3. Neglecting or postponing guideline 3 may even further increase the scientists' effort, because the next guideline 4 recommends to implement completely new data services. As discussed in Section 4.3.1.2, this would result in a remarkable implementation overhead. Guideline 4 is however necessary, since scenarios may exist where neither available data services nor data resources offer the necessary functionality.

4.4 Implications for Simulation Workflow Systems

The guidelines discussed in Section 4.3 provide scientists with a generic method to choose appropriate data provisioning concepts or their underlying techniques. However, existing workflow systems also need to offer particular features that are necessary for effectively applying the guidelines in practice. As a simple example, guideline 3 requires workflow systems to offer adequate kinds of data management activities supporting the used data resources. Section 4.4.1 illustrates the impacts of the comparison results summarized in Section 4.2 and of the guidelines derived in Section 4.3 on these necessary features of workflow systems. In particular, a set of features are discussed that are missing in the majority of currently available systems. Section 4.4.2 then illustrates extensions to a simulation workflow system offering all the missing features in a holistic way. Finally, the prototypical implementation of this extended workflow system, which has been developed while working on this thesis, is depicted in Section 4.4.3. Detailed results of the evaluation of this prototype are covered by Chapters 5 and 6.

4.4.1 Missing Features of Current Workflow Systems

At first glance, the discussion of missing features of currently available workflow systems improves the situation of developers of these systems. It helps them to identify the features they need to incorporate into their systems, depending on which of the guidelines discussed in Section 4.3.1 they want to support. Nevertheless, it is likewise beneficial for scientists using the workflow systems to design and execute workflows. These scientists may follow the sequence of decisions depicted in Figure 4.4 and determine the guidelines they have to apply. The discussion in this section then helps scientists to identify the features a particular workflow system has to offer so that they can effectively apply the guidelines. This way, they are able to systematically select a workflow system that offers all the features they need.

Guideline 1 recommends to avoid local data processing in workflow systems whenever possible. As mentioned in Section 4.2.2, the by far most common case where local data processing may be inevitable is represented by the Data Iteration Patterns depicted in Figures 3.5 and 3.6. These patterns however only require carrying out less complex operations on very small data sets within the workflow context in order to properly control the data partitioning. For instance, the coupling workflow shown in Figure 4.6 only carries out simple loops or count operations on small lists of daily routines or available computers. As discussed in Section 4.1.1.3, any available workflow system offers means that can smoothly cope with these less complex operations and small amounts of data. Hence, guideline 1 does not imply any missing feature of these workflow systems.

Guideline 4 should likewise only be used if this is absolutely inevitable. Hence, workflow systems should offer enough features so that scientists have to apply guideline 4 as seldom as possible. In other words, the features offered by workflow systems should make it possible to apply guidelines 2 and 3 in the majority of all cases. So, there is no need to derive missing features for guideline 4, but only for the other guidelines. Table 4.3 summarizes the resulting missing features.

4.4.1.1 Missing Features for Guideline 2

Guideline 2 suggests to use data services in case one or more services exist that support all desired functional and the most relevant non-functional requirements. Most of today's workflow systems offer general means to invoke services and thus

Table 4.3: Missing features of currently available workflow systems that are required to effectively apply the guidelines discussed in Section 4.3.1.

Guideline 2: Use data service(s)	1. Registry that describes data services and data resources in a domain-specific manner. 2. Mechanisms to enrich the information on the data management of a workflow.
Guideline 3: Use data management activities	3. Generic data management activities supporting various data resources, formats, and operations. 4. Abstraction support for workflow design that is tailor-made for scientists.

also support this data provisioning concept (see Section 4.1.1.1) Nevertheless, this guideline and the comparison results regarding data services summarized in Section 4.2 though entail two missing features of available workflow systems:

1. Guideline 2 is only applicable if the used workflow system (1) enables scientists to describe the requirements they have and (2) supports mechanisms to search for services matching these requirements. In the service domain, many generic repository or registry solutions exist that support these scenarios [Ley03, Ley05]. This is oftentimes based on policies to specify requirements that are matched against metadata describing the capabilities and non-functional properties of available services. However, these generic solutions always have to be adapted to the domain of the service consumers in order to facilitate an effective search for services [SKRM13]. This is especially challenging for simulations that are coupled across different scientific domains, as each domain has its own requirements. Previous work, however, has widely neglected the specifics of this large area of coupled simulations – and often also of the individual domains, such as bio-mechanics or systems-biology. Hence, the first missing feature of available workflow systems is a *repository or registry solution that allows for a domain-specific description of data services that is especially tailored to (coupled) simulations.*
2. On the one hand, data services offer the best abstraction support for specifying data management operations (see Table 4.2). On the other hand and as discussed in Section 4.2.3.1, this abstraction support tends to hide much of the information on the data management of a workflow. Such information is however essential for a successful application of optimization techniques

and for a holistic provenance support. So, a workflow system has to offer *mechanisms to enrich the information on the data management of a workflow* again in case it has previously been hidden by data services. However, only very few workflow systems exist tackling this problem at all, and these few systems usually do this to a rather marginal degree [CVDK+12].

4.4.1.2 Missing Features for Guideline 3

Scientists may only apply guideline 3, i. e., use data management activities for data provisioning, if the underlying data resources offer the necessary functionality (see Figure 4.4). So, scientists need to be able to search for data resources matching their requirements. This may again be facilitated by a domain-specific description of data resources in repositories or registries. For that purpose, the first missing feature discussed above not only considers data services, but also data resources (see Table 4.3). Moreover, guideline 3 and the comparison results summarized in Section 4.2 entail the following missing features of workflow systems:

3. As discussed in Section 4.1.1.2, only a couple of today's workflow systems support the data provisioning concept of data management activities. This encompasses the workflow products of IBM, Microsoft, and Oracle [VSRM08], as well as the scientific workflow systems Kepler and Trident [LAB+06, BJA+08]. A distinguishing drawback of all these systems is that they are often restricted to certain kinds of data resources, e. g., SQL database systems [VSRM08], or even to specific proprietary data management operations [BAJ+10]. Especially simulations that are coupled across different scientific domains however require a *generic solution with data management activities supporting various kinds of data resources, data formats, and data management operations* [RSM14a, RSM14b].
4. As depicted in Table 4.2, data management activities offer the worst abstraction support of all data provisioning concepts. Scientists often do not use data management activities, as they would have to spend much effort to successfully specify data management operations in such activities [RLSR+06, RS14, RSM14b]. Hence, workflow systems should offer an *abstraction support* for specifying low-level data management operations *that is particularly tailor-made for scientists conducting simulations*. However, none of today's workflow systems offers such a domain-specific and user-centric abstraction support [RSM14a, RSM14b] (see Section 1.2.2).

4.4.2 Extended Simulation Workflow System

Figure 4.7 shows the main components of the architecture of an extended simulation workflow system supporting all missing features discussed above. It corresponds to the SIMPL framework mentioned in Section 1.3.2 as one of the contributions of this thesis [RRS+11, RS13b, RS14, RSM14a, RSM14b]. Note that, for better readability, Figure 4.7 leaves out components of a general architecture of workflow systems that are less relevant for supporting the missing features discussed above. This especially includes a monitoring tool, a library managing workflow fragments, or components for binding data resources, services, or workflow fragments at runtime [KWvL+07, SK10, GSK+11, SK11, SKK+11, SK13]. The main components being relevant here are the *workflow design tool*, the *workflow execution environment*, the *rule-based pattern transformer*, the *simulation artifact registry*, the *provenance framework*, and the *service bus*. The following subsections respectively detail how this extended workflow system supports the individual missing features summarized in Table 4.3.

4.4.2.1 Domain-specific Registry for Data Services and Data Resources

The *simulation artifact registry* mainly supports the first missing feature. Its major purpose is to help scientists to find data services or data resources that match their particular requirements. Therefore, it does not only contain metadata describing these *data services* and *data resources*, but also other domain-specific key artifacts scientists conducting simulations are especially interested in. This mainly encompasses mathematical *simulation models*, *simulation methods*, e. g., numerical methods, and *simulation software* [RS13b, RSM14a]. As a further contribution, the registry manages dependencies between these different artifacts. These dependencies facilitate the domain-specific search for data services or data resources and thus make this search tailor-made for scientists. More precisely, scientists may search for data services or data resources by actually querying the simulation models, methods, or software. For instance, they may search for data resources storing the data a particular simulation software requires to realize a specific simulation model by means of a certain simulation method [RS14, vS15a]. Other examples are coupled simulations, where scientists may ask for data services implementing the numerical, multi-scale transformations between mathematical variables of the coupled simulation models [Gat14, RSM14b, vS15b].

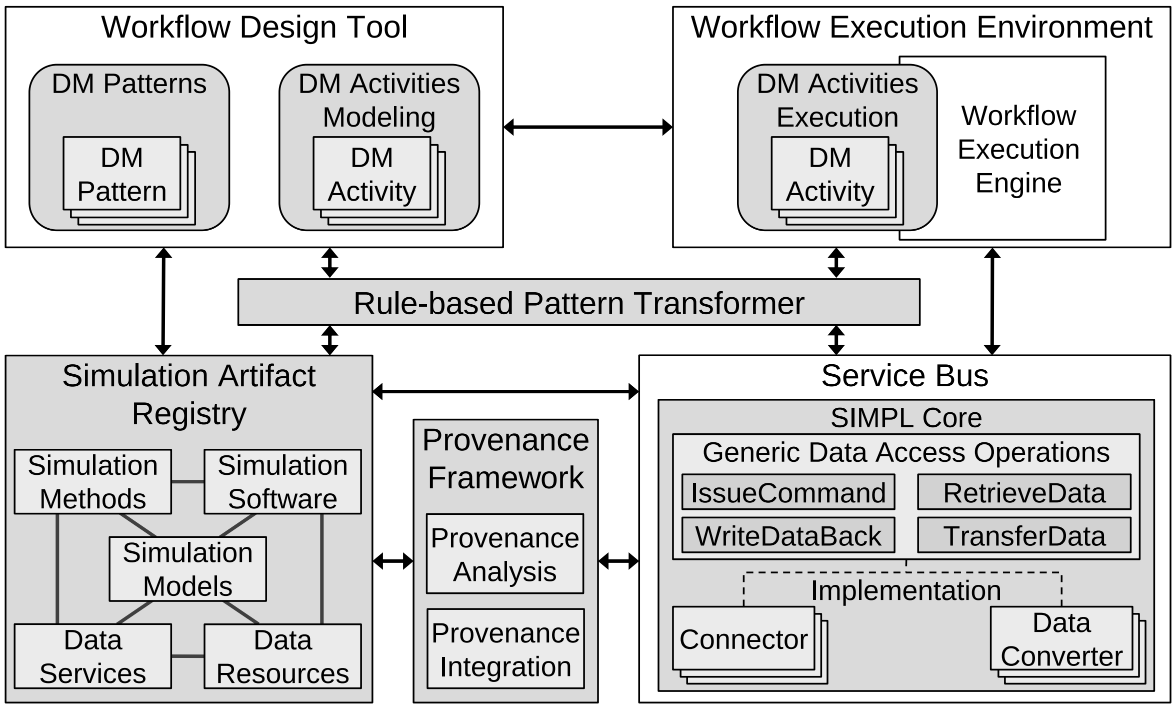


Figure 4.7: Main components of a simulation workflow system and its extension by SIMPL to support the missing features of available workflow systems summarized in Table 4.3; cf. [RRS+11, RS13b, RS14].

4.4.2.2 Provenance Framework to Enrich Information on Data Management of a Workflow

The second missing feature is provided by the *provenance framework* depicted in Figure 4.7. This component offers mechanisms to enrich the metadata managed by the simulation artifact registry with additional and more detailed information about the data management of a workflow. This is especially relevant in case data services or another kind of abstraction support hide much of this information [RSM14a]. Thereby, different kinds of provenance information may be captured and managed by various provenance sources. Examples of such provenance sources are the individual components of the workflow system depicted in Figure 4.7, as well as external data resources or service execution environments such as cloud infrastructures [CVDK+12]. Even simulation software usually logs information about calculations and about parameterizations of numerical methods like the FEM. Since different provenance sources may manage their information in various kinds of structured or unstructured data formats, the provenance framework needs to *integrate* and aggregate this information [CVDK+12]. After having captured and integrated provenance information, it may serve as basis for manifold analyses. Such a *provenance analysis* among different provenance sources may significantly enhance the reproducibility of a simulation and of its outcome [HTT09]. Furthermore, it may help to derive recommendations for sophisticated optimizations, e. g., increasing the efficiency of workflows [RM09].

As an example, assume that the quality of the result data of Pandas rapidly decreases during the calculation in the workflow depicted in Figure 4.5 [RBD+11]. A possible reason of such a data quality violation is an inaccurate parameterization of the FEM, e. g., a too coarse granularity of the finite element grid or too short numerical time step sizes [RTD+12]. A prerequisite to identify this reason is that the provenance framework not only integrates information about data, but also about such FEM-specific parameters, which are logged by the Pandas software. As optimization, the provenance analysis may recommend to change the FEM-specific parameters and to re-execute affected parts of the calculation [RBKK12, SK12].

4.4.2.3 Generic Data Management Activities

The ETL workflow approach introduced in Section 1.3.2.1 facilitates generic data management activities supporting various data resources, formats, and operations. It is corporately provided by several components or plug-ins of the workflow system depicted in Figure 4.7. The *SIMPL core* offers a data access abstraction via generic data access operations. These operations allow for a unified access to arbitrary data resources and thereby support the most common use cases for data access in simulation workflows [RRS+11]. This covers the operations `IssueCommand` for data manipulation or data definition, `RetrieveData` to load external data into the workflow context, `WriteDataBack` to write data back to external resources, and `TransferData` to transfer data between several resources. Technical details of data access mechanisms, e. g., how to connect to a specific data resource, are covered by the *connector* plug-ins, which implement the generic access operations for concrete data resources. For the `RetrieveData` and `WriteDataBack` operations, *data converters* transform data between the format of a connector and the format a workflow requires. Additionally, the simulation artifact registry manages metadata to explicitly describe *data resources*. These metadata particularly define the mappings between the SIMPL core operations and the technical details of how to access individual data resources. For instance, they associate each resource with the proper connector and data converters.

In addition, SIMPL extends both the workflow design tool and the workflow execution engine. The data management (DM) activities offered by these extensions correspond to the SIMPL core operations `IssueCommand`, `RetrieveData`, `WriteDataBack`, and `TransferData`. They allow for using the respective functionality directly within workflows [RRS+11]. Workflow developers, e. g., scientists, may assign an arbitrary data resource to such an activity and specify a command in the command language of this resource, e. g., a SQL statement or a shell command. During activity execution, this command is issued over the SIMPL core against the relevant data resource. This helps scientists to design their workflows without being forced to provide any technical details of data access mechanisms, as these details are already covered by the SIMPL core.

4.4.2.4 Pattern-based Abstraction Support for Scientists

Scientists however still have to specify many sophisticated data management operations in data management activities, e. g., in terms of complex SQL or XQuery statements [RSM14b]. To offer an abstraction support that is tailor-made for scientists, SIMPL further extends the workflow design tool by a component that supports pattern-based workflow design [RS13b, RS14, RSM14a, RSM14b]. This component comprises a customizable list of data management (DM) patterns as templates and building blocks for typical data provisioning tasks in simulation workflows, e. g., the patterns described in Section 3.3. Scientists only need to select a few patterns, set them into their workflow models, and specify a small set of abstract parameter values for each pattern.

Figure 1.2 on page 34 exemplifies this core idea in that it shows how such patterns may simplify the design of the bio-mechanical simulation workflow depicted in Figures 1.1 and 4.5. As discussed in Section 1.3.2.2, this pattern-based approach significantly reduces the number of workflow tasks that are visible to scientists. Furthermore, the patterns shown in Figure 1.2 are particularly meaningful to scientists conduction simulations, as they represent use cases these scientists are interested in. Thereby, all pattern parameters correspond to artifacts or concepts scientists already know from their simulation models or domain-specific methodology. Hence, the pattern-based approach makes simulation workflow design especially tailor-made for scientists. In fact, it completely removes their burden to specify any low-level details of data provisioning or data exchange in their workflows [RSM14a, RSM14b].

The *rule-based pattern transformer* shown in Figure 4.7 manages an extensible set of rewrite rules that specify the transformation of patterns into executable workflow fragments [SKK+11, RSM14a]. The workflow fragments may then contain data service calls, data management activities, or even workflow-local data processing steps realizing the original patterns. The rewrite rules also need to map the abstract, simulation-specific pattern parameters onto more concrete implementation details. Therefore, they use metadata of the simulation artifact registry describing dependencies between all artifacts depicted in Figure 4.7. For instance, dependencies between simulation models and data resources may refer to concrete data containers that store the input and output data of the bio-mechanical simulation model used as pattern parameter in Figure 1.2.

4.4.3 Prototypical Implementation of the Extended Simulation Workflow System

The extended simulation workflow system depicted in Figure 4.7 has been prototypically implemented in the course of various student projects that have accompanied the work on this thesis [HSR+10, Pie11, Ari12, Pie12, Boh14, Rie14, Boh15, Kal15, vS15a, vS15b]. As discussed in Section 2.3, this prototype is based on the workflow language BPEL [OAS07]. It uses the workflow design tool Eclipse BPEL Designer[7] version 0.8.0 and the workflow execution environment Apache Orchestration Director Engine (Apache ODE)[8] version 1.3.5. Moreover, Apache Tomcat[9] version 7.0 and Axis2[10] version 1.5.4 provide the core functionality of a service bus. All other components shown in Figure 4.7 have been implemented as plug-ins of the respective tools or as separate Java-based Web Services. In its current state, the prototype covers the extensions introduced in Sections 4.4.2.1, 4.4.2.3, and 4.4.2.4. Detailed results of evaluating these extensions are discussed in subsequent chapters. Thereby, Chapter 5 deals with the extensions proposed in Section 4.4.2.3, while Chapter 6 covers those depicted in Sections 4.4.2.1 and 4.4.2.4. The implementation and evaluation of the provenance framework introduced in Section 4.4.2.2 is subject to future work.

[7]Eclipse BPEL Designer: `http://www.eclipse.org/bpel/`
[8]Apache ODE: `http://ode.apache.org/`
[9]Apache Tomcat: `http://tomcat.apache.org/`
[10]Apache Axis2: `http://axis.apache.org/axis2/java/core/`

4.5 Summary and Future Work

While designing simulation workflows, scientists are facing challenges related to accessing and providing huge amounts of heterogeneous data. Most existing workflow systems or other simulation tools provide some means to realize corresponding data provisioning tasks. However, scientists rarely leverage existing data provisioning techniques due to their high diversity and complexity. As a further step to reduce this diversity and complexity, this chapter derives and assesses three generic and representative data provisioning concepts from the large set of available data provisioning techniques. This helps scientists to focus on the selection of the most appropriate data provisioning concept, instead of being overstrained by the multitude of low-level options. In addition, the resulting concepts are evaluated with respect to data management patterns that typically occur in simulation workflows and also by considering relevant non-functional issues. It turns out that using local data processing in simulation workflows is an option with remarkable drawbacks and thus should be avoided whenever possible. In contrast, data services and data management activities are more often the concepts of choice. In the end, the right decision among these two concepts depends on the concrete scenario. Therefore, the results of comparing the data provisioning concepts are used in this chapter to propose and evaluate a set of guidelines for simulation workflow design. These guidelines especially assist scientists in choosing appropriate data provisioning techniques for their workflows. Another outcome of the discussion and of the guidelines is a set of essential missing features current workflow systems do not support. For instance, these workflow systems lack an abstraction support for defining data provisioning tasks that is especially tailor-made for scientists. Hence, the last contribution introduced in this chapter is an extended simulation workflow system that supports all missing features in a holistic way. This extended workflow system and the results of its evaluation are detailed in the following chapters.

Future work should mainly encompass the prototypical implementation of the provenance framework of the extended simulation workflow system depicted in Figure 4.7. This prototype may then be evaluated, especially how it enhances the reproducibility of simulations and of their outcome. Another focus of this future evaluation is to what extent such a provenance framework supports a holistic optimization of the data processing in workflows.

Chapter 5

A Framework for Accessing External Data in Workflows

The general data provisioning concept of data management activities illustrated in Section 4.1.1.2 offers a high functional flexibility regarding the support of data resources and data management operations (see also Table 4.1). However, this functional flexibility is often decreased in practice, i. e., when adopting this concept with real workflow systems. This is because these real workflow systems are usually restricted to certain kinds of data resources, e. g., SQL database systems [VSRM08], or even to proprietary data management operations [BAJ$^+$10]. As discussed in Section 4.4.1.2, computer-based simulations however require a generic solution with data management activities supporting various kinds of data resources, data formats, and data management operations [RSM14a, RSM14b]. This is especially important for multi-scale simulations that are coupled across different scientific domains, since such simulations render the data environment even more heterogeneous. Generic data management activities and an associated framework that allows workflows to uniformly access external data correspond to the second major contribution of this thesis, as discussed in Section 1.3.2.1. Figure 5.1 indicates those components of the overall SIMPL framework shown in Figure 4.7 that offer this contribution. These components are briefly described in Section 4.4.2.3, and they are detailed and assessed in this chapter:

- Firstly, Section 5.1 discusses major related work. Thereby, the main focus of this discussion is why related approaches, systems, and technologies lack a generic and consolidated solution to design data access and data provisioning tasks in simulation workflows.

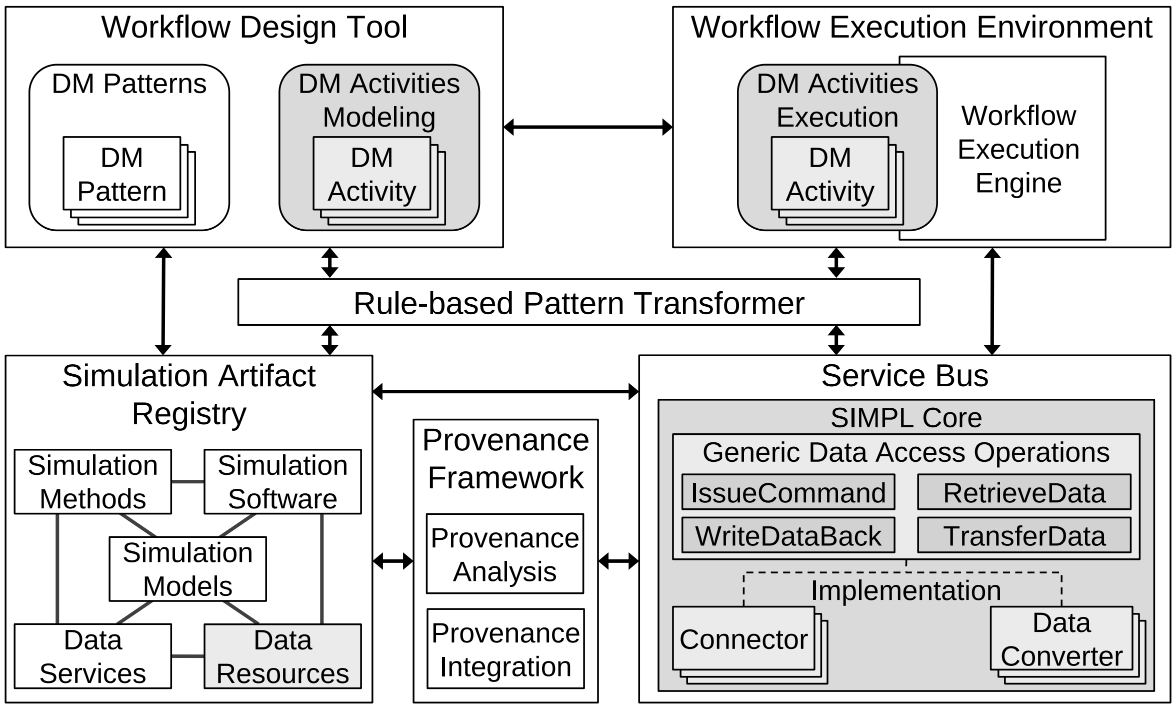


Figure 5.1: Architecture of the SIMPL framework shown in Figure 4.7. Gray-colored components are discussed in this chapter; cf. [RRS+11, RS13b, RS14].

- In Section 5.2, necessary extensions to workflow languages are discussed that address this lack of generality for data provisioning. Plug-ins integrated into the workflow design tool and workflow execution engine shown in Figure 5.1 offer such extensions as data management (DM) activities. They allow for specifying any data management operation directly within workflows that is provided by the involved external data resources in terms of their command languages, e. g., SQL, XQuery, or shell languages (see Section 2.2).
- Section 5.3 discusses how to support a uniform access to arbitrary external data resources. This is mainly provided by the SIMPL core shown in Figure 5.1 and by its generic data access operations covering the most common use cases for data access in simulation workflows. This component facilitates the design of data management activities in that it abstracts from any technical details of data access mechanisms. Furthermore, the simulation artifact registry manages metadata describing involved data resources. Regarding the uniform data access, these metadata specify the mappings between the generic operations offered by the SIMPL core and the concrete access mechanisms required by individual resources.

- The purpose of Section 5.4 is to illustrate how the contributions introduced in this chapter may be used to realize the data provisioning in concrete examples of simulation workflows. Therefore, it depicts the prototypical implementation of the gray-colored components shown in Figure 5.1. Furthermore, it discusses how to apply them to the running example of a bone simulation and to the corresponding workflows shown in Figures 4.5 and 4.6.
- Subsequently, Section 5.5 discusses the results of evaluating this prototype and its application to the bone simulation. Furthermore, it summarizes the most important implications and conclusions that have been derived when applying the prototype to other examples listed in Section 3.1. The major focus of this evaluation is whether all proposed extensions to workflow systems offer a generic solution to data access and data provisioning in simulation workflows. Nevertheless, it also considers the benefits and drawbacks of these extensions regarding the challenges discussed in Section 1.2.

Finally, Section 5.6 summarizes the major aspects and lists possible future work. This chapter is a revised version of a previous author publication [RRS$^+$11].

5.1 Related Work

The main focus of this thesis is on data provisioning in simulation workflows. Hence, the following subsections discuss major related work with respect to (1) workflow systems and their native support for data provisioning, (2) simulation data management systems, and (3) solutions to data integration or data exchange. Afterwards, Section 5.1.4 summarizes the main conclusions of this discussion.

5.1.1 Data Provisioning Support of Workflow Systems

Only a small set of today's workflow systems support the data provisioning concept of data management activities at all. To make things worse, these workflow systems are frequently restricted to certain kinds of data resources or even to specific data management operations. For instance, the solutions to data management activities offered by the workflow products of IBM, Microsoft, and Oracle are restricted to accessing SQL database systems [VSRM08]. This by far does not cover all the heterogeneous kinds of data resources required by computer-based simulations, e. g., those depicted in Tables 2.1 and 3.1.

The so-called external variables of the workflow engine Apache ODE may likewise be used to access data from SQL database systems[1]. However, they even limit the set of feasible operations to tuple-oriented retrievals and manipulations of data. This makes it necessary to implement set-oriented data management operations by embedding tuple-oriented operations into some kind of loop-based workflow constructs. Vrhovnik et al. proof that such a loop-based execution of several tuple-oriented operations shows weak performance related to the native set-oriented capabilities offered by SQL database systems [VSS$^+$07].

Kepler is one of the very few available workflow systems offering a solution to data management activities that may seamlessly access not only SQL database systems, but also file or operating systems and sensor networks [LAB$^+$06, BAJ$^+$10]. This at least constitutes a tolerably generic solution with respect to different kinds of data resources. However, the tool box for workflow design supported by Kepler contains at least one proprietary type of workflow activities – so-called actors [BL05] – for each individual kind of relevant data resources. Moreover, some of the actors are even tailored to specific proprietary data formats or data management operations. On the one hand, this may in turn restrict the generality of Kepler with respect to specific formats and operations that are not supported so far. On the other hand, the design decision relying on proprietary actors forces Kepler to offer a vast amount of diverse actors scientists may use in their workflows. As discussed in Section 1.2.1, the resulting diversity and complexity of the set of available actors usually overburdens scientists. This problem has even been admitted by some of the developers of Kepler [BAJ$^+$10].

5.1.2 Simulation Data Management Systems

Industrial applications, e. g., FEA-based applications [CMPW07], often require the consecutive or even concurrent execution of several kinds of simulations and of multiple simulation runs. This is for instance necessary to predict the behavior of different variants of a product under specific circumstances, as mentioned in Section 2.2.2. Each individual run of each relevant kind of simulation uses huge data sets as input and usually generates even bigger data sets as output [HG05]. This issue has recently convinced companies in the area of manufacturing that dedicated systems are mandatory being able to manage the vast amount of simulation data of all simulation runs. These dedicated systems are commonly

[1]Apache ODE external variables: `http://ode.apache.org/external-variables.html`

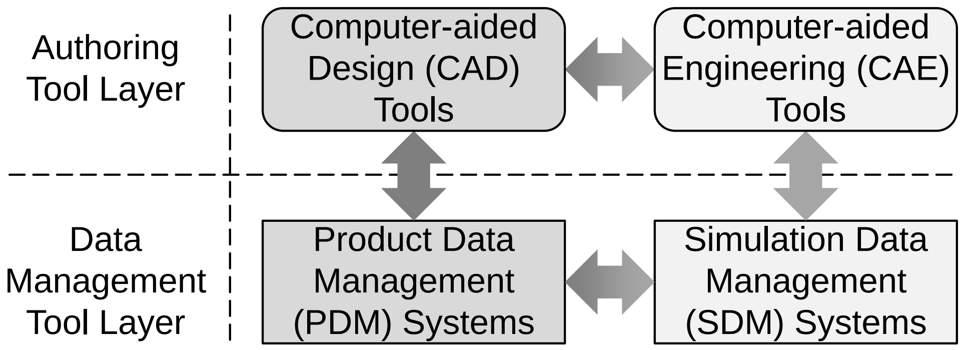

Authoring Tool Layer
Computer-aided Design (CAD) Tools
Computer-aided Engineering (CAE) Tools
Data Management Tool Layer
Product Data Management (PDM) Systems
Simulation Data Management (SDM) Systems

Figure 5.2: Simulation Data Management (SDM) Systems in the context of Product Data Management (PDM) systems and of tools for Computer-aided Design (CAD) and Computer-aided Engineering (CAE); cf. [BBF+09, vS16].

called Simulation Data Management (SDM) systems [BBF+09]. An example is the ANSYS Engineering Knowledge Manager (EKM)[2].

Figure 5.2 depicts how to embed SDM systems into the context of traditional Product Data Management (PDM) systems and authoring tools for Computer-aided Design (CAD) and Computer-aided Engineering (CAE) [BBF+09, vS16]. The purpose of PDM systems is to manage data and models of different variants of several products [Sta15]. For instance, they may manage multiple sophisticated CAD product models that have been designed using common CAD tools. CAE tools in turn do not focus on designing product models, but on investigating the characteristics of different variants of product designs [RS13a]. Such CAE tools increasingly rely on computer-based simulations to support these investigations. Hence and as shown in Figure 5.2, they likewise use SDM systems to manage the vast amount of data being used and generated during simulation processes.

The major purpose of SDM systems is to offer a systematic storage, management, archiving, and versioning of simulation data [BBF+09, vS16]. Another aspect is to facilitate the search for and re-use of such data in different simulation runs. Furthermore, SDM systems help to ensure the reproducibility of high-level simulation processes that are governed by CAE tools. This is achieved by associating individual steps of simulation processes with their respective input and output data. All this is usually based on a framework for a comprehensive metadata management to further describe the simulation data. This includes default metadata, e. g., that associate data to the steps of simulation processes. Furthermore, it encompasses possibilities to define domain-specific metadata.

[2]ANSYS EKM: `http://www.ansys.com/Products/Platform/ANSYS-EKM`

SDM systems solely rely on a document-centric management of data [BBF+09, vS16]. Hence, they only support the different file formats frequently used by computer-based simulations as summarized in Table 3.1. This is a problem that may inhibit the generic usage of SDM systems in case simulations rely on other data resources as well. For instance, they cannot be used to manage the data stored in the SQL database system that is employed by the running example of a bone simulation (see Figures 4.5 and 4.6).

Most of available SDM systems and CAE tools claim that they offer a comprehensive set of operations to realize the data provisioning for simulations. However, a detailed analysis of such systems and tools revealed that the usual set of offered operations is rather limited. This set often only consists of basic data transformations that frequently occur in a large set of prevalent industrial simulation applications. For instance, it mainly includes common conversions of default data formats for a CAD model into other default formats for a description of a finite element grid. Altogether, this does not constitute a sufficiently generic solution, which often prohibits its adoption in multi-scale simulations that are coupled across different scientific domains. Such coupled simulations usually require more complex and domain-specific data transformations than those offered by available SDM systems or CAE tools.

5.1.3 Solutions to Data Integration or Data Exchange

Federated information systems provide applications with a uniform global data schema that integrates diverse local data structures of several data resources [SL90, BKLW99]. Moreover, federated systems offer a uniform query language that applications may use to access the integrated data. The simulation applications finally accessing relevant data are usually diverse numerical simulation tools. Individual simulation tools frequently expect their input data in different formats or even in different levels of granularity. This heterogeneity of both the application and the data environment is even increased in case of multi-scale simulations that are coupled across different scientific domains. Hence, the approach of federated systems to provide a single uniform global data schema for all kinds of simulation applications is not feasible. In addition, different simulation tools rely on different mechanisms to read their input data and to write their output. So, it is likewise impractical to offer only a single uniform query language.

Table 5.1: High-level comparison of major related work enabling a peer-to-peer-like data exchange between several data resources and/or applications.

Criteria	**Schema mappings**	**Ontology-based**	**ETL technology**
Range of supported data resources / data formats	◑	●	●
Power to specify data transformation operations	◕	◕	●
Decisive drawback why an approach is usually not adopted for computer-based simulations	Not applicable to text or binary files of computer-based simulations	Only applicable to scientific domains where ontologies already exist	Multiplicity and diversity of ETL operators overwhelming scientists

A more flexible peer-to-peer-like data exchange between individual data resources and/or applications better matches the scenario of data provisioning for simulations than the approach to data integration offered by federated systems [ABLM14]. Table 5.1 summarizes the main results of comparing major related work in the area of data exchange, i. e., schema mappings, ontology-based approaches, and ETL technology. The table firstly indicates the supported range of the various data resources or data formats used in computer-based simulations (see Tables 2.1 and 3.1). The next aspect is the expressive power each individual approach offers to specify data management or data transformation operations. Furthermore, Table 5.1 illustrates the most decisive drawback of each approach why it is usually not adopted for computer-based simulations.

Schema mappings are specifications of relationships between a source data schema and a target data schema [Kol05, ABLM14]. Usually, they correspond to expressions of first-order predicate logic and some algebraic extensions that describe sophisticated structural dependencies between both schemata [JPT14]. This logical and algebraic foundation leads to a high expressive power with respect to the specification of different data transformation operations. Schema mappings may also be converted into rules or queries that finally implement these data transformation operations, e. g., based on the language SchemaSQL [LSS01]. Approaches to schema matching even enable the opposite way, i. e., deriving a set of schema mappings given specific source and target schemata [RB01].

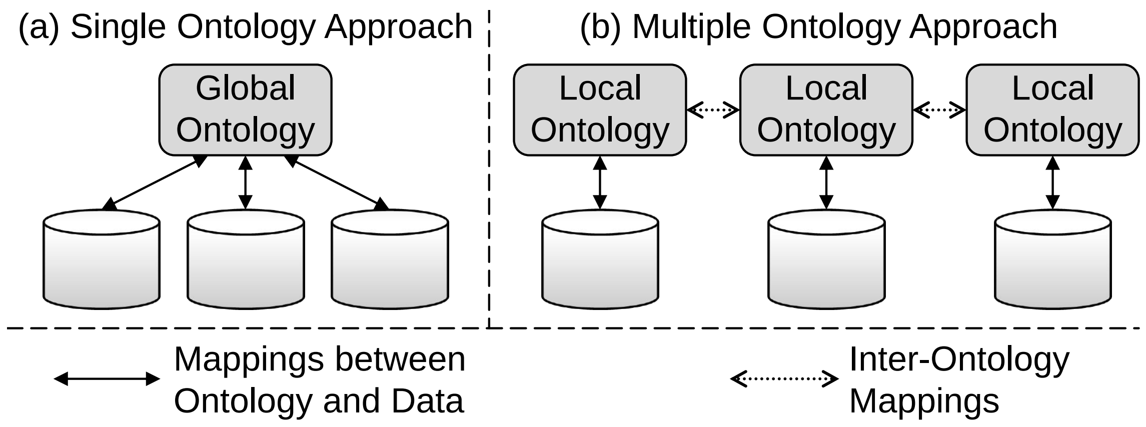

(a) Single Ontology Approach
Global Ontology
(b) Multiple Ontology Approach
Local Ontology
Local Ontology
Local Ontology
Mappings between Ontology and Data
Inter-Ontology Mappings

Figure 5.3: Ontology-based approaches to data integration or data exchange. The left side shows approaches relying on a single ontology, while approaches based on multiple ontologies are depicted on the right side; cf. [WVV+01].

However, schema mappings require the underlying data to have a well-defined structure that is somehow described explicitly – the schema. Regarding the data formats used most commonly in computer-based simulations (see Table 3.1), SQL or XML databases show this property. To some extent, this also holds for XML documents and CSV-based files. Nevertheless, schema mappings are not applicable to proprietary, unstructured text or binary files. Such text or binary files are however the most common kinds of data formats used in simulations, as discussed in Section 3.2. Hence, schema mappings do not offer a generic solution that may be seamlessly used in any simulation example.

Other solutions make use of *ontologies* that describe the intended meaning of and semantic relationships between data of different data resources [She99, WVV+01]. Some approaches rely on a single ontology that provides a global and uniform view on all relevant data resources, as depicted on the left side of Figure 5.3. As described for federated systems above, such a single uniform view on data is not sufficient to support all kinds of simulation applications with their heterogeneous requirements regarding data formats. In contrast and as shown on the right side of Figure 5.3, a more suitable peer-to-peer-like approach to data exchange employs multiple ontologies. Thereby, each data resource is individually described via a local, resource-specific ontology. Mappings between several data resources are then specified via mappings between the concepts of involved local ontologies – so-called inter-ontology mappings [WVV+01]. This approach is also used in the scientific workflow domain to describe semantic relationships between input and output data of several workflow tasks [LAG03, MCD+05].

Ontology-based approaches support a much bigger range of data resources and data formats than schema mappings, as depicted in Table 5.1. In fact, they may even be used to extract data from unstructured text files and from many other kinds of digital media [She99, WD10, KFPI11, ARR13]. The specifications of inter-ontology mappings are usually based on description logic [CGP00, WVV$^+$01]. Schema mappings typically employ first-order predicate logic and are thus more expressive than inter-ontology mappings to describe structural data dependencies. Nevertheless, ontology-based approaches add native support to resolve the semantic heterogeneity between several data resources. In summary, both approaches have their strengths and weaknesses in respectively different scenarios. All in all, they thus have a very similar expressive power with respect to the specification of data transformation operations.

A decisive drawback of ontology-based approaches is however that they often entail a high initial effort to develop ontologies describing the scientific domains of interest [WVV$^+$01]. Scientists conducting simulations would usually not accept such a high effort. Ontology-based approaches are hence restricted to those domains, where corresponding ontologies already exist. In the large area of computer-based simulations and its multiple scientific domains, ontologies are however used rather seldom in only specific kinds of applications [GCMS12]. So, they do not constitute a completely generic solution.

ETL technology encompasses various tools and frameworks that enable the design and execution of processes or pipelines for data preparation [KRRT98, LN07]. Examples are the ETL tools offered by IBM[3], Javlin[4], Pentaho[5], and Talend[6], as well as the framework Apache nifi[7]. As depicted in Table 5.1, these tools and frameworks offer the most generic solution to data provisioning and data exchange. Most of them support virtually all data resources and data formats being relevant for computer-based simulations as summarized in Tables 2.1 and 3.1. Furthermore, they typically support a wide range of data management or data transformation operations that are essential for data provisioning in simulation workflows. Some ETL tools even increase their expressive power by additionally incorporating approaches to schema mappings or ontology-based data integration or data exchange [DHW$^+$08, SSS09].

[3]InfoSphere Data Stage: `http://www-03.ibm.com/software/products/en/ibminfodata`
[4]Javlin CloverETL: `http://www.cloveretl.com/`
[5]Pentaho: `http://www.pentaho.com/`
[6]Talend: `https://www.talend.com/`
[7]Apache nifi: `https://nifi.apache.org/`

However, the flexible support of manifold data resources, data formats, and operations entails a decisive drawback that often prevents the adoption of ETL technology for computer-based simulations. ETL tools and frameworks commonly share a design decision and the corresponding negative implication with the scientific workflow system Kepler described in Section 5.1.1. They likewise offer a vast amount of diverse ETL operators that may be arranged in ETL processes or pipelines. Most of these ETL operators are special solutions to certain kinds of data formats and/or data transformation operations. The resulting multiplicity and diversity of available ETL operators again overwhelms scientists (see also Section 1.2.1). Furthermore, these scientists often have difficulties with understanding the ETL-specific meanings of individual ETL operators and with specifying corresponding low-level data transformations (see Section 1.2.2). These issues usually induce scientists not to leverage ETL technology at all.

5.1.4 Main Conclusions

Among available workflow systems discussed in Section 5.1.1, Kepler is the only one offering at least a tolerably generic solution regarding different kinds of data resources. However, Kepler usually overwhelms scientists with a multiplicity of diverse and proprietary actors that may be used to design the data provisioning in workflows. As illustrated in Section 5.1.2, SDM systems and CAE tools do not provide a generic solution at all. In fact, they are restricted to document-centric and file-based data, as well as to only a few set of data transformation operations frequently occurring in prevalent industrial applications.

Each individual solution to data integration or data exchange considered in Section 5.1.3 has a decisive drawback usually preventing its adoption for computer-based simulations (see Table 5.1). For instance, ETL technology offers a multiplicity of diverse ETL operators, which may again overwhelm scientists when designing the data provisioning in their workflows. Nevertheless, this ETL technology promises to be the most generic solution to data provisioning in simulation workflows among related work. Due to this reason and as proposed by Maier et al. [MMLW05], the SIMPL framework combines the general ideas and aspects of ETL technology with conventional workflow technology [RRS+11]. The following sections discuss how the resulting ETL workflow approach offers a generic solution to data provisioning in simulation workflows that yet does not entail the decisive drawback of ETL technology summarized in Table 5.1.

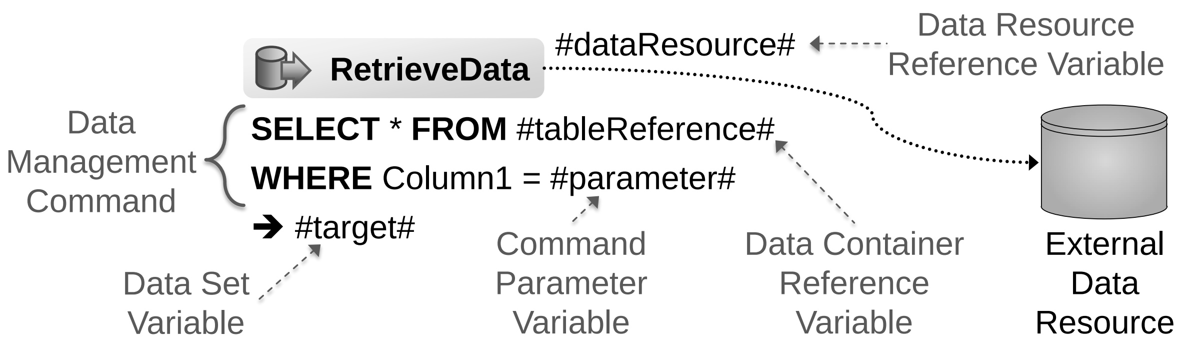

RetrieveData
#dataResource#
Data Resource Reference Variable
Data Management Command
SELECT * FROM #tableReference#
WHERE Column1 = #parameter#
➔ #target#
Data Set Variable
Command Parameter Variable
Data Container Reference Variable
External Data Resource

Figure 5.4: Design principle of the proposed data management activities exemplified by a `RetrieveData` activity and by a SQL `SELECT` statement.

5.2 Generic Workflow Language Extensions for Data Management

This thesis proposes a set of extensions to common workflow languages offering a systematic and generic solution to the data provisioning concept of data management activities summarized in Section 4.1.1.2. Section 5.2.1 discusses the basic aspects of modeling such generic data management activities in workflows. Section 5.2.2 then details the definitions of individual types of these activities.

5.2.1 Basic Modeling Aspects

The main reason why ETL technology overwhelms scientists with a multiplicity of diverse ETL operators is that each operator is usually tailored to certain kinds of data resources or even data management operations. The design principle of the generic data management activities proposed by this thesis is completely different, i. e., these activities work independently of the specifics of data resources or operations. In other words, each of these activities allows for specifying various data management operations for any kind of data resource [RRS+11]. Figure 5.4 exemplifies this design principle of generic data management activities via a `RetrieveData` activity that loads external data into the workflow context.

Each data management activity is associated with a *data resource reference variable*. Such a variable contains a logical resource descriptor that determines the external data resource finally executing the relevant data management operation. A logical resource descriptor is either a logical name or a document describing some functional and/or non-functional requirements for a data resource. The metadata managed by the simulation artifact registry of SIMPL uniquely associates a logical name with a particular data resource (see Section 5.3.3). On the contrary, a requirements description indicates the use case of dynamically selecting and binding a data resource at runtime of workflows [GSK+11, RRS+11].

Most of the proposed activities embed a *data management command* that specifies the relevant data management operation. During activity execution, the workflow engine issues this command over the SIMPL core against the involved data resource that then executes the command. Hence, the command must be a valid expression according to the command language offered by this data resource. For instance, the SQL `SELECT` statement shown in Figure 5.4 requires the data resource reference variable to point to a corresponding SQL database system.

Such an embedded data management command may contain various placeholders that are replaced by certain values during runtime of workflows. To demarcate a placeholder from the rest of the command, it is marked by surrounding hash marks (e. g., #). The first main type of such placeholders is a *data container reference variable*. This kind of variable points to a particular data container, e. g., to a database table that is used in the SQL statement shown in Figure 5.4. The content of a data container reference variable may be a resource-specific identifier of the data container, i. e., an identifier the involved data resource is able to interpret directly [HSR+10, Pie11]. An example is a combination of a schema name and table name for a SQL database table. The other option to point to a data container is again a logical name that is finally mapped to a resource-specific identifier by the simulation artifact registry (see Section 5.3.3). The second main type of placeholders is a *command parameter variable*. This is an ordinary workflow variable such as a string or an integer variable that is used as simple parameter in a data management command. For instance, the SQL statement shown in Figure 5.4 uses a parameter variable in the selection predicate to compare the variable value with the values of a specific table column.

Especially for the `RetrieveData` activity, *data set variables* act as target to store the retrieved data within the workflow context. The type definitions and workflow-internal data structures of these data set variables must account for the specifics of relevant kinds of external data resources and data formats. For instance, variables of a BPEL workflow may rely on a generic XML RowSet structure to store any tabular data, e. g., coming from a SQL database or from CSV-based files [VSRM08, Pie11]. Proprietary text files may however require application-specific XML schema definitions to store their contents in workflows.

Table 5.2: Characteristics of the proposed data management activities.

Activity	**Mandatory parameters**	**Returned result**
Issue-Command	• data resource reference variable • data management command	notification of success / failure
Retrieve-Data	• data resource reference variable • data management command / data container reference variable • data set variable	result data / notification of failure
WriteData-Back	• data set variable • data resource reference variable • data container reference variable	notification of success / failure
Transfer-Data	• data resource reference variable (source) • data management command / data container reference variable (source) • data resource reference variable (target) • data container reference variable (target)	notification of success / failure

5.2.2 Types of Data Management Activities

The above-discussed design principle of data management activities working independently of the specifics of data resources and operations entails another major benefit. It allows for providing scientists with a small set of systematic and reasonable types of data management activities. In fact, this thesis proposes four types of such activities, where each type covers a prevalent use case for data access in simulation workflows. Hence, scientists are not overwhelmed with a vast amount of diverse workflow building blocks, which is an important issue according to the challenge discussed in Section 1.2.1. This constitutes a significant advantage over ETL technology (and also over the scientific workflow system Kepler), which does not meet this challenge adequately. Table 5.2 summarizes the main characteristics of the data management activities proposed by this thesis. This includes (1) the mandatory parameters of each activity and (2) the result an activity returns to the workflow engine in case of a successful / faulty execution of the relevant data management operation.

The major purpose of the `IssueCommand` activity is data manipulation or data definition. Its first parameter is a data resource reference variable that points to the external data resource whose data shall be manipulated or where data structures need to be defined. The concrete operation for data manipulation or data definition is specified via a data management command. After the external data resource has successfully executed this command, the `IssueCommand` activity returns a notification of success to the workflow engine. The workflow engine may then continue workflow execution according to the specified control flow or dataflow. In case of a faulty command execution, a notification of failure signals the workflow engine to trigger fault handling mechanisms. Note that similar procedures are also applied to the other types of data management activities.

The `RetrieveData` activity covers the use case of loading some data from an external data resource into the workflow context. The external data resource is again determined by a data resource reference variable. The data to be retrieved may be specified via two different options. The first option is a data management command that is executed by the external data resource. One restriction is that this command must produce data, e. g., only `SELECT` statements are valid in case of a SQL database system. The second option is to use a data container reference variable. This means that the whole content of the corresponding external data container, e. g., of a database table or of a file, is loaded into the workflow context. Finally, the `RetrieveData` activity stores the produced result data into the data set variable being specified as the last parameter of the activity.

The `WriteDataBack` activity works the opposite way round, i. e., it writes data from the workflow context to an external data resource. More precisely, it writes the content of a data set variable into a particular data container, e. g., into a database table or a file, of the relevant external data resource. This scenario is reflected by the three mandatory parameters of this activity listed in Table 5.2.

The use cases reflected by the `TransferData` activity are data transfers between several external data resources, which are essential parts of the Data Transfer and Transformation Pattern depicted in Figure 3.4. So, it is the first data management activity requiring two data resource reference variables as parameters, i. e., one for the source and one for the target of the data transfer. The data to be transferred may be specified via the two options that are also valid for the `RetrieveData` activity. So, the first option is a command that is usually issued against the data source to firstly produce the relevant data before it is transferred afterwards. The second option is a data container reference variable, which indicates to transfer a whole data container, e. g., a whole file, from the source to the target. Finally, another data container reference variable specifies the concrete container where to store the transferred data within the target resource.

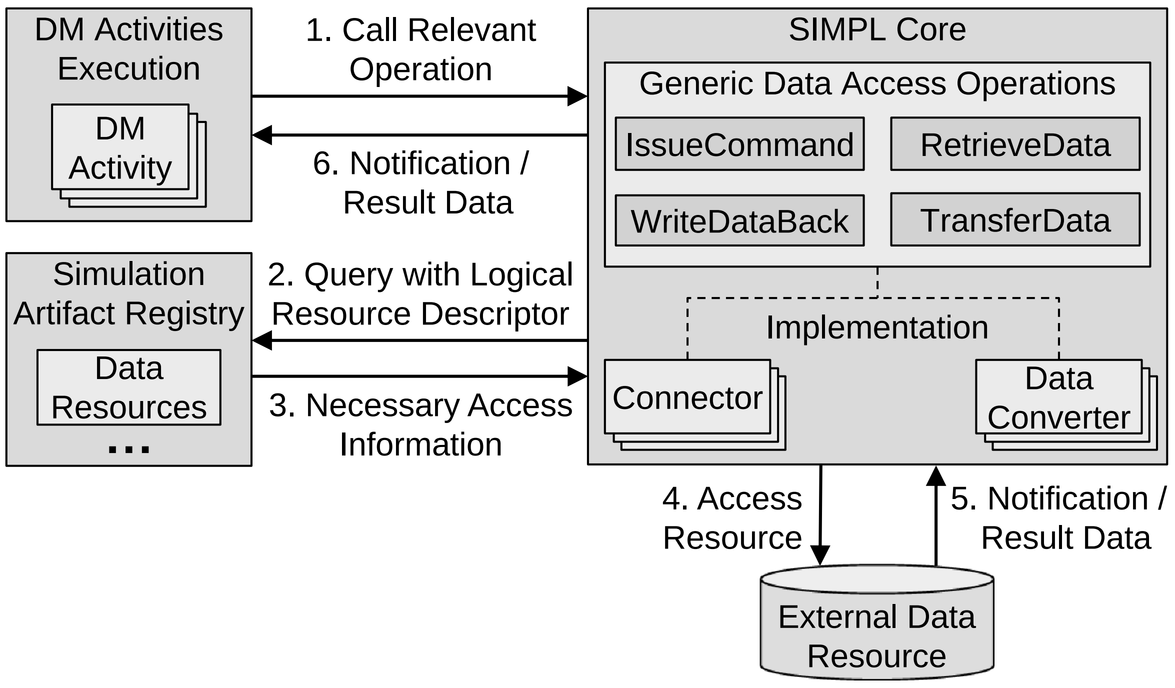

DM Activities Execution
DM Activity
1. Call Relevant Operation
6. Notification / Result Data
SIMPL Core
Generic Data Access Operations
IssueCommand
RetrieveData
WriteDataBack
TransferData
Simulation Artifact Registry
Data Resources
...
2. Query with Logical Resource Descriptor
3. Necessary Access Information
Implementation
Connector
Data Converter
4. Access Resource
5. Notification / Result Data
External Data Resource

Figure 5.5: Interaction between relevant components of the architecture shown in Figure 5.1 during data resource access; cf. [RRS+11].

5.3 Generic and Uniform Data Resource Access

The generic data management activities introduced in Section 5.2 make use of a uniform access to arbitrary data resources that is provided by the SIMPL core shown in Figure 5.1. Figure 5.5 depicts how the data management (DM) activity plug-in of the workflow execution engine, the simulation artifact registry, and the SIMPL core interact during data resource access. A data management activity firstly calls its correspondent access operation of the SIMPL core, i. e., the `IssueCommand` activity calls the `IssueCommand` operation and so on. Thereby, the activity forwards certain values of its parameters summarized in Table 5.2, e. g., a logical resource descriptor contained in a data resource reference variable. In the second step shown in Figure 5.5, the SIMPL core queries the simulation artifact registry mainly with this logical resource descriptor. The registry uses its metadata describing data resources to map the resource descriptor to all information the SIMPL core needs to access the relevant resource. This information, amongst others, consists of an URI of the resource and of identifiers of a proper connector and data converter. The registry sends all information back to the SIMPL core (step 3), which then uses it to access the data resource and to carry out the relevant data management operation (step 4). Depending on the used access operation, this data resource returns a notification of success / failure or the result data in step 5 (see also Table 5.2). The SIMPL core finally forwards this notification or result data to the data management activity in step 6.

Section 5.3.1 details the generic data access operations of the SIMPL core and how they support the individual data management activities. Afterwards, Section 5.3.2 discusses how to implement these generic access operations for different data resources via connectors and data converters. The focus of Section 5.3.3 is on the metadata of the simulation artifact registry mapping logical resource descriptors to concrete information that describes how to access a resource.

5.3.1 Generic Data Access Operations

The data access operations offered by the SIMPL core share the basic design principle of the data management activities introduced in Section 5.2. So, the specifications of these SIMPL core operations are especially independent of any characteristics of underlying data resources. This facilitates generic and lightweight SIMPL core operations that allow for accessing arbitrary external data resources. Table 5.3 illustrates the parameters the respective SIMPL core operations expect as input from their correspondent data management activities and their parameters shown in Table 5.2.[8]

To identify the data resource(s) to be accessed, each SIMPL core operation expects either one or two logical resource descriptors as input. Such a logical resource descriptor corresponds to the content of the relevant data resource reference variable of an associated data management activity (see Section 5.2.1). The data container references mentioned in Table 5.3 are likewise the contents of the corresponding data container reference variables depicted in Table 5.2. So, they are either a logical name or a resource-specific identifier of the container. Data management commands are in contrast sent from a data management activity to its SIMPL core operation as they are.

The `RetrieveData` operation moreover expects a description of the workflow-internal data type of the data set variable into which the `RetrieveData` activity finally stores the data. This is necessary to identify a proper data converter that is able to convert the format of the relevant external data resource or

[8]Note that the results the SIMPL core operations return to data management activities are not shown in Table 5.3, because they are exactly the same as those depicted in Table 5.2.

Table 5.3: Parameters the SIMPL core operations expect from correspondent data management activities; see Table 5.2 for the parameters of the activities.

Operation	**Expected input parameters**
Issue-Command	• logical resource descriptor • data management command
Retrieve-Data	• logical resource descriptor • data management command / data container reference • workflow-internal data type of data set variable
WriteData-Back	• content and data type of data set variable • logical resource descriptor • data container reference
Transfer-Data	• logical resource descriptor (source) • data management command / data container reference (source) • logical resource descriptor (target) • data container reference (target)

data container into the workflow-internal data type (see Section 5.3.3). The `WriteDataBack` operation not only gets the workflow-internal data type of the relevant data set variable as input, but also the whole content of this variable. It then stores this variable content into the specified external data container.

5.3.2 Implementation of the SIMPL Core Operations

Different kinds of data resources often rely on different access mechanisms. Hence, the generic data access operations of the SIMPL core have to be implemented in likewise different ways for concrete data resources or types of resources. As shown in Figure 5.5, *connectors* provide this implementation and account for the specifics of relevant data resources. For instance, one connector may support all kinds of file or operating systems that offer a shell interface based on SSH [Pie12]. Pietranek shows how to realize one connector that supports any database system relying on the Java Database Connectivity (JDBC)[9] API as access mechanism, i. e., most of available SQL database systems [Pie11]. Different XML database systems or gateways to sensor networks usually rely on proprietary access mechanisms or APIs and thus require likewise specific connectors [Pie11].

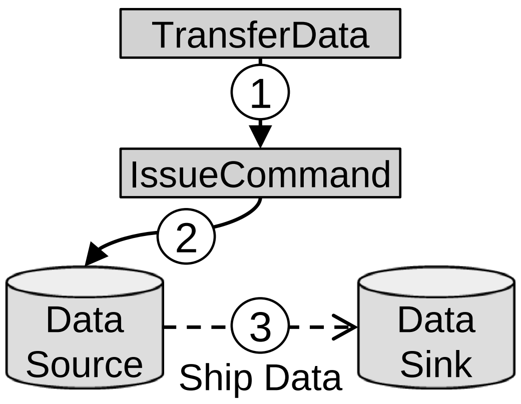
[9]JDBC API: `http://www.oracle.com/technetwork/java/javase/jdbc/index.html`

Some data resources do not support each individual SIMPL core operation. For instance, gateways to sensor networks usually do not allow applications to write data back as they are only able to deliver data (see Section 2.2.4). In such a case, the corresponding connectors do not provide these operations as well. Furthermore, scientists then also must not use the associated data management activities in their workflows. For that purpose, the workflow design tool needs access to some information provided by the simulation artifact registry describing which operations are valid for which (types of) data resources.

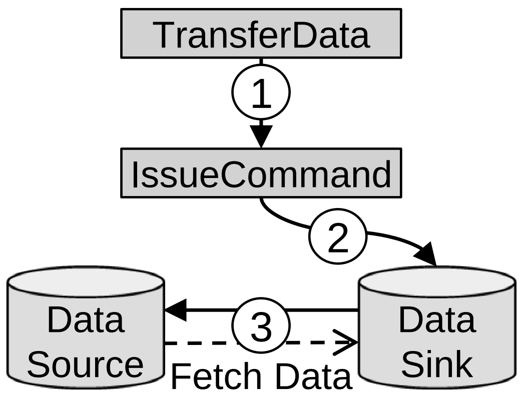
The SIMPL core additionally provides *data converters* that are especially important for the `RetrieveData` and `WriteDataBack` operations. They convert data between a format that is specific for external data resources or their connectors and a format that is tailored to store data within the workflow. For instance, one data converter transforms data between the JDBC result set format used for SQL database systems and the XML RowSet format mentioned in Section 5.2.1 [HSR$^+$10]. Other converters support certain kinds of file formats or the formats used in XML database systems or gateways to sensor networks [Pie11].

The `TransferData` operation features the highest operational complexity and thus entails the biggest effort to implement it in connectors and data converters. To reduce this effort, the implementation of this operation in the SIMPL core also uses the functionality of the other SIMPL core operations for parts of a data transfer. The `IssueCommand`, `RetrieveData`, or `WriteDataBack` operations facilitate (parts of) data transfers in various ways, as depicted in Figure 5.6.

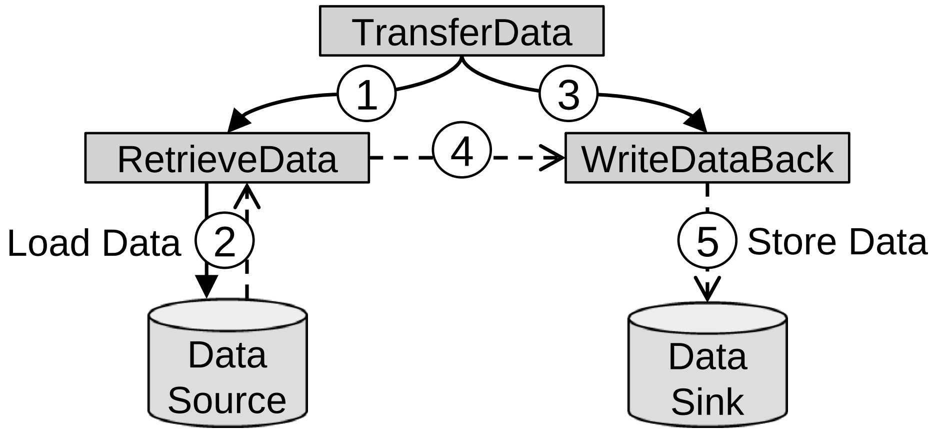
Firstly, an `IssueCommand` operation may initiate a direct data transfer between the resources. Figure 5.6a shows a variant, where the `TransferData` operation calls an `IssueCommand` operation accessing the data source. This data source then produces the relevant data and ships it to the data sink. In the variant shown in Figure 5.6b, the `IssueCommand` operation accesses the data sink. So, this data sink is the initiator of the data transfer and fetches the data from the data source. In both cases, the `TransferData` operation may need to adjust the command it gets as input before forwarding it to the `IssueCommand` operation. This may be necessary in order that the final command that is issued against either the data source or data sink describes a valid data transfer. For instance, assume that the `TransferData` operation gets a SQL `SELECT` statement as input and that the result of this statement shall be copied from the SQL database to a local file. Here, the `TransferData` operation needs to embed this `SELECT` statement into a proper `EXPORT` statement before forwarding it to the `IssueCommand` operation.



(a) `IssueCommand` operation accessing the data source.



(b) `IssueCommand` operation accessing the data sink.



(c) Combination of a `RetrieveData` and a `WriteDataBack` operation.

Figure 5.6: Major variants of implementing the `TransferData` operation based on other SIMPL core operations. Solid lines represent a function call or function shipping, e. g., by issuing a command. Dashed lines represent a dataflow or data shipping between resources or operations.

The third way to realize the `TransferData` operation is a combination of a `RetrieveData` and `WriteDataBack` operation, as shown in Figure 5.6c. Here, the `TransferData` operation firstly sends the command producing the relevant data to a `RetrieveData` operation. This operation then issues this command against the data source in order to intermediary load the data within its operation context. Note that, for efficiency reasons, the data is thereby not stored in a data set variable of a workflow. Instead, it is stored in the object space of the SIMPL core or in another data cache of the workflow system. Afterwards, the `TransferData` operation instructs a `WriteDataBack` operation to access the cached data and to store it in the proper data container of the data sink.

Not every data resource supports all variants to realize the `TransferData` operation shown in Figure 5.6. For instance, the variants using an `IssueCommand` operation require the data source and/or data sink to support the relevant kind of a direct transfer (see Section 4.2.2.1). Furthermore, the proper choice depends on non-functional aspects as well, e. g., on efficiency issues. Therefore, the SIMPL core queries the simulation artifact registry and its metadata about data resources for the variant that is the best available one for a given pair of data resources.

5.3.3 Metadata for Mappings to Access Mechanisms

The simulation artifact registry further facilitates designing data management activities in that it manages metadata describing all available data resources that may be assigned to the activities. Furthermore and as shown in Figure 5.5, it provides the SIMPL core with all information that is necessary to access the data resources. This also includes information about associated connectors and data converters, as well as about specific data containers managed by the resources. Figure 5.7 offers a high-level view on the most important metadata.

A *logical resource name* is unique for each data resource and acts as its major identifier within the SIMPL framework. As discussed in Section 5.2.1, it may be used as logical resource descriptor in data resource reference variables of data management activities. As alternative, such a logical resource descriptor in workflows may be a requirements description indicating the use case of selecting a data resource at runtime of workflows. This is where the *properties description* comes into play, which describes functional and non-functional characteristics of a data resource. So, such properties descriptions are matched against the requirements description during runtime in order to identify a proper resource. In a similar way, this facilitates the choice among different variants of realizing the `TransferData` operation depicted in Figure 5.6. Additional important information about a data resource are an *endpoint* to access the resource and a set of *security credentials*, e. g., user names and passwords or authentication keys.

The simulation artifact registry may be used to associate a data resource with its most important data containers, e. g., containers that are frequently accessed by several workflows. Thereby, a registered data container is again associated with a *logical name* that may be used as reference in a workflow. The simulation artifact registry maps this logical name to a *local, resource-specific container identifier* the involved data resource is able to interpret. For instance, operating

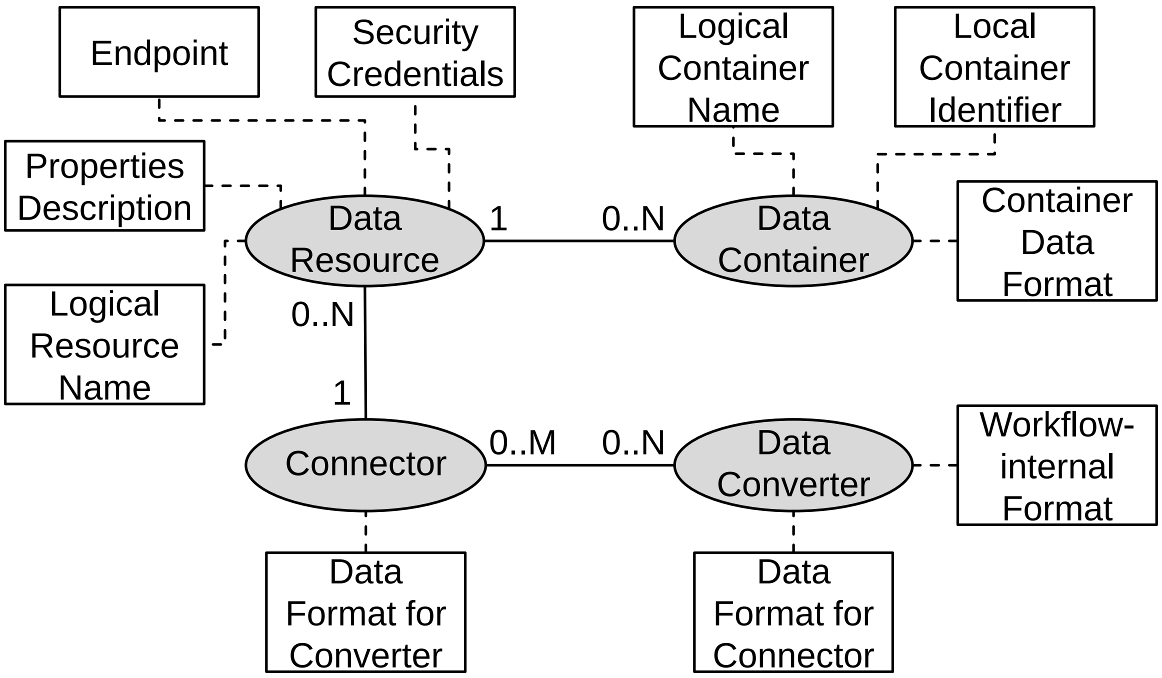


Figure 5.7: High-level view on metadata describing all information that is necessary to access data resources; cf. [RRS+11].

systems assume a combination of a file name and a directory name as resource-specific container identifier. In addition, the metadata about a data container includes a description of its *data format*, e. g., of a particular file format.

As described in Section 5.3.2, one connector may implement the SIMPL core operations for one or even for multiple data resources. Furthermore, there may be several data converters transforming data between different external or connector-specific data formats and different workflow-internal formats. The simulation artifact registry associates a connector with a *data format for a converter*, i. e., a format in which the connector delivers output data to a converter and expects input data from it. A data converter is analogously associated with a *data format for a connector*. Thereby, only the connectors and data converters are assigned to each other that rely on the same data format. In addition, a data converter is associated with the *workflow-internal data format* it supports. As illustrated in Section 5.3.1, the `RetrieveData` and `WriteDataBack` operations also query the simulation artifact registry with a specification of the required workflow-internal data format. The registry then uses all individual descriptions of data formats being associated with connectors, data converters, and also data containers (see Figure 5.7) to select a data converter that offers the proper format conversion.

5.4 Prototype and its Application to the Bone Simulation

The prototype of the system components highlighted via gray color in Figure 5.1 has been developed in the course of various student projects that have accompanied the work on this thesis [HSR+10, Pie11, Pie12]. As discussed in Section 4.4.3, it is based on the workflow language BPEL [OAS07] and on a BPEL-based system environment. The plug-in of the workflow design tool shown in Figure 5.1 extends the Eclipse BPEL Designer version 0.8.0 by all activity types summarized in Table 5.2. The plug-in of the workflow execution environment correspondingly extends Apache ODE version 1.3.5. This is based on a framework enabling the seamless extension of BPEL engines by additional activity types [KKL07]. The SIMPL core and its generic data access operations are implemented as Java-based Web Service, which is deployed on Apache Tomcat version 7.0 using Axis2 version 1.5.4. The same holds for the simulation artifact registry and its interface to access relevant metadata. Thereby, the metadata describing data resources are managed using a PostgreSQL version 9.2 database system.

This prototype has been used to realize the data provisioning in concrete examples of simulation workflows, e. g., in most examples listed in Section 3.1 [HSR+10, Pie11, Pie12, Rie14, Deh15, Kal15, vS15b]. In analogy to the reasons discussed in Section 4.3.2, the running example of a bone simulation is again well-suited to illustrate the major aspects of applying the prototype to real simulations. In the following subsections, this illustration is split according to the most challenging scenarios also considered in Sections 3.2 and 4.3.2.

5.4.1 Provisioning of Input Data for Single Simulations

Figure 5.8 depicts how the data management activities of SIMPL may be used to realize the data provisioning steps in the bio-mechanical simulation workflow shown in Figure 4.5. The original data service implementing the file transfer in workflow step *Transfer Input File* is even replaced by a `TransferData` activity to show the potential of the proposed data management activities. This is a negligible violation of guideline 2 proposed in Section 4.3.1.2, which actually recommends to prefer data services because they usually offer a better abstraction support to scientists. In fact, the data service used in this example and the `TransferData` activity require scientists to specify the same input parameters.

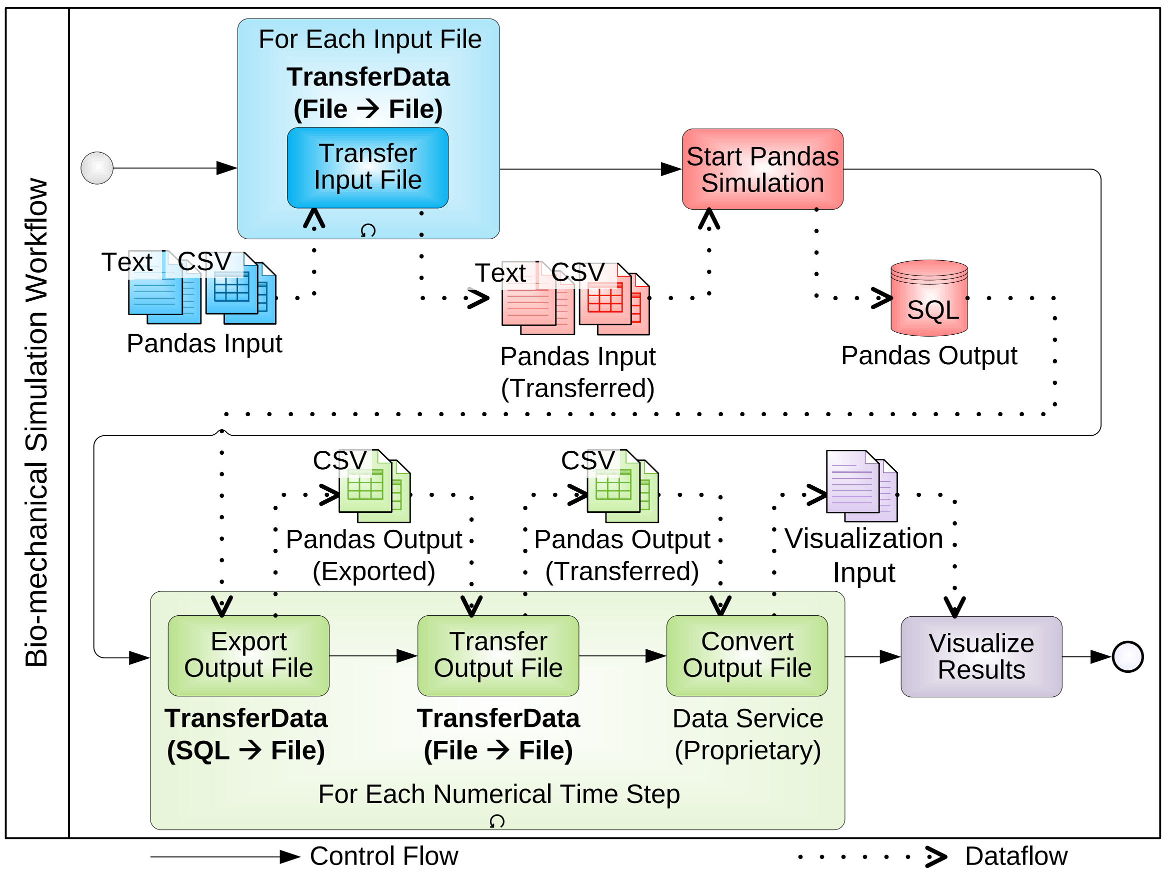


Figure 5.8: Usage of data management activities in the bio-mechanical simulation workflow shown in Figure 4.5. Relevant workflow steps and the respectively used kinds of data management activities are highlighted via bold labels.

These parameters are references to (1) the source computer, (2) the source file, (3) the target computer, and (4) the target directory where to copy the file to. So, both kinds of data provisioning techniques entail the same effort scientists have to spend on workflow design. In this case, the `TransferData` activity is preferred, as it is more flexible regarding the support of different kinds of data transfers (see Figure 5.6). Its implementation in the `TransferData` operation of the SIMPL core embeds all above-listed parameters of the activity into an appropriate Secure Copy (SCP) command. The `TransferData` operation then uses the variant shown in Figure 5.6a, i. e., it sends the SCP command to the `IssueCommand` operation. The `IssueCommand` operation issues the SCP command against the source computer in order to finally ship the data to the target computer.

The workflow step *Export Output File* is realized by another `TransferData` activity. Its data source is specified via a reference to the SQL database of

Pandas and via a `SELECT` statement as embedded data management command. Furthermore, the `TransferData` activity specifies that the result data produced by this `SELECT` statement shall be stored in a CSV-based file in the local operating system. The `TransferData` operation of the SIMPL core finally embeds the `SELECT` statement into a proper SQL `EXPORT` statement. It then again uses the variant shown in Figure 5.6a and issues the `EXPORT` statement over the `IssueCommand` operation against the SQL database of Pandas.

The next workflow step *Transfer Output File* copies the exported CSV-based file to another computer. This may again be realized by a `TransferData` activity specifying similar parameters and being analogously implemented as the one in workflow step *Transfer Input File*. Finally, the workflow step *Convert Output File* is still implemented by a proprietary data service, and not by a data management activity. This is because none of the involved data resources and thus also no data management activity offer the required functionality (see Section 4.3.2.1).

5.4.2 Data Exchange between Different Simulations

Figure 5.9 correspondingly depicts how the data management activities of SIMPL may be used in the coupling workflow shown in Figure 4.6. Here, none of the original data services or local data processing steps are replaced by data management activities, as this would really be a severe violation of guidelines 1 and 2 proposed in Section 4.3.1. The first workflow step realized by a data management activity is thus the one called "*Load # of Spatial Points*" in Figure 5.9. It loads the total number of spatial evaluation points for which Pandas has stored its outcome in its database. Hence, a `RetrieveData` activity embedding a SQL `SELECT` statement is the choice to realize this data load into the workflow context.

The workflow step *Export Octave Input Files* accesses the database of Pandas to export relevant parts of the bio-mechanical simulation outcome into a CSV-based file. This data export is again realized by a `TransferData` activity that is parameterized and implemented in a similar way as the `TransferData` activity in workflow step *Export Output File* shown in Figure 5.8. So, the activity specifies a SQL `SELECT` statement, and the `TransferData` operation finally embeds this `SELECT` statement into a proper `EXPORT` statement, which is then issued against the data source, i. e., the database of Pandas. The only difference is that the `SELECT` statement used in the coupling workflow is much more complex than the one used in the bio-mechanical simulation workflow (see also Listing 5.1).

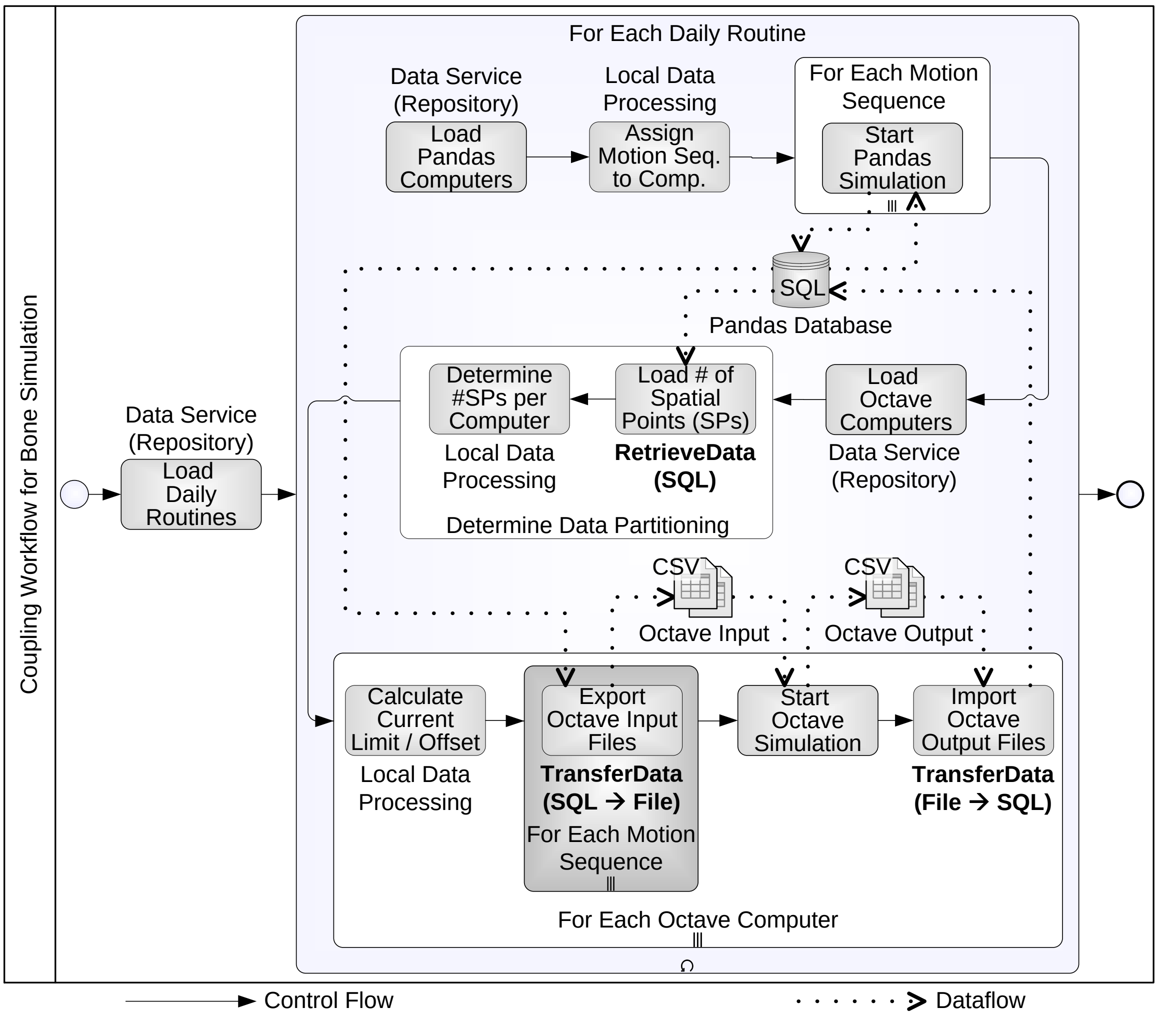

Coupling Workflow for Bone Simulation
Data Service (Repository)
Load Daily Routines
For Each Daily Routine
Data Service (Repository)
Load Pandas Computers
Local Data Processing
Assign Motion Seq. to Comp.
For Each Motion Sequence
Start Pandas Simulation
SQL
Pandas Database
Determine #SPs per Computer
Local Data Processing
Load # of Spatial Points (SPs)
RetrieveData (SQL)
Load Octave Computers
Data Service (Repository)
Determine Data Partitioning
CSV
Octave Input
CSV
Octave Output
Calculate Current Limit / Offset
Local Data Processing
Export Octave Input Files
TransferData (SQL → File)
For Each Motion Sequence
Start Octave Simulation
Import Octave Output Files
TransferData (File → SQL)
For Each Octave Computer
Control Flow
Dataflow

Figure 5.9: Usage of data management activities in the coupling workflow shown in Figure 4.6. Relevant workflow steps and the respectively used kinds of data management activities are highlighted via bold labels; cf. [RSM14b].

The workflow step *Import Octave Output Files* works exactly the opposite way round, i. e., it imports a CSV-based file into the SQL database of Pandas. Nevertheless, it may again be realized by a `TransferData` activity. Its parameters define the data transfer via references to (1) the source operating system, (2) the input CSV-based file, (3) the target SQL database system of Pandas, and (4) the target database table. This time, the `TransferData` operation of the SIMPL core adopts the variant shown in Figure 5.6b to realize this file import. So, it uses the parameters of the activity to generate a proper SQL `IMPORT` statement. It then issues this `IMPORT` statement over the `IssueCommand` operation against the database of Pandas, which is specified as the data sink of this data transfer.

5.5 Evaluation

The prototypical implementation illustrated in Section 5.4 and its application to example simulations listed in Section 3.1 has been the basis for a profound evaluation of the data management activities proposed by this thesis. Table 5.4 summarizes the most important results of this evaluation regarding the primary goals of these data management activities. Furthermore, it correspondingly compares the data management activities of SIMPL with their major competitors of related work discussed in Section 5.1.

As discussed in Section 4.4.1.2, the first primary goal of the data management activities of SIMPL is to offer a generic solution that is sufficient for a majority of simulation examples of any scientific domain. More precisely, the activities have to support various kinds of data resources, data formats, and data management operations. This aspect is covered by the first two rows of Table 5.4 and discussed in Section 5.5.1. This primary goal likewise determines the major competitors of SIMPL among related work. Regarding existing workflow systems discussed in Section 5.1.1, Kepler is the only one offering at least a tolerably generic solution. Furthermore, ETL technology offers the most generic approach among the solutions to data integration and data exchange analyzed in Section 5.1.3. Note that SDM systems and CAE tools illustrated in Section 5.1.2 do not provide a generic solution at all and are thus not considered in Table 5.4.

The other primary goal of the data management activities of SIMPL is covered by the third row of Table 5.4 and discussed in Section 5.5.2. It is a direct implication of the selection of the major competitors Kepler and ETL technology. Both of them have the same decisive drawback often preventing their adoption for computer-based simulations. They usually overwhelm scientists with a multiplicity of diverse workflow actors or ETL operators that may be used to design the data provisioning in workflows. Hence, another primary goal of SIMPL is to provide scientists with only a small and reasonable set of different kinds of workflow building blocks, i. e., data management activities. This is also one of the main conclusions of discussing related work, as summarized in Section 5.1.4. Furthermore note that this goal is closely related to the challenge regarding the diversity of available data provisioning techniques discussed in Section 1.2.1.

The remaining Sections 5.5.3 to 5.5.5 accordingly discuss the major benefits or drawbacks of the data management activities offered by SIMPL regarding the challenges illustrated in Sections 1.2.2 to 1.2.5.

Table 5.4: Most important results of comparing the data management (DM) activities offered by SIMPL with the major competitors of related work.

Criteria / Goal	**Kepler**	**ETL technology**	**SIMPL DM activities**
Range of supported data resources / data formats	◕	●	●
Power to specify data transformation operations	◑	●	◕
Small set of workflow building blocks	◔	◔	●

5.5.1 Generic Data Management in Workflows

As shown in Table 5.4, the data management activities of SIMPL support a broad range of data resources and data formats. Furthermore, they offer a high expressive power to specify data management or data transformation operations. These aspects are respectively discussed in the following two subsections.

5.5.1.1 Range of Supported Data Resources and Data Formats

The current prototype of the SIMPL core offers several connectors that allow workflows to access various kinds of external data resources [HSR$^+$10, Pie11, Pie12]. This includes any Unix-based and Windows file or operating system, regardless of whether a workflow accesses it locally or remotely. Furthermore, workflows may use SIMPL to seamlessly access any SQL database system offering a JDBC API. The current prototype also supports several proprietary XML database systems, as well as TinyDB mentioned as gateway to sensor networks in Section 2.2.4 [Pie11]. In summary, the prototype covers virtually all kinds of data resources that are commonly used in computer-based simulations, as summarized in Tables 2.1 and 3.1. In particular, it supports each individual data resource that is used in the example simulations listed in Section 3.1 [HSR$^+$10, Pie11, Pie12, Rie14, Deh15, Kal15, vS15b]. Furthermore, it offers a broad set of data converters supporting various data formats used in relevant `RetrieveData` and `WriteDataBack` operations of these example simulations (see also Section 5.3.2).

Over and above, the plug-in mechanism of the SIMPL core facilitates a seamless extension by new connectors or data converters supporting additional kinds of data resources or data formats. Pietranek discusses that it is straightforward to implement new connectors or data converters [Pie11, Pie12]. In particular, the `IssueCommand`, `RetrieveData`, and `WriteDataBack` operations are very lightweight and thus entail only a low implementation overhead. As discussed in Section 5.3.2 and depicted in Figure 5.6, the more complex `TransferData` operation makes use of the three other SIMPL core operations to realize most parts of a data transfer. So, the remaining application logic to be implemented directly in the `TransferData` operation is usually straightforward as well.

Altogether and as depicted in Table 5.4, SIMPL supports a comparable range of data resources and data formats as ETL technology. This is especially true for the resources and formats that are commonly used in computer-based simulations (see Tables 2.1 and 3.1). The straightforward possibility to extend SIMPL to support additional kinds of data resources and data formats is another significant advantage. It is the main reason why SIMPL is rated better in Table 5.4 than the workflow system Kepler. In fact, Kepler usually requires implementing whole new workflow actors in order to add support for additional data resources or data formats. These new actors often not only have to deal with the respective technical details of data access mechanisms, but also with low-level details of more complex data management operations. This may result in a remarkable implementation overhead when additional resources or formats have to be supported.

5.5.1.2 Power to Specify Data Transformation Operations

The SIMPL core operations and the data management activities support the most relevant use cases for data access, i. e., data manipulation, data definition, data retrieval, writing data back to external resources, and data transfers. These use cases for data access are common and sufficient for virtually all data provisioning scenarios in the majority of existing simulation applications. As shown in Figures 5.8 and 5.9, they may realize nearly all data provisioning steps in the workflows of the bone simulation – except for those steps where the guidelines proposed in Section 4.3.1 strictly recommend to use another data provisioning concept than data management activities. This has also been confirmed when applying SIMPL and the data management activities to other simulation examples listed in Section 3.1 [HSR+10, Pie11, Pie12, Rie14, Deh15, Kal15, vS15b].

As additional benefit, the data management activities allow for specifying a wide range of sophisticated data management or data transformation operations.

In fact, they support any operation that may be specified as a valid expression of the command languages offered by involved external data resources. For instance, the command languages SQL, XQuery, or XPath offered by relational or XML database systems are highly expressive and support various operations (see Table 2.1). Nevertheless, shell languages of file or operating systems typically offer only a moderate power to specify complex data transformation operations.

In contrast to the data management activities of SIMPL, the corresponding actors offered by Kepler are often tailored to specific operations. This may even restrict the functionality offered by involved data resources via their command languages. Hence and as depicted in Table 5.4, Kepler offers a less expressive power to specify data transformation operations than SIMPL. On the contrary, ETL technology may in certain cases support more features than the data management activities of SIMPL. This is especially true when file or operating systems are involved, whose command languages are often not intended for very complex data transformation operations. Here, ETL technology and corresponding tools may additionally incorporate proprietary solutions that extend the set of operations supported by involved data resources. For instance, GeoKettle[10] represents such an extension adding support for processing spatial data to the ETL tool of Pentaho. Pietranek proposes to correspondingly augment SIMPL with additional operations by integrating complementary data transformation services with the service bus shown in Figure 5.1 [Pie12].

5.5.2 Diversity of Available Data Provisioning Techniques

As shown in Figure 2.7, many of the common phases of a simulation workflow are usually classified as orchestration workflows. This especially concerns the platform and software provisioning phases, and in many cases even the calculation phase [GSK+11, VHKL13]. In contrast to Kepler, ETL technology, and most of other related work, SIMPL relies on this conventional orchestration workflow technology as well [MMLW05], e. g., its prototype is based on the workflow language BPEL. Hence, SIMPL and its data management activities allow for designing both the data provisioning phases of a simulation workflow and most of its other phases at the same level of abstraction, i. e., at the level of orchestration workflows. This leads to a seamless design environment that removes the burden from scientists to get accustomed to many diverse design tools or technologies.

[10]GeoKettle: `http://www.spatialytics.org/projects/geokettle/`

As another unique selling point, SIMPL provides scientists with a small and reasonable set of different kinds of workflow building blocks. In fact, scientists are faced with only four different types of data management activities, as summarized in Table 5.2. Furthermore, scientists are often quite familiar with the abstract meanings of the above-mentioned use cases for data access that are covered by the data management activities. Altogether and as depicted in Table 5.4, this constitutes a decisive advantage of SIMPL over Kepler and ETL technology, which usually overwhelm scientists with a multiplicity of diverse workflow building blocks. It is one of the main reasons why SIMPL has so far been adopted in more simulation examples than related work, e. g., in most examples listed in Section 3.1 [HSR+10, Pie11, Pie12, Rie14, Deh15, Kal15, vS15b].

5.5.3 Multiplicity and Complexity of Low-Level Data Management Operations

The data management activities of SIMPL allow scientists to define data management operations without being forced to provide any technical details of data access mechanisms. Instead, these technical details are covered by the SIMPL core and its connectors and data converters. However, scientists still have to specify many complex data management commands in `IssueCommand`, `RetrieveData`, and `TransferData` activities. Furthermore, they are often not familiar with the command languages offered by involved data resources, especially with SQL, XPath, or XQuery. So, they have to spend a considerably high effort to design corresponding data provisioning tasks [RLSR+06, RS14, RSM14b]. In some cases, the complexity of data management commands even entails that scientists are not able to specify them at all. This is for instance the case for the SQL `SELECT` statement embedded in the `TransferData` activity *Export Octave Input Files* of the coupling workflow shown in Figure 5.9. This `SELECT` statement has actually been specified by experts of this query language [Ges14, RSM14b].

Listing 5.1 illustrates the complexity of the corresponding `SELECT` statement. It contains three sub-queries, i. e., one for each relevant mathematical variable to be exchanged between the bio-mechanical and systems-biological calculations [Kra14]. Each of these sub-queries specifies two selection predicates: one for the relevant variable and one for the current motion sequence of the outer loop of the `TransferData` activity (see Figure 5.9). Furthermore, each sub-query contains an average function and an associated `GROUP BY` clause to carry out the

```
1  SELECT A.elementID, A.evalpointID, NSTS_avg, SIG_V_avg, CNUF_avg
2  FROM
3    ( SELECT elementID, evalpointID, AVG(value) AS NSTS_avg
4    FROM pandas_output
5    WHERE variable = 'NSTS' AND motionsequence = @CURMOTIONSEQ
6    GROUP BY elementID, evalpointID
7    ) AS A,
8
9    ( SELECT elementID, evalpointID, AVG(value) AS SIG_V_avg
10   FROM pandas_output
11   WHERE variable = 'SIG_V' AND motionsequence = @CURMOTIONSEQ
12   GROUP BY elementID, evalpointID
13   ) AS B,
14
15   ( SELECT elementID, evalpointID, AVG(value) AS CNUF_avg
16   FROM pandas_output
17   WHERE variable = 'CNUF' AND motionsequence = @CURMOTIONSEQ
18   GROUP BY elementID, evalpointID
19   ) AS C
20
21 WHERE A.elementID = B.elementID AND A.evalpointID = B.evalpointID
22   AND B.elementID = C.elementID AND B.evalpointID = C.evalpointID
23 ORDER BY elementID, evalpointID
24 LIMIT = @CURLIMIT OFFSET = @CUROFFSET
```

Listing 5.1: SQL statement embedded in the `TransferData` activity *Export Octave Input Files* of the coupling workflow shown in Figure 5.9; cf. [Ges14].

data aggregations among all numerical time steps. The join predicates in lines 21 and 22 of Listing 5.1 combine the individual results of the sub-queries into one result set. The `ORDER BY`, `LIMIT`, and `OFFSET` clauses in lines 23 and 24 are used to properly partition the data among available Octave computers.

Altogether, this calls for an abstraction support for specifying low-level data management operations in simulation workflows. As discussed in Section 4.4.1.2, none of today's workflow systems offers such an abstraction support that is particularly tailor-made for scientists conduction simulations. This thesis addresses this problem and proposes an adequate approach in Chapter 6 that completely removes the burden from scientists to specify any low-level details of data management or data provisioning in their workflows [RS14, RSM14a, RSM14b].

5.5.4 Efficient Data Processing and Optimization

The data management activities of SIMPL treat descriptions of data management operations as integral parts of workflow models. Hence and as discussed in Section 4.2.3.1, a workflow system may analyze these workflow models to get much information about the data management of a workflow. It may then use this information to properly apply optimization techniques that increase the efficiency of data processing in simulation workflows [BHW$^+$07, VSS$^+$07, OdOV$^+$11].

For instance, Vrhovnik et al. propose an optimization approach that uses this enhanced information to properly re-structure workflow models with embedded SQL statements [VSS$^+$07]. So, this approach makes workflows more efficient, e. g., by re-arranging the order of such SQL statements within the workflow model or by merging several statements together. Kalyoncu discusses how to apply this approach to simulation workflows that make use of the data management activities of SIMPL [Kal15]. Furthermore, he investigates additional optimization scenarios, where workflows not only access SQL database systems, but also XML database systems or even files. Altogether, this offers a huge potential to induce significant performance improvements for simulation workflows [VSS$^+$07, Kal15].

MapReduce is a scalable approach that increases the efficiency of data-intensive applications by enabling a massive parallelization of corresponding data processing tasks [DG08]. Gessler shows that it is especially suited to accelerate the filtering, aggregation and partitioning of data carried out in the coupling workflow shown in Figure 5.9 [Ges14]. SIMPL and its data management activities may seamlessly access and exploit certain MapReduce-based systems. This holds for those systems that offer some kind of command language, as well as an API or CLI over which commands may be issued against the systems. For instance, Apache Hive[11] and Spark SQL[12] offer this opportunity. They even provide a JDBC API, which makes it possible to use the JDBC-based connector that is already implemented in the current prototype of SIMPL (see Section 5.5.1.1).

Cloud computing technologies are key enablers for elastic and efficient data-intensive applications [HKR13, LQ14]. The OASIS standard Topology and Orchestration Specification for Cloud Applications (TOSCA) is a representative solution to automate platform and software provisioning tasks for cloud-based applications [OAS13]. TOSCA has already been applied successfully to simula-

[11]Apache Hive TM: `https://hive.apache.org/`
[12]Apache Spark SQL: `http://spark.apache.org/sql/`

tions [VHKL13, SAKVH15]. It allows for describing a cloud-based application as a so-called service template that consists of two major ingredients. Firstly, a service topology defines how to compose an application of various software components and their relations to each other. Secondly, plans automate tasks to deploy and manage the application and its components within a cloud environment. TOSCA recommends using conventional orchestration workflow technology in order to realize these plans [OAS13]. For instance, OpenTOSCA is a TOSCA engine relying on BPEL for plan execution [BBH+13].

SIMPL relies on conventional orchestration workflow technology as well. Hence, SIMPL-based workflows, e. g., parts of the workflows shown in Figures 5.8 and 5.9, may be seamlessly integrated as plans into TOSCA service templates. This then facilitates and automates the data provisioning and data exchange for cloud-based simulation applications. More details of how to integrate SIMPL with TOSCA are illustrated in a previous author publication [RWWS14]. Dehghanipour discusses more in-depth design considerations, e. g., of integrating the workflow systems of SIMPL and OpenTOSCA [Deh15]. Furthermore, she introduces and evaluates a corresponding prototype for the bone simulation also considered in Section 5.4. For instance, her evaluation confirms that SIMPL is generic enough to seamlessly support the data provisioning for various cloud-based simulation applications.

5.5.5 Data Quality and Provenance Support

As the data management activities of SIMPL offer detailed information about the data management of a workflow, they likewise facilitate both the optimization of data quality and a holistic provenance support. For instance and as discussed in Section 4.2.1, such detailed information is crucial to properly reconstruct the correlation between (1) collected quality or provenance information and (2) affected parts of a workflow model [KSB+10, RBKK14, RSM14a, MBBL15]. Furthermore, it enhances the prospective data provenance of a workflow [MBBL15]. More precisely, it may indicate in detail which data a particular workflow run will access and how it will process this data before the workflow run is actually started. This significantly improves the reproducibility of a simulation and of its outcome [HTT09]. In addition, it may be used to predict more accurately how data quality will evolve during individual future steps of a workflow. On this note, it may be the basis for purposeful and tailored actions that preserve or even enhance data quality in future workflow steps.

5.6 Summary and Future Work

Most of existing workflow systems lack a generic and consolidated solution to data provisioning in simulation workflows. Only Kepler is at least tolerably generic regarding different kinds of data resources [LAB+06, BAJ+10]. Nevertheless, the proprietary workflow actors offered by Kepler are frequently tailored and thus limited to specific data formats and/or data management operations. Among related work, solely ETL technology is sufficiently generic to support all kinds of data resources, formats, and operations required for most examples of computer-based simulations [KRRT98, LN07]. However, corresponding ETL tools offer a multiplicity of diverse ETL operators that may be arranged in data provisioning pipelines. This multiplicity and diversity of ETL operators often overwhelms scientists and induces them not to leverage ETL technology at all.

This chapter discusses a set of extensions to conventional workflow languages that incorporate the general ideas of ETL technology and combine them into an ETL workflow approach [MMLW05, RRS+11]. The resulting data management activities offer a generic solution to data provisioning in simulation workflows that yet does not entail the decisive drawback of ETL tools. In fact, scientists are faced with only four reasonable types of data management activities. Hence, they are not overwhelmed by a vast amount of diverse workflow building blocks. Furthermore, the proposed activities employ the SIMPL core that provides a uniform access to arbitrary data resources. This eases the design of data management activities as it abstracts from technical details of data access mechanisms. A prototypical implementation and its application to several simulation examples has been the basis for a profound evaluation of all contributions. This evaluation has especially confirmed that the proposed data management activities allow for specifying a broad range of data management operations for any kind of data resource or data format. They are thus sufficient to design the data provisioning for virtually all simulation examples of various scientific domains.

Future work may deal with further investigating the potential of the proposed data management activities to facilitate optimizations of workflows. In particular, different kinds of optimization approaches, e. g., those discussed in Section 5.5.4, should be evaluated in detail regarding their suitability to make data processing in simulation workflows more efficient. Note that this goes hand in hand with possible future work discussed in Section 4.5, i. e., a framework that analyzes provenance information to derive recommendations for optimizing workflows.

Chapter 6

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

As discussed in Section 5.5, the ETL workflow approach provided by the data management activities of SIMPL entails various advantages for designing simulation workflows. However, it does not offer all four benefits an adequate abstraction support should offer according to Section 1.2.2 (see also Table 1.1). First of all, it does not reduce the number of tasks scientists have to specify in their workflows. This is also highlighted by the workflows shown in Figures 5.8 and 5.9, which consist of the same high number of tasks as their original versions depicted in Figures 4.5 and 4.6. Furthermore, the ETL workflow approach does not provide scientists with abstract and meaningful workflow building blocks. The data management activities correspond to common use cases for data access, but scientists are typically interested in other use cases, e. g., focusing on coupling simulation models. Finally, the data management activities do not allow for a domain-specific and thus easy parameterization. Scientists actually prefer to work with terms or concepts they already know from their simulation methodology or simulation models. However, data management activities force them to specify many low-level data management operations, e. g., in terms of sophisticated SQL or XQuery statements. Sometimes, the high complexity of data management operations even hinders scientists to specify them at all. This is for instance the case for the SQL `SELECT` statement shown in Listing 5.1, which is part of the coupling workflow depicted in Figure 5.9.

An obvious solution to this problem might be to always opt for guideline 2 proposed in Section 4.3.1.2, i. e., to use data services for any data provisioning task of a simulation workflow. This is because data services usually offer a better abstraction than data management activities. However and as discussed in Section 4.3.2, many simulation workflows exist where available data services do not offer the functionality required by several data provisioning tasks. Guideline 3 proposed in Section 4.3.1.2 then recommends to use data management activities again. Note that this is also the case for several data provisioning tasks of the example workflows shown in Figures 4.5 and 4.6. To make things worse, one task in the workflow shown in Figure 4.5 even requires to opt for guideline 4, i. e., scientists have to implement a completely new data service. As discussed in Section 4.3.2.3, this results in a remarkable implementation overhead, which is even higher than in case scientists rely on data management activities.

Anyway, scientists designing simulation workflows are faced with a considerably high complexity of data provisioning [RLSR$^+$06, RS14, RSM14b]. The resulting huge effort to be spent on workflow design often hinders scientists to concentrate on their core issues, i. e., the actual simulation application and the interpretation of their results. Hence, an adequate abstraction support for designing data provisioning tasks is essential for a wide adoption of simulation workflow technology. As discussed in Section 1.2.2 and summarized in Table 1.1, neither existing workflow systems nor related work from several research areas offer such an adequate and consolidated abstraction support. In particular, none of them provides scientists with all four essential benefits listed in Section 1.2.2.

This thesis fills this gap by proposing a novel pattern-based approach to simulation workflow design that effectively conquers the complexity of data provisioning in simulation workflows. This approach corresponds to the third major contribution introduced in Section 1.3.2.2. Figure 6.1 indicates those components of the SIMPL framework shown in Figure 4.7 that offer this contribution. These components are briefly described in Sections 4.4.2.1 and 4.4.2.4. The *data management (DM) pattern* plug-in of the workflow design tool shown in Figure 6.1 provides a customizable list of patterns representing high-level building blocks for typical data provisioning tasks in simulation workflows. Scientists just need to select a few patterns and combine them in their workflow models. Instead of specifying many low-level details of data provisioning, they afterwards only need to define a small set of domain-specific parameter values for each selected pattern. Examples of such domain-specific parameter values are references to simulation models or

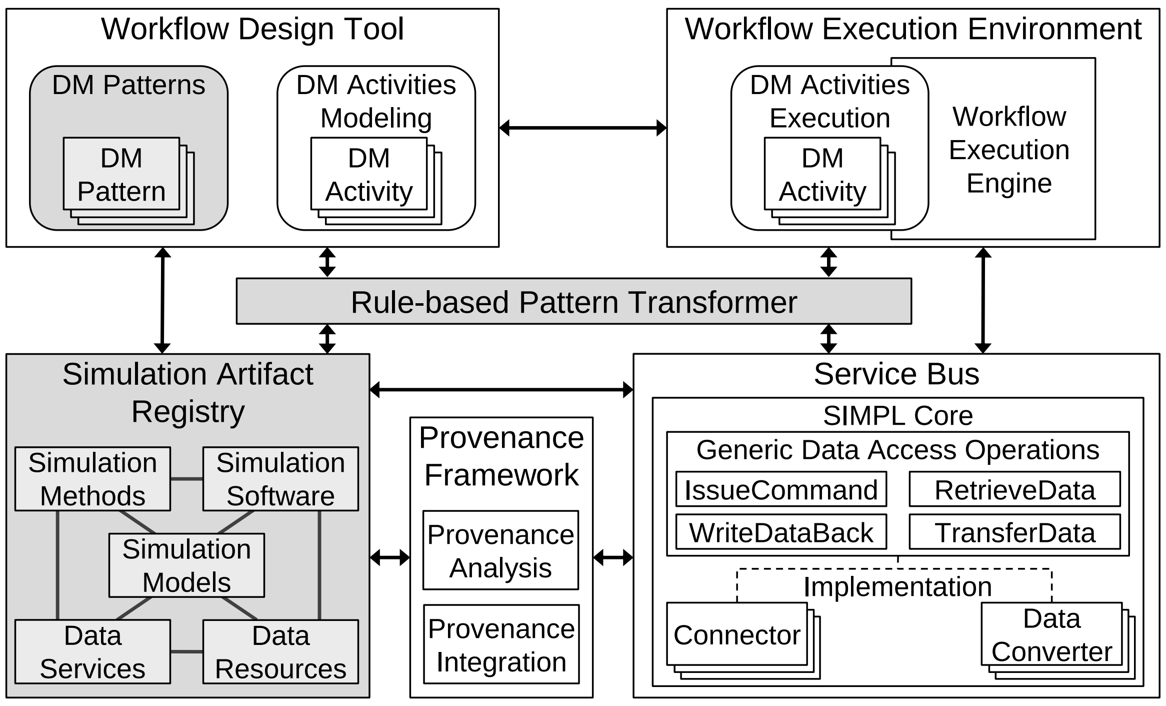


Figure 6.1: Architecture of the SIMPL framework shown in Figure 4.7. Gray-colored components are discussed in this chapter; cf. [RRS+11, RS13b, RS14].

to their mathematical variables, as shown in Figure 1.2. The *rule-based pattern transformer* finally manages an extensible set of rewrite rules that specify how to map abstractly parameterized patterns onto executable workflows. These rewrite rules also make use of the metadata managed by the *simulation artifact registry*. In particular, they use metadata describing dependencies between individual artifacts shown in Figure 6.1 in order to map the domain-specific parameter values of high-level patterns onto more concrete implementation details.

This pattern-based approach and its numerous contributions to a full-fledged and principled abstraction support for simulation workflow design are detailed and assessed in this chapter as follows:

- Section 6.1 illustrates the core idea of the pattern-based approach to simulation workflow design. It therefore depicts the procedure of the overall workflow design approach and its application to the running example of the pattern-based bio-mechanical simulation workflow shown in Figure 1.2.
- Afterwards, Section 6.2 discusses major related work. The main focus of this discussion is why related approaches do not offer a full-fledged abstraction support for simulation workflow design that is tailor-made for scientists.

- The first purpose of Section 6.3 is to present the comprehensive set of patterns that has been devised while working on this thesis. These patterns may be used to alleviate the design of any data provisioning task in several kinds of simulation workflows, e. g., in the examples listed in Section 3.1. Furthermore, Section 6.3 discusses how to organize these patterns in a pattern hierarchy with clearly distinguished abstraction levels. As a decisive contribution, this pattern hierarchy facilitates a separation of concerns between different persons that may now be involved in workflow design, e. g., scientists and data engineers. According to his or her own skills, each person may choose the abstraction level of the pattern hierarchy s/he is most familiar with. S/he may then provide other persons with templates of parameterized patterns and/or workflow fragments at the chosen level.
- Section 6.4 discusses major design considerations for the rule-based transformation of patterns into executable workflows. It illustrates the general rule-based processing model and argues under which circumstances patterns should be transformed during either design time or runtime of workflows.
- Subsequently, Section 6.5 depicts the prototypical implementation of the system components colored gray in Figure 6.1. Furthermore, it exemplifies how the pattern-based approach may be applied to real-world simulations. While Section 6.1 already covers the application to the bio-mechanical simulation workflow, Section 6.5 discusses the most important aspects regarding the more complex coupling workflow shown in Figures 4.6 and 5.9.
- The focus of Section 6.6 is finally to discuss the results of evaluating this prototype and its application to example simulations. This again mainly concerns the benefits and drawbacks of the pattern-based approach to simulation workflow design regarding the challenges discussed in Section 1.2.

Finally, Section 6.7 summarizes the major aspects and lists possible future work. This chapter is a revised and composite version of several previous author publications [RS13b, RS14, RSM14a, RSM14b].

6.1 Pattern-based Simulation Workflow Design

As discussed above, this section illustrates the core idea of the pattern-based approach by (1) depicting the procedure of the overall workflow design approach and by (2) exemplifying its application to the bio-mechanical simulation workflow.

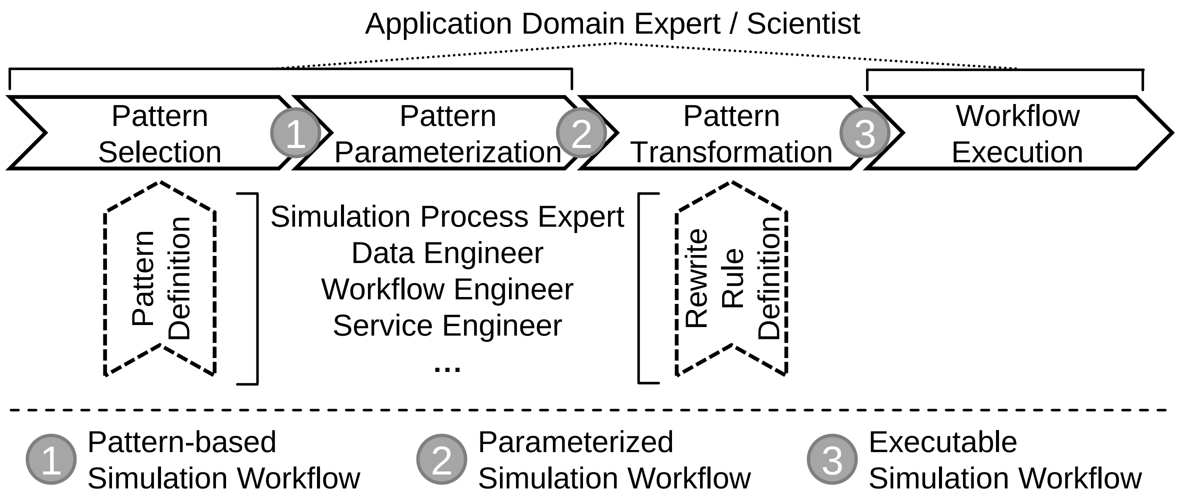


Figure 6.2: Overall pattern-based workflow design approach; cf. [RSM14b].

6.1.1 Overall Workflow Design Approach

The main steps of the overall procedure for the pattern-based approach to workflow design are depicted in Figure 6.2 [RSM14b]. Before scientists come into play, different other persons define and provide certain patterns that may be used as building blocks to design simulation workflows.[1] As discussed above, each pattern may be associated with another abstraction level according to a pattern hierarchy (see also Section 6.3). Hence, a specific pattern is usually provided by a person having tailored skills to cope with the abstraction level of the relevant pattern. As shown in Figure 6.2, typical examples of these persons are simulation process experts, as well as data, workflow, or service engineers [RSM14a].

Application domain experts, i. e., scientists conducting simulations, then select appropriate patterns and combine them in their workflow models. Usually, they select the most abstract patterns that provide all benefits related to an adequate abstraction support according to Section 1.2.2, e. g., the patterns depicted in Figure 1.2. These high-level patterns abstract from multiple low-level workflow tasks, which significantly reduces the number of tasks that are visible to scientists. Furthermore, the patterns are closely related to the simulation models and to the use cases scientists are interested in. This likewise facilitates their domain-specific and easy parameterization, which is the next main step shown in Figure 6.2.

[1]Note that this thesis already proposes a comprehensive set of patterns that are sufficient for various simulation examples. Nevertheless, the pattern plug-in of the workflow design tool shown in Figure 6.1 is designed to be extensible by additional kinds of patterns [Pie12].

After scientists have selected, combined, and parameterized high-level patterns, these patterns need to be transformed into executable workflows. As discussed above, this is achieved by a set of rewrite rules. These rules recursively transform the high-level patterns over the afore-mentioned pattern hierarchy into more concrete workflow patterns, templates of workflow fragments, or data services. They thereby query the simulation artifact registry for certain metadata specifying how to map high-level, domain-specific pattern parameters onto more low-level ones. Depending on their degree of implementation details, the rewrite rules, the metadata, and the associated more concrete patterns, workflow fragments, or services are again provided by different persons having adequate skills (see Figure 6.2). More details about this separation of concerns are given in Section 6.3.

6.1.2 Application to the Bio-Mechanical Simulation

Figure 1.2 exemplifies a high-level pattern-based workflow model that results after scientists have applied the steps of pattern selection and pattern parameterization to the bio-mechanical simulation workflow depicted in Figure 1.1. As discussed in Section 1.3.2.2, the patterns used in this example significantly alleviate workflow design, thereby providing scientists with all four benefits listed in Section 1.2.2. Figure 6.3 depicts how the first pattern shown in Figure 1.2, i. e., the Simulation Calculation Pattern, is recursively transformed into an executable workflow fragment. Thereby, the figure also illustrates more concrete patterns of intermediary abstraction levels of the afore-mentioned pattern hierarchy. More details on applying patterns to the bio-mechanical simulation workflow and on transforming them into executable workflows are given in several student theses that have accompanied the work on this PhD thesis [Ari12, Pie12, Boh14, vS15a].

In a first transformation step, a rewrite rule actually maps the Simulation Calculation Pattern onto two different workflow steps. The first workflow step is another pattern that abstracts from the four data provisioning tasks colored blue in Figure 1.1. The second workflow step is the red task shown in Figure 1.1, i. e., a service call starting the calculation in Pandas. For the sake of clarity and since this thesis focuses on data provisioning in simulation workflows, Figure 6.3 only depicts the pattern abstracting from the four data provisioning tasks.

This Simulation-oriented Data Provisioning Pattern still describes the data provisioning mainly via terms or concepts scientists know from the relevant bio-mechanical simulation model. Its first parameter refers to this model, i. e.,

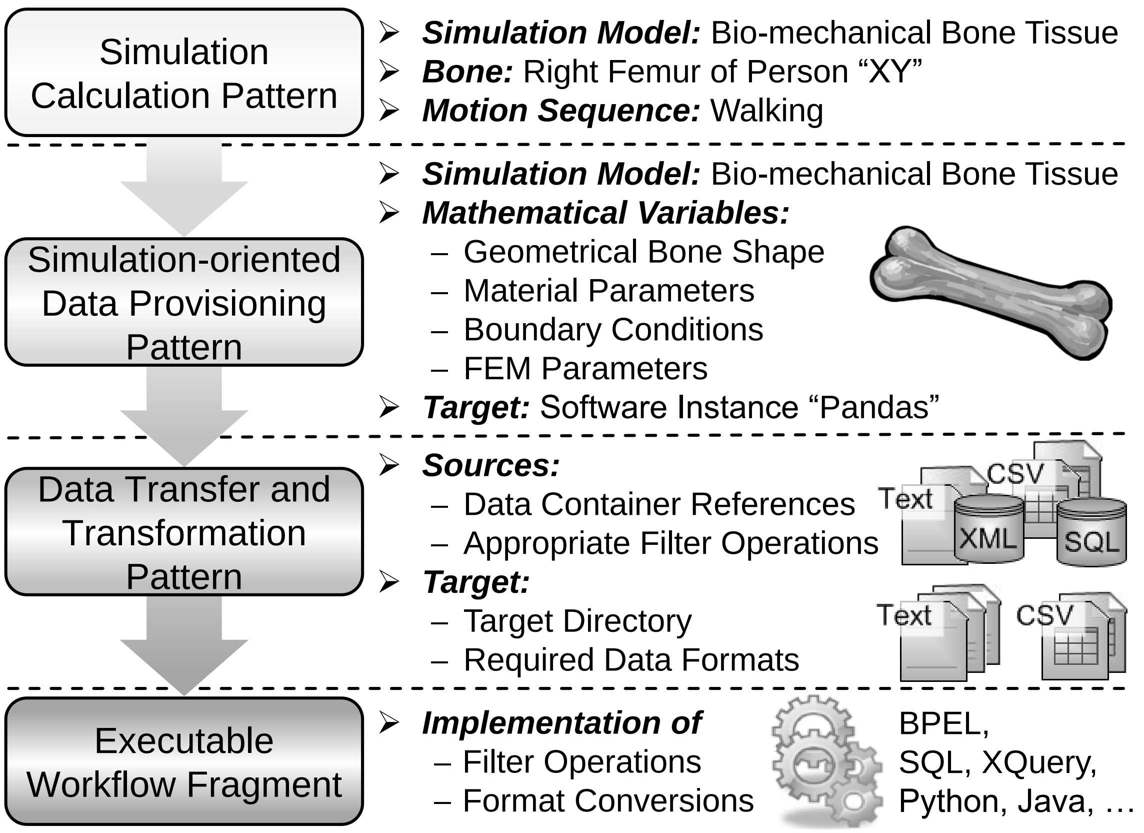

Simulation Calculation Pattern
Simulation Model: Bio-mechanical Bone Tissue
Bone: Right Femur of Person "XY"
Motion Sequence: Walking
Simulation-oriented Data Provisioning Pattern
Simulation Model: Bio-mechanical Bone Tissue
Mathematical Variables:
– Geometrical Bone Shape
– Material Parameters
– Boundary Conditions
– FEM Parameters
Target: Software Instance "Pandas"
Data Transfer and Transformation Pattern
Sources:
– Data Container References
– Appropriate Filter Operations
Target:
– Target Directory
– Required Data Formats
Text
CSV
XML
SQL
Executable Workflow Fragment
Implementation of
– Filter Operations
– Format Conversions
BPEL, SQL, XQuery, Python, Java, …

Figure 6.3: Rule-based transformation of the Simulation Calculation Pattern depicted in Figure 1.2 over several abstraction levels; cf. [RS14].

the rewrite rule of this transformation step directly adopts the model from the superordinate Simulation Calculation Pattern. Furthermore, this rewrite rule maps the bone and motion sequence onto concrete instances of the mathematical or numerical input variables of the bio-mechanical model. These instances of the geometrical bone shape, material parameters, boundary conditions, and FEM parameters thereby have to properly represent the relevant bone and motion sequence (see Section 3.2). To ensure this, the rewrite rule queries the simulation artifact registry and its metadata describing simulation models for the right instances of the bio-mechanical variables [Boh14]. Finally, the Simulation-oriented Data Provisioning Pattern defines the target of the data provisioning as a particular instance of the Pandas software that realizes the bio-mechanical simulation. The rewrite rule of this transformation step may again ask the simulation artifact registry to search for a proper software instance. The registry therefore manages metadata describing simulation software and their instances, as well as which simulation models a particular software may realize [vS15a].

In the second transformation step shown in Figure 6.3, the Simulation-oriented Data Provisioning Pattern is mapped to a Data Transfer and Transformation Pattern (see Figure 3.4). According to its definition illustrated in Section 3.3.1, this pattern describes the data provisioning process via more generic parameters having a stronger relation to underlying data resources and data transformation operations. The rewrite rule of this second transformation step maps the mathematical variables of the bio-mechanical simulation model onto references to heterogeneous data containers that store the data representing these variables. In addition, the rule adds implementation details via specifications of filter operations extracting appropriate data from the data containers. The target of the data provisioning is specified as a directory, where the Pandas software expects its input files. Furthermore, the Data Transfer and Transformation Pattern defines the text-based and CSV-based data formats Pandas requires for these files. The simulation artifact registry facilitates all mentioned mappings from domain-specific parameter values of the superordinate pattern onto low-level and data-specific parameter values of the Data Transfer and Transformation Pattern. This is supported by metadata describing simulation models, data resources, simulation software, and dependencies between these different artifacts [Boh14, vS15a].

Afterwards, another rewrite rule maps the Data Transfer and Transformation Pattern to an executable workflow fragment finally realizing the data provisioning. This workflow fragment has to implement all necessary filter operations or data format conversions and thus contains many complex implementation details. These implementation details are specified via different kinds of query, scripting, or programming languages, as indicated in Figure 6.3. Examples of appropriate executable workflow fragments are variants of the blue workflow steps shown in Figures 1.1, 4.5, and 5.8 [Ari12, Pie12, Boh14].

6.2 Related Work

As discussed in Section 1.2.2, no available workflow system reduces the complexity of data provisioning in workflows to an extent that is especially suitable for scientists conduction simulations. Hence, workflow systems covered as related work in Section 5.1.1 are not considered in detail here. Nevertheless, the next two subsections assess existing work in related research areas of workflow patterns and data-centric workflow design. Afterwards, Sections 6.2.3 and 6.2.4 discuss related work that is also considered in Sections 5.1.2 and 5.1.3, i. e., concerning simulation data management systems and solutions to data integration or data exchange. Finally, Section 6.2.5 summarizes the main conclusions of all discussions.

Table 6.1 on the following page depicts the most important results of assessing major related work and of comparing it with the pattern-based approach to simulation workflow design proposed by this thesis. Thereby, the first four table rows indicate to what extent individual approaches provide scientists designing simulation workflows with the four benefits listed in Section 1.2.2. The last row depicts whether related work offers a holistic separation of concerns that further facilitates simulation workflow design. Here, the main question is whether individual approaches provide a systematic framework that allows for seamlessly incorporating multiple kinds of persons and their specific skills into workflow design. For instance, these persons may be simulation process experts as well as data, workflow, and service engineers (see also Figure 6.2 and Section 6.3).

6.2.1 Common Workflow Patterns

Section 4.2.2 already discusses a set of *common workflow patterns*. This encompasses patterns for control flow, data, resource, and exception handling in business processes, as proposed by Russel et al. and van der Aalst et al. [vdAtHKB03, RtHvdAM06, RtHEvdA05a, RtHEvdA05b, RvdAtH06]. In addition, Pautasso et al., Yildiz et al., and Migliorini et al. introduce further patterns that are especially important for scientific workflows [PA06, YGN09, MGLRtH11]. The main purpose of all these patterns is to provide a comprehensive benchmark to evaluate and compare the basic features offered by different workflow languages and workflow systems. On this note, they represent an extensive set of basic workflow design primitives that, as a whole, are sufficiently generic to express the tasks of any workflow in multiple domains (see Table 6.1).

However, each of these common patterns corresponds to a basic and low-level workflow building block. For instance, some of the data patterns discussed by Russel et al. deal with the question whether workflow tasks or workflow instances may exchange data by value or by reference [RtHEvdA05a]. Such fine-grained patterns are rather suited to characterize low-level details of the executable workflow fragments that finally realize the more abstract patterns proposed by this thesis. Hence and as depicted in Table 6.1, they do not offer the first three essential benefits listed in Section 1.2.2. They are neither suitable to reduce the number of visible workflow tasks, nor are they especially meaningful to scientists

Table 6.1: Assessment of major related work and comparison with the pattern-based approach to simulation workflow design proposed by this thesis.

	Common workflow patterns	**Ontology-based scientific workflows**	**Artifact-centric business processes**	**SDM systems and CAE tools**	**Schema mappings**	**ETL technology**	**SIMPL – Pattern-based workflows**
Reduced number of workflow tasks	○	◑	◑	◑	○	○	●
Meaningful workflow building blocks	○	○	○	◔	○	○	●
Domain-specific parameterization	○	●	●	◕	○	○	●
Generic support of any domain	●	◑	◑	◑	◑	●	●
Holistic separation of concerns	○	◑	◑	◔	○	○	●

or allow for a domain-specific parameterization. Since scientists do not want to struggle with low-level details of workflows, they would typically not accept such fine-grained patterns as initial building blocks for simulation workflow design.

Due to their strong relation to low-level details of executable workflows, the common workflow patterns only cover the lowest abstraction level of patterns shown in Figure 6.3. So, they are not suited to assemble a pattern hierarchy with multiple, clearly distinguished abstraction levels of patterns. This likewise prevents a holistic separation of concerns, where any person may choose an abstraction level that best matches his or her own skills in workflow design.

6.2.2 Data-centric Workflow Design

Researchers from the scientific workflow domain investigate how *ontologies* may be used to abstractly specify workflow tasks, as well as their input and output data [LAG03, BL05, MCD+05, WAH+07, dOCT+09, dOOD+12]. This provides scientists with the third benefit listed in Section 1.2.2, i. e., the ontologies enable a domain-specific parameterization of workflows (see also the third row in Table 6.1). Note that it also allows for incorporating ontology-based approaches to data exchange discussed in Section 5.1.3. In particular, relations between different ontological concepts may be used to describe semantic dependencies between the output data of one task and the input data of another one [BL05]. Furthermore, logical rules may define how to map such semantic dependencies onto additional workflow tasks or services [LAG03]. These additional workflow tasks or services may then realize the necessary low-level data transformations [RLSR^{+}06]. Hence, scientists may use ontologies to abstractly describe the data their workflows shall process, but they do not need to specify all low-level workflow tasks required for data provisioning. This at least constitutes a moderate reduction of the number of tasks that are visible to scientists, as indicated in Table 6.1.

A severe issue is however that these ontology-based approaches do not offer workflow building blocks that are especially meaningful to scientists conducting computer-based simulations. In fact, scientists still need to design particular workflow tasks that consume and generate the abstractly specified data. According to the major application area of data-intensive scientific workflows, these workflow tasks often focus on basic data analysis functions [TDG07, SR09]. Workflow building blocks reflecting basic data analysis functions may be tailored to the skills and needs of data scientists. However, scientists conducting computer-based simulations are typically interested in other use cases. For instance, the pattern shown at the top level of Figure 6.3 represents such a use case that is related to the calculation of a mathematical simulation model.

As indicated in Table 5.1 and discussed in Section 5.1.3, ontologies may be the basis to integrate a big range of data resources and data formats. Furthermore, the foundation on description logic and the native capabilities to support semantic data heterogeneity lead to a high expressive power to specify data transformation operations. Altogether, this entails the potential to provide a generic solution supporting data resources, formats, and operations required by multiple scientific domains. However, this potential is frequently not exploited in practice due to one decisive drawback of ontology-based approaches: They are restricted to those domains, where ontologies already exist. This is mainly because scientists typically do not accept the high initial effort that is required for developing new ontologies [WVV^{+}01]. In the large area of computer-based simulations, ontologies however only exist in very few applications [GCMS12]. As conclusion, the sole use of ontologies does not constitute a completely generic solution, which prevents its seamless adoption in several scientific domains (see Table 6.1).

A limited separation of concerns is generally possible when applying ontology-based approaches to workflow design. Application domain experts, i. e., scientists may parameterize workflows in a domain-specific way using the ontologies. Workflow engineers – and sometimes even data engineers – may then provide the low-level workflow tasks or services implementing necessary data transformations. Nevertheless, this only corresponds to a two-stage transformation from (1) the domain-specific descriptions of workflows to (2) the executable workflows. Related work lacks a clear distinction between multiple abstraction levels, e. g., as provided by the patterns shown in Figure 6.3. Such a clear distinction would however even facilitate a holistic separation of concerns and a more systematic collaboration among different persons having likewise different skills in workflow design. Mork et al. discuss that current scientific workflow systems and related approaches do not adequately support such an interdisciplinary collaboration [MMZ15].

The business process domain proposes analogous approaches to *artifact-centric business process modeling* [NC03, Hul08, CH09, KR11]. These approaches treat data and their evolution over time as first-class citizens to describe and govern a business process. Thereby, so-called business artifacts represent the data of a process in an abstract way. These artifacts correspond to business-relevant entities, e. g., a customer order or an invoice, and manage important information about these entities. Furthermore, the artifacts control the lifecycle of business entities, e. g., how services may be invoked on them and how the underlying data may be changed over time. Figure 6.4 shows an example business artifact representing a deal between a company and one of its customers – including high-level information and lifecycle models for this deal artifact and for the underlying data [CH09]. Related technologies even allow for mapping such artifacts onto low-level data structures or even onto concrete workflow schemas [FHS09, SSWY14].

Regarding the assessment criteria considered in Table 6.1, these artifact-centric approaches bear much resemblance to the ontology-based approaches to scientific workflow design. The artifacts abstract from key functions to access and manipulate data that might otherwise be realized within low-level workflow tasks. Hence, they moderately reduce the number of workflow tasks the application domain experts – in this case business experts – have to design. Furthermore, the artifacts and their information about business-relevant entities provide the means for a domain-specific parameterization of workflows or processes.

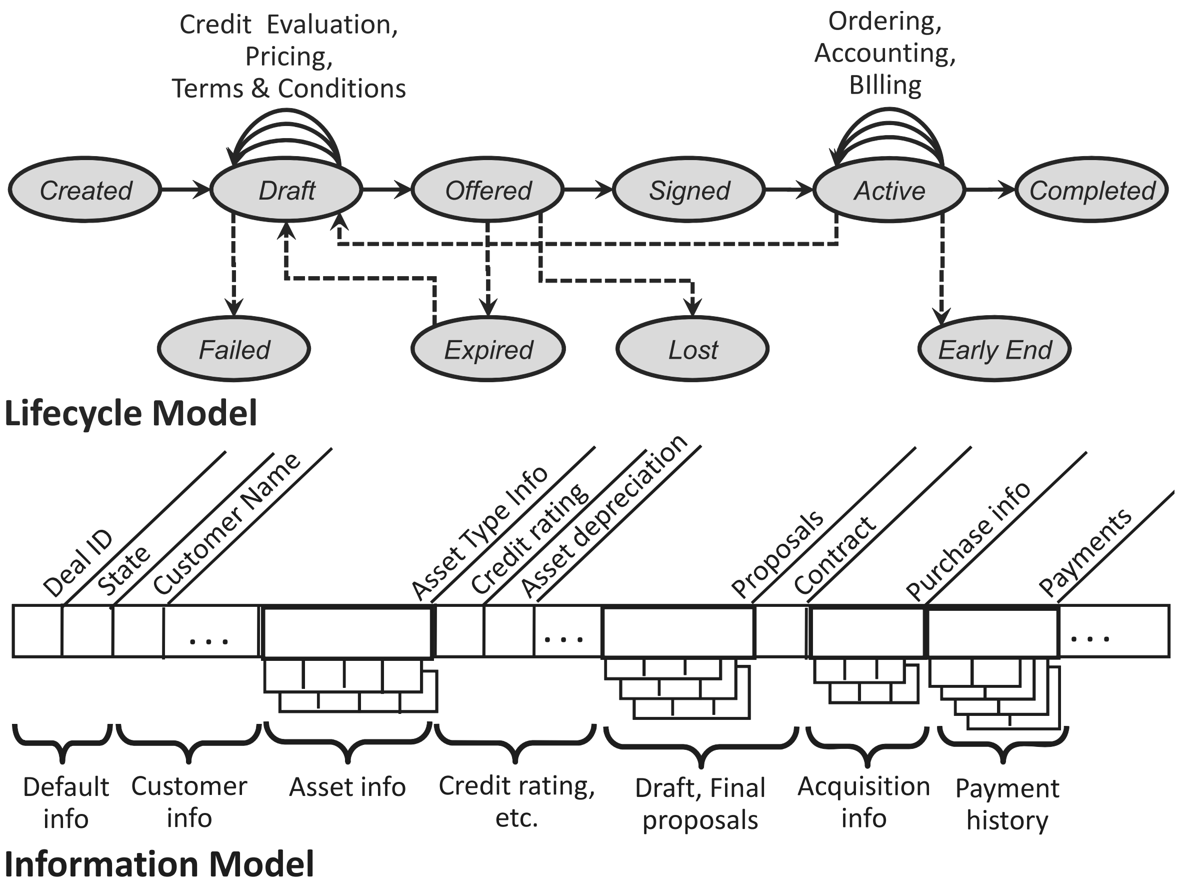


Figure 6.4: Exemplary business artifact representing a deal between a company and a customer [CH09]. The *information model* comprises important data that further describes a deal, whereas the *lifecycle model* defines a state-transition diagram specifying the way a deal and its data may be processed.

Current artifact-centric approaches and concrete technologies mainly focus on the requirements of certain business domains. Hence, available artifacts and their domain-specific descriptions are tailored to the skills and needs of business experts. However, scientists conducting simulations are interested in completely different artifacts, e. g., in mathematical simulation models or simulation methods. Furthermore, they are interested in likewise different use cases to work with these artifacts, e. g., coupling various simulation models. As conclusion, artifact-centric approaches so far do not offer meaningful workflow building blocks that especially reflect the simulation artifacts and the use cases scientists are interested in.

The general flexibility of artifacts to specify important information and the lifecycle of underlying data may offer a high potential to provide a generic solution for any domain. Nevertheless, current artifact-centric approaches and technologies employed in practice are limited to particular business domains and thus do not offer a completely generic solution (see the fourth row of Table 6.1). In fact, they are only applicable to those domains and applications, where the details of corresponding artifacts are already worked out. This is however not the case for most domains related to computer-based simulations.

In an analogous way as ontology-based approaches to scientific workflow design, artifact-centric approaches to business process modeling enable a two-stage separation of concerns. This two-stage separation of concerns is in line with the general process design methodology in the area of business processes [Wes12]. Business experts may use the artifacts of interest to describe a high-level and/or declarative model of the process they have in mind. Workflow engineers or other IT experts may then provide low-level workflow tasks or services that implement some of the functions to be carried out on the artifacts, e. g., basic functions for accessing or manipulating underlying data. Again, a multi-stage mapping of high-level process descriptions onto executable workflows or services over multiple, clearly distinguished abstraction levels would offer a more holistic separation of concerns. In particular, it would provide a systematic framework facilitating the collaboration among more kinds of persons than only business experts and workflow engineers. For instance, simulation workflows and their special challenges regarding data provisioning and data exchange would also benefit from seamlessly incorporating the skills of data engineers.

6.2.3 Simulation Data Management Systems

Besides their mechanisms for a systematic data management, most *SDM systems* or *CAE tools* offer means to design simulation workflows or processes [BBF+09]. Usually, this includes the possibility that scientists or CAE engineers may firstly specify a high-level description of the simulation process they have in mind. This high-level process description then encompasses only a moderate number of process steps. As discussed in Section 5.1.2, SDM systems furthermore offer a comprehensive metadata management, which also allows for a domain-specific description of relevant simulation data. These domain-specific metadata may be adopted in simulation processes as well. More precisely, they may be used to parameterize individual high-level process steps in order to define the data they have to access. Nevertheless, different SDM systems offer likewise different and proprietary solutions for these domain-specific metadata. This typically ranges from purely textual descriptions of data over keyword-based indexing to large taxonomies. These solutions offer a less expressive power than the ontologies or the flexible artifact models employed by approaches to data-centric workflow design. Hence, SDM systems and CAE tools are rated a bit worse in Table 6.1 concerning a domain-specific parameterization of workflows or processes.

The majority of SDM systems or CAE tools however provide scientists or CAE engineers with only a limited set of workflow building blocks that are meaningful to them. As discussed in Section 5.1.2, this set often consists of a small number of data transformation operations that frequently occur in prevalent industrial simulation applications. For instance, this encompasses conversions of default data formats for a CAD model into other default formats for a description of a finite element grid. All other high-level steps of a simulation process that are not directly supported by this limited set of workflow building blocks have to be implemented prior to process execution. Here, low-level techniques are most often the solution of choice, e. g., based on scripting or programming languages.

This limitation to only few common data transformation operations is also one reason why SDM systems and CAE tools do not offer a completely generic solution, as discussed in Section 5.1.2. It often prohibits their adoption in multi-scale simulations that are coupled across different scientific domains, which usually require more complex and domain-specific data transformations. Another issue is that most SDM systems and CAE tools are restricted to document-centric and file-based data. Hence, they cannot be used when simulations rely on other data resources as well. For instance, this concerns the SQL database system used by the running example of a bone simulation (see Figures 5.8 and 5.9).

In principle, a two-stage separation of concerns may again be possible in accordance with the general design methodology for business processes [Wes12]. Scientists or CAE engineers may specify the above-mentioned abstract descriptions of high-level simulation processes. IT specialists may then employ scripting or programming languages to implement those high-level process steps that are not directly supported by the chosen SDM system or CAE tool. However, common SDM systems or CAE tools usually lack a systematic framework that helps companies to accomplish such a two-stage separation of concerns. In fact, each company has to establish its own organizational structure that enables the collaboration among CAE engineers and IT specialists. As a consequence, SDM systems and CAE tools support the required holistic separation of concerns even to a less degree than approaches to data-centric workflow design (see Table 6.1).

6.2.4 Solutions to Data Integration or Data Exchange

As discussed in Section 5.1.3, solutions to data integration, e. g., federated information systems [SL90, BKLW99], do not adequately match the scenario of data provisioning in simulation workflows. Instead, more flexible solutions to a peer-to-peer-like data exchange between individual data resources and/or applications, e. g., those summarized in Table 5.1, constitute a better choice [ABLM14]. Ontology-based approaches to data exchange are already covered by ontology-based approaches to scientific workflows discussed in Section 6.2.2. Thus, remaining related work that is relevant here are schema mappings and ETL technology.

Schema mappings actually offer a high expressive power to specify data transformation operations [Kol05, ABLM14]. This is due to their foundation on first-order predicate logic including some algebraic extensions to describe sophisticated structural dependencies between several data schemata [JPT14]. However and as shown in Table 6.1, schema mappings do not offer a completely generic solution that may be seamlessly used in any simulation domain. In fact, they are not applicable to unstructured text or binary files, which are however most common in computer-based simulations (see Table 3.1). Furthermore, existing solutions to schema mappings do not consider any relation to simulation workflow design. Hence, they likewise do not provide scientists with the first three benefits listed in Section 1.2.2 that are important to conquer the complexity of workflow design. Moreover, the specification of logical and/or algebraic schema mappings is mainly or even only tailored to the skills and needs of data engineers. So, corresponding approaches completely lack a separation of concerns that might also incorporate scientists, workflow engineers, or other kinds of persons.

ETL technology promises to be the most generic solution to data provisioning or data exchange among related work [KRRT98, LN07]. Associated ETL tools cope with virtually all data resources, formats, and operations that are essential for computer-based simulations. However, they do not support the other four assessment criteria depicted in Table 6.1. Scientists relying on ETL technology would need to specify a multiplicity of complex ETL operators in data provisioning pipelines. Furthermore, they would have difficulties with understanding the ETL-specific meanings of individual ETL operators, as discussed in Section 5.1.3. The numerous low-level operator types offered by common ETL tools likewise do not allow for their domain- and simulation-specific parameterization. In fact, the complexity of ETL operators and corresponding data provisioning pipelines is the main reason why common ETL tools are rather tailored to the skills of only data engineers. So, these tools usually do not consider other kinds of persons as their users and thus likewise do not focus on any separation of concerns.

6.2.5 Main Conclusions

As summarized in Table 6.1, none related approach comprehensively offers all four benefits listed in Section 1.2.2 and a holistic separation of concerns. The pattern-based approach to simulation workflow design proposed by this thesis exactly fills this gap and supports all these desiderata in a consolidated and principled fashion. It therefore combines, augments, and considerably goes beyond the general ideas of various related approaches as follows:

- Scientists merely have to combine very few high-level patterns in a workflow, instead of specifying any low-level workflow task. This not only moderately, but even significantly reduces the number of tasks they have to specify.
- As a unique selling point that is not provided by any related approach, this thesis introduces a set of high-level patterns that are particularly meaningful to scientists conducting computer-based simulations. More precisely, these high-level patterns represent the main use cases scientists are interested in, e. g., related to numerically calculating a mathematical simulation model.
- This thesis transfers the core ideas of ontology-based scientific workflows, artifact-centric business processes, and domain-specific metadata of SDM systems to simulation workflow design and augments them by the pattern-based approach. This facilitates a domain-specific parameterization of patterns, where pattern parameters mainly correspond to ontological concepts or artifacts scientists already know from their domain-specific methodology.
- In line with common workflow patterns, the patterns proposed by this thesis are designed to be sufficiently generic to be applicable in any simulation domain. Furthermore, the executable workflows finally realizing patterns may rely on the ETL workflow approach introduced in Chapter 5, which supports a similar range of data resources and operations as ETL technology.
- Another major contribution is provided by the pattern hierarchy and its clearly distinguished abstraction levels of patterns. It enables a holistic separation of concerns and a systematic framework to seamlessly incorporate various skills of different kinds of persons into workflow design.

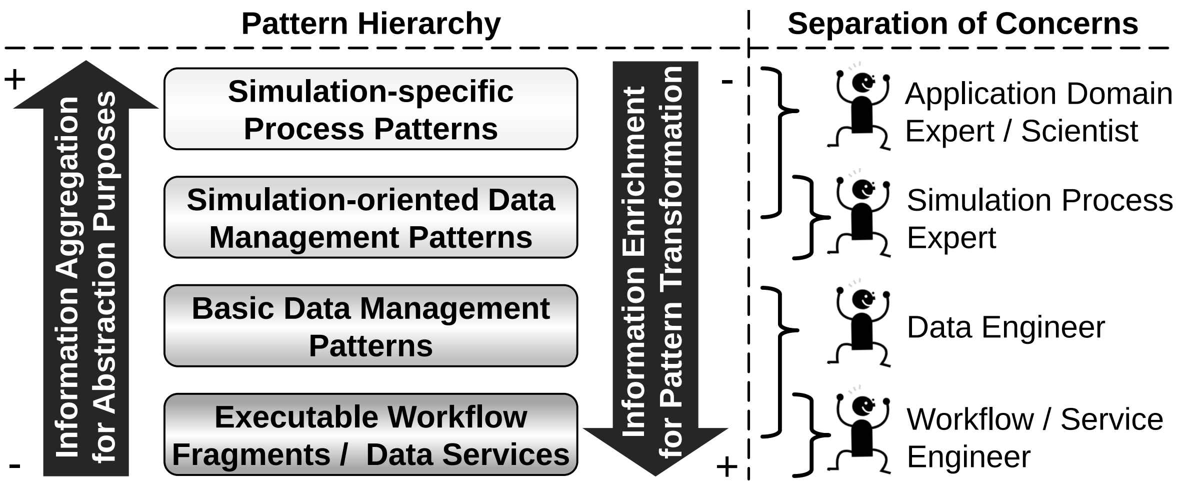

Pattern Hierarchy
Separation of Concerns
+
Information Aggregation for Abstraction Purposes
-
Simulation-specific Process Patterns
Simulation-oriented Data Management Patterns
Basic Data Management Patterns
Executable Workflow Fragments / Data Services
Information Enrichment for Pattern Transformation
-
+
Application Domain Expert / Scientist
Simulation Process Expert
Data Engineer
Workflow / Service Engineer

Figure 6.5: Hierarchy of data management patterns with clearly distinguished abstraction levels enabling a holistic separation of concerns; cf. [RS13b, RSM14a].

6.3 Pattern Hierarchy and Separation of Concerns

Figure 6.5 shows the afore-mentioned pattern hierarchy, which arranges different kinds of patterns according to clearly distinguished abstraction levels [RSM14a]. This pattern hierarchy ranges from (1) simulation-specific process patterns over (2) simulation-oriented data management patterns and (3) the basic data management patterns illustrated in Section 3.3 to (4) executable workflow fragments or data services. Individual patterns at the respective abstraction levels have again been identified by investigating both several academic use cases for simulations (e. g., see [Har96, GZC14]) and the real-world examples listed in Section 3.1.

As discussed above, the major contribution of the clear distinction between individual abstraction levels in the pattern hierarchy is that it facilitates a holistic separation of concerns. Thereby, it allows for systematically incorporating different kinds of persons and their various skills into workflow design. Figure 6.5 proposes a separation of concerns between (a) application domain experts, i. e., scientists, (b) simulation process experts, (c) data engineers, and (d) workflow or service engineers. According to his or her skills, each person may choose the abstraction levels s/he is most familiar with. The person may then provide other persons with templates of parameterized patterns and/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

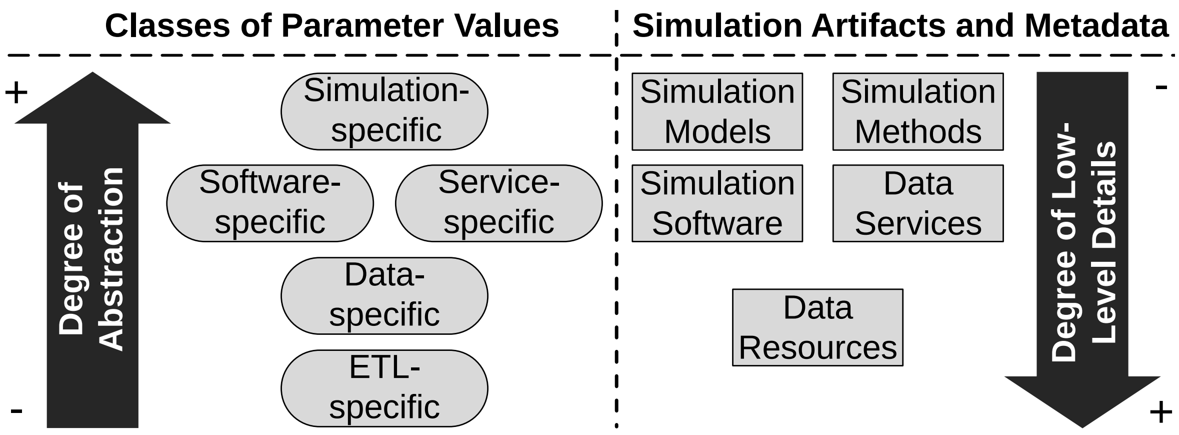


Figure 6.6: Main classes, degree of abstraction, and associated simulation artifacts of parameter values for patterns; cf. [RSM14a].

service engineers may offer executable workflow fragments or services at the lowest level of the hierarchy. In doing so, they only need to know how to implement basic data management patterns provided by data engineers one level above, but they do not need to deal with 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 and may specify more abstract parameter values of patterns. Figure 6.6 classifies such parameter values and relates the resulting main classes according to their degree of abstraction, ranging from simulation-specific to data- or ETL-specific values. As discussed in Section 6.2, this thesis transfers the core idea of artifact-centric business process modeling [NC03, Hul08, CH09, KR11] to simulation workflow design and to corresponding simulation artifacts. Furthermore, it augments the artifact-centric idea by the novel pattern-based approach to workflow design. So, workflow developers may use different kinds of simulation artifacts and their properties to specify parameter values of patterns at different abstraction levels. Figure 6.6 associates each class of parameter values with the corresponding simulation artifacts, i. e., mathematical simulation models, simulation methods, simulation software, data services, and data resources. The simulation artifact registry shown in Figure 6.1 manages comprehensive metadata describing these individual kinds of artifacts and their properties. This assists workflow developers in that they may choose values of some of the pattern parameters from the metadata, thereby facilitating a domain-specific parameterization of patterns.

Note that the four benefits listed in Section 1.2.2 are provided by patterns of all abstraction levels of the pattern hierarchy. Nevertheless, each abstraction level reflects the different skills and needs of the persons associated with the level. For instance, basic data management patterns constitute workflow building blocks that are meaningful to data engineers. On this note, they allow for their tailored parameterization, i. e., mainly relying on data- or ETL-specific parameter values, with which data engineers are familiar. The following subsections detail patterns and their parameters at the individual levels of the pattern hierarchy.

6.3.1 Simulation-specific Process Patterns

The top level of the pattern hierarchy shown in Figure 6.5 comprises simulation-specific process patterns that are good candidates to be selected, combined, and parameterized by scientists. Figure 1.2 depicts examples of such patterns abstractly describing the bio-mechanical simulation workflow, i. e., the Simulation Calculation Pattern and the Simulation Result Interpretation Pattern.[2] Simulation-specific process patterns significantly reduce the number of tasks scientists have to specify in a workflow. For instance, the two patterns depicted in Figure 1.2 abstract from seven original tasks of the bio-mechanical workflow shown in Figure 1.1. Furthermore, the simulation-specific patterns focus on use cases scientists are interested in, e. g., the execution of simulation calculations based on a mathematical model and the interpretation of simulation results. The patterns thus represent workflow building blocks that are particularly meaningful to scientists. According to the classification of parameter values shown in Figure 6.6, scientists may use simulation-specific values for virtually all parameters of these patterns. These values correspond to artifacts or their properties scientists already know from their domain-specific simulation methodology and are thus familiar with. Major examples of such artifacts are mathematical simulation models and simulation methods, e. g., numerical methods such as the FEM.

The simulation artifact registry shown in Figure 6.1 manages metadata describing relevant simulation models and simulation methods and this way helping scientists to parameterize simulation-specific process patterns. For instance, metadata describing simulation models may indicate, which mathematical variables of a particular model are valid parameter values of the patterns shown in Figure 1.2.

[2]Simulation-specific process patterns that may alleviate the design of the more complex coupling workflow are depicted in Figure 6.8 on page 188 and discussed in Section 6.5.1.

The metadata-based description of domain-specific simulation models and simulation methods may rely on likewise domain-specific languages (DSLs). Ontologies are a good basis for such DSLs, at least in case they already exist in the relevant application domain. For instance, Cook et al. and Dao et al. propose ontologies representing biological or bio-mechanical issues [CMJNG08, DMHBT07]. These ontologies – together with a generic ontology for the domain systems-biology[3] – have been successfully revised to special ontologies describing the simulation models and methods of the running example of a bone simulation [Boh14, vS15a, vS15b]. This way, this thesis also transfers the core idea of ontology-based scientific workflows [LAG03, MCD+05, WAH+07, dOCT+09, dOOD+12] to simulation workflow design and augments this idea by the pattern-based approach. More precisely, scientists may use corresponding ontological terms or concepts to specify parameter values of simulation-specific process patterns.

Another prominent example, where ontologies already exist, is the scientific domain of life sciences [SWLG04]. Some other scientific domains at least employ shared vocabularies or taxonomies, which may likewise be the basis to develop ontologies describing simulation models and related methods. Note that shared vocabularies or taxonomies are often the solution of choice for domain-specific metadata managed by SDM systems (see Section 6.2.3). Hence, SDM systems may in some cases also be a foundation to develop parts of the metadata managed by the simulation artifact registry shown in Figure 6.1.

6.3.2 Simulation-oriented Data Management Patterns

Table 6.2 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 constitute workflow building blocks that focus on use cases related to data management, especially to data provisioning and data exchange. As depicted in Figure 6.5, this is where simulation process experts come into play, who may assist scientists in properly combining these patterns in their workflows.

The *Simulation-oriented Data Provisioning Pattern* is also exemplified on the second abstraction level of patterns shown in Figure 6.3. It abstracts from data provisioning processes for simulation calculations or result interpretations, i. e., as part of the simulation-specific process patterns Simulation Calculation or

[3]Systems Biology Ontology: `http://www.ebi.ac.uk/sbo/main/`

Table 6.2: Simulation-oriented Data Management Patterns; cf. [RSM14a].

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 Exchange	• Simulation models <2> • Relationships between mathematical variables: – From first to second model <1..n> – From second to first model <0..n>
Parameter Sweep	• Parameter List <1> • Operation <1>: Simulation model, service, or workflow fragment to be executed for each parameter in the list • Completion Condition <0..1> • Parallel <0..1>: “yes” / “no”, default is “no”

Simulation Result Interpretation. The data to be provisioned is represented by a *simulation model* and by a set of its *mathematical variables*. In case of providing data for simulation calculations, these mathematical variables correspond to input variables of the simulation model, e. g., its parameter variables as shown in Figure 2.1. When data is provided for result interpretations, the common way is to use unknown variables of a simulation model.

Only the *target* of the data provisioning needs to be specified via a less abstract software- or service-specific parameter value. This target specification is usually a reference to a software instance or to a service that needs the data as input. Note that scientists are often still quite familiar with such software or service references. The reason is that they employ software or services in their everyday life to conduct simulation calculations. Furthermore, the simulation artifact registry again manages metadata describing simulation software and services and thereby providing scientists with suggestions for proper software or service references. Metadata describing software and services as simulation artifacts may be based on common repository or registry solutions [Ley03, Ley05]. For instance, they may be backed up by mature database technology [Pie12].

The *Simulation-oriented Data Exchange Pattern* reflects the scenario, where data has to be exchanged between *two mathematical simulation models* that are coupled together. Scientists may specify data dependencies between these two models completely via simulation-specific parameter values [RSM14b]. More precisely, these data dependencies correspond to the *relationships between the mathematical variables* of the two models, e. g., as depicted for the bio-mechanical and systems-biological models in Figure 2.1 [vS15b]. The relationships between mathematical variables have to be specified at least in a unidirectional way, i. e., *from the first to the second model.* This is sufficient when the calculation of the second model is not started until the calculation of the first model has been finished. For instance, this is the case in the coupling process shown in Figure 3.2 executing bio-mechanical and systems-biological calculations one after another. The other possibility is to define the relationships in a bidirectional way, i. e., additionally *from the second to the first model.* This may be necessary when applying more complex coupling algorithms [UGM14]. For example, it is relevant when both simulation models are calculated concurrently and need to exchange their data mutually at certain integration steps.

The *Parameter Sweep Pattern* supports processes that iterate over a *list of* simulation-specific *parameters.* Furthermore, it carries out an *operation* for each parameter in this list. This pattern for instance occurs in the coupling workflow shown in Figure 5.9. Here, the list of daily routines represents a simulation-specific parameter list. The outer loop iterating over this list of daily routines, as well as the workflow tasks embedded into this loop correspond to the operation of the pattern. So, this operation is specified as a sophisticated workflow fragment containing the loop and its embedded tasks. Another option is to provide a reference to a service representing the operation. Nevertheless, scientists may also use a convenient simulation-specific value, i. e., a mathematical simulation model. This means that the simulation model is to be calculated for each parameter in the specified list.[4] An optional *completion condition* of the pattern defines whether and when the iteration shall be finished before the whole parameter list is processed [OAS07]. Finally, another pattern parameter indicates whether several instances of the specified operation shall be executed in *parallel* or not.

6.3.3 Basic Data Management Patterns

On the way to executable workflows, these simulation-oriented patterns are intermediately mapped onto the basic data management patterns illustrated in Section 3.3. Table 6.3 summarizes the parameters to be specified for these basic patterns. *Data Transfer and Transformation Patterns* (see also Figure 3.4)

[4]More details on how this kind of operation specification works are given in Section 6.5.2.

Table 6.3: Basic Data Management Patterns; cf. [RSM14a].

Pattern	**Parameters (<n..m> indicates cardinality)**
Data Transfer and Transformation	• Sources <1..n>: references to data containers • Targets <1..n>: references to data containers • Dependencies from sources to targets <0..n>: schema mappings, inter-ontology mappings, or ETL operations
Data Iteration	• Data set <1>: one or a set of data containers • Operation <1>: service or workflow fragment 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 <0..1>: data-specific partitioning mode or parameters of Data Transfer and Transformation Pattern • Data merge <0..1>: similar to data split

may be used to implement Simulation-oriented Data Provisioning Patterns and Simulation-oriented Data Exchange Patterns. The *sources* and *targets* of the underlying data provisioning processes are specified as references to data containers, e. g., to database tables or to files. Corresponding references to data are classified as data-specific parameter values according to Figure 6.6. They are situated at a lower abstraction level, since scientists are usually more familiar with software and services than with references to heterogeneous data. Structural and/or semantic *dependencies from the sources to the targets* may be specified using the solutions to data exchange discussed in Sections 5.1.3 and 6.2.4. So, this covers schema mappings [Kol05, ABLM14], inter-ontology mappings [WVV$^+$01], or even specifications of sophisticated ETL operations [KRRT98, LN07]. Figure 6.6 classifies such values of pattern parameters relying on solutions to data exchange as ETL-specific values. They cover many low-level details describing complex data transformation operations. Hence, they exhibit the lowest degree of abstraction of all classes of parameter values for patterns.

The major use case of *Data Iteration Patterns* depicted in Figures 3.5 and 3.6 is to provide more low-level details of simulation-oriented Parameter Sweep Patterns. References to one or more data containers specify the *data set* over which a Data Iteration Pattern shall iterate. Furthermore, the pattern indicates an *operation*, where this data set or relevant subsets of it serve as input, e. g., a particular service. The operation may be executed on a set of possibly distributed *resources*. Again, this execution may optionally end as soon as a *completion condition* holds, and the operation may be executed either in *parallel* or not. A *data split* parameter may define how to partition the data set among the resources. A possible parameter value is a data-specific partitioning mode, e. g., according to the equal distribution of tuples in a database table [Pie12]. As alternative, the data split may also be defined via ETL-specific parameters of a Data Transfer and Transformation Pattern. In this case, it may additionally cover operations for preparing the data set prior to its distribution, e. g., for filtering the data. In a similar way, a *data merge* parameter may define how to integrate the results of the operation back into the original data set.

In summary, the majority of the parameters of basic data management patterns have to be defined using data-specific or even more low-level ETL-specific values. So, these parameters have a strong relation to underlying data resources and data transformation operations. According to the separation of concerns depicted in Figure 6.5, data engineers may use their knowledge to provide other persons with templates of basic data management patterns and their low-level parameterization. Thereby and as shown in Figure 6.6, relevant simulation artifacts that assist data engineers in the data-specific and ETL-specific pattern parameterization are data resources. The metadata describing these data resources in the simulation artifact registry may be based on those illustrated in Section 5.3.3 [RRS+11].

6.3.4 Executable Workflow Fragments

Executable workflow fragments finally realize the patterns discussed above. Examples of such workflow fragments are the individual parts of the bio-mechanical simulation workflow shown in Figure 5.8 that are respectively colored blue, red, green, or purple. Executable workflow fragments usually contain many complex implementation details. In particular, they may employ various low-level data provisioning techniques, e. g., represented as data services or as the data management activities summarized in Table 5.2. As shown in Figure 6.5, workflow or service engineers may incorporate their skills to provide templates of workflow fragments and to properly combine different data provisioning techniques in them. Nevertheless, they may also need help from data engineers to design very complex data provisioning tasks. This is for instance necessary for specifying the sophisticated SQL `SELECT` statement shown in Listing 5.1.

6.4 Rule-based Pattern Transformation

The proposed pattern-based approach to simulation workflow design needs to be complemented by a strategy to transform abstract patterns into executable workflows. This strategy is provided by the rule-based pattern transformer depicted in Figure 6.1 and by its extensible set of rewrite rules. The following subsections discuss major design considerations of this rule-based transformation of patterns. This includes (1) its general processing model and (2) a discussion under which circumstances patterns should be transformed during either design time or runtime of workflows.

6.4.1 Rule-based Processing Model

The general processing model of the rule-based pattern transformation, as shown in Figure 6.7, extends some basic ideas proposed by Vrhovnik et al. [VSS+07]. The rule-based pattern transformer traverses the graph of parameterized patterns given by the relevant workflow model. For each pattern in this graph, the *pattern transformation engine* tries to apply rewrite rules. Thereby, the goal is to recursively map patterns over the pattern hierarchy depicted in Figure 6.5 onto more concrete patterns and finally onto executable workflow fragments.

The application of rewrite rules to patterns is governed by several *rule sequences.* For instance, each level of the pattern hierarchy shown in Figure 6.5 may be associated with a particular rule sequence. It defines the set of rules that may generally be applied to patterns of the relevant abstraction level. Furthermore, it determines the order in which these rules are tested for applicability. In other words, the first rule in a sequence that is found to be applicable is actually applied to the relevant pattern, while all remaining rules are neglected. Note that the order of rule application determined by a rule sequence has to reflect the guidelines proposed in Section 4.3.1 and depicted in Figure 4.4. For instance and according to guideline 1, rewrite rules that are firstly tested for applicability should lead to workflow fragments that avoid using local data processing steps.

Each rewrite rule includes a *condition* part that specifies the circumstances under which the remaining parts of the rule may be applied to the given pattern. The conditions may for instance depend on certain parameter values of the pattern [Pie12]. Depending on the concrete goal of their application, rewrite rules may occur in two different types. A *rewrite rule of type 1* leads to a workflow

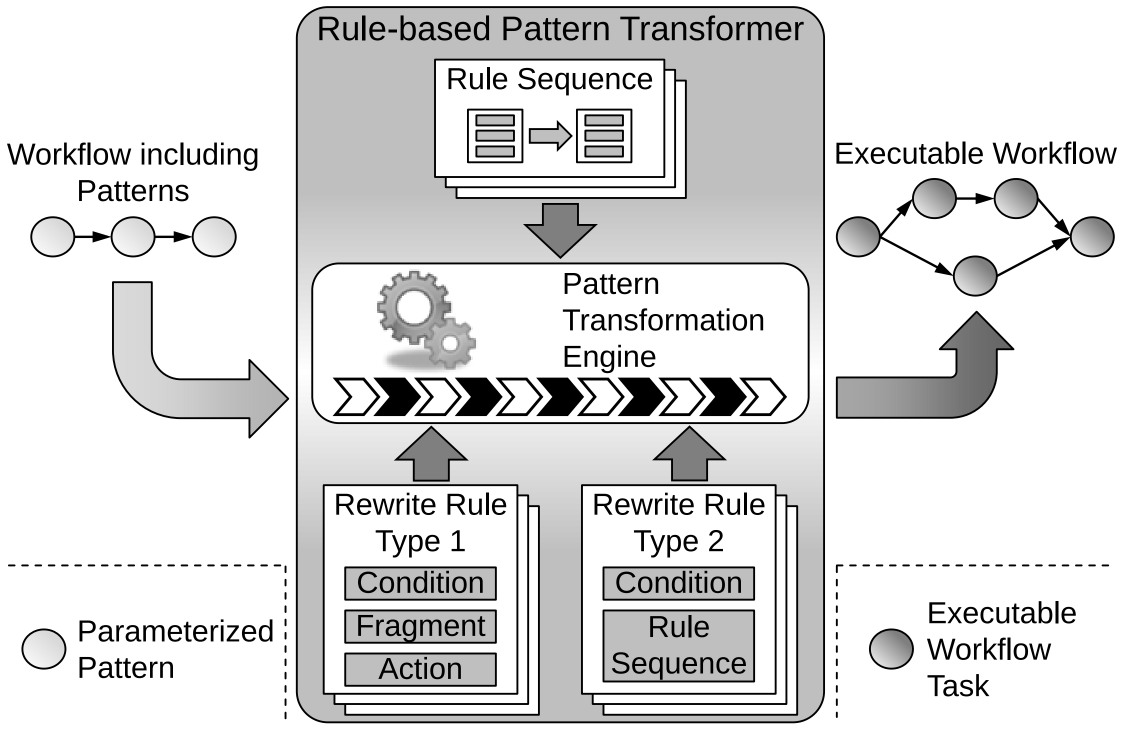


Figure 6.7: Processing model of the rule-based transformation of patterns into executable workflows; cf. [RSM14a].

fragment providing more low-level details for the pattern. Its *fragment* part identifies a template of such a workflow fragment, e. g., via a query to a workflow fragment library [SKK+11]. The *action* part then defines transformation steps that add implementation details to the workflow fragment. Thereby, this action part enriches the previously aggregated information about data management operations with descending levels of the pattern hierarchy shown in Figure 6.5. It therefore may also need to map parameter values of patterns from high to low abstraction levels according to the classification given in Figure 6.6. As discussed in Section 6.1, the simulation artifact registry shown in Figure 6.1 supports this mapping of parameter values by providing additional metadata describing dependencies between individual kinds of simulation artifacts.

Finally, the rule application replaces the pattern in the workflow model with the resulting workflow fragment. This workflow fragment may either be completely executable or it may embed other patterns. The first case finishes the transformation of the relevant pattern. In the second case, the pattern transformer engine recursively applies further rewrite rules to the workflow fragment and to its embedded patterns until all final workflow fragments are executable.

Rewrite rules of type 2 do not directly lead to workflow fragments realizing a pattern. Instead, they establish a hierarchical rule approach in that they identify further *rule sequences* to be evaluated on the pattern. This may be suitable, e. g., for multi-stage optimization decisions. Here, rules of type 2 may firstly evaluate functional requirements of a pattern. The rules in the resulting rule sequence may then be of type 1 and may refer to candidate workflow fragments that fulfill these functional requirements. Furthermore, these rules of type 1 evaluate non-functional requirements to select an optimal workflow fragment.

6.4.2 Point in Time for Pattern Transformation

In several scenarios, a pattern transformation exclusively during design time of workflows is not feasible, as discussed by Bohrn [Boh15]. For instance, rewrite rules might embed optimization decisions that require accurate information on the size of data to be processed by the final executable workflow [BHW^{+}07, Kal15]. However, this information on data sizes is often not available during design time. Nevertheless note that such optimization decisions may usually be postponed until basic data management patterns have to be transformed into executable workflows [Kal15]. This argues for a hybrid approach, where simulation-specific process patterns and simulation-oriented data management patterns are firstly transformed during design time of workflows. Only the subsequent basic data management patterns may then call for their transformation during runtime – at least in case this is necessary due to some optimization decisions.

Another, even more important motivation to transform patterns during runtime of workflows is given by the fact that scientists often make ad-hoc changes to workflows at runtime [SK10] (see Section 1.2.5). So, they may also make ad-hoc changes to high-level simulation-specific process patterns. This even makes it necessary to re-iterate the pattern transformation over all abstraction levels of the pattern hierarchy shown in Figure 6.5 during runtime [Boh15].

From a technical point of view, a pattern transformation during runtime is possible, because the approach presented by Gómez Sáez et al. enables a dynamic modification and replacement of workflow parts [GSAH^{+}15]. However, the overhead caused by the pattern transformation then also affects the duration of workflow executions. Section 6.6.4 therefore discusses experimental results showing that the usual overhead caused by the pattern transformation is negligible compared to the typical duration of simulation workflows.

6.5 Prototype and its Application to the Bone Simulation

The prototype of the proposed pattern-based approach to simulation workflow design has again been developed in the course of various student projects that have accompanied the work on this thesis [Ari12, Pie12, Boh14, Rie14, Boh15, Kal15, vS15a, vS15b]. It is based on the prototype presented in Sections 4.4.3 and 5.4. The data management (DM) patterns plug-in shown in Figure 6.1 extends the Eclipse BPEL Designer version 0.8.0 by multiple kinds of patterns of all abstraction levels shown in Figure 6.5. This especially includes the patterns shown in Figures 1.2 and 6.8, as well as those summarized in Tables 6.2 and 6.3 [Pie12, Boh14, Rie14, vS15b]. All other system components highlighted via gray color in Figure 6.1 are implemented as separate Java-based Web Services, which are deployed on Apache Tomcat Version 7.0 using Axis2 version 1.5.4.

As discussed in Section 5.4, the simulation artifact registry uses a PostgreSQL version 9.2 database system to manage metadata describing data resources. This also holds for metadata describing data services [Pie12]. Because properties of simulation models, simulation methods, and partly also of simulation software are often domain-specific, the prototype relies on ontologies to realize corresponding metadata [Boh14, vS15a, vS15b]. These ontologies are managed using Apache Jena[5] version 2.11.2. The dependencies between individual kinds of simulation artifacts are covered by foreign key definitions or by ontological annotations.

The Java-based Web Service implementing the rule-based pattern transformer extends the rule engine Drools[6]. These extensions support all rewrite rules and rule sequences that are necessary to transform the patterns implemented within the prototype [Pie12, Boh14, Rie14, Boh15, Kal15, vS15b]. Bohrn especially designed the pattern transformer so that it enables the transformation of patterns not only during design time of workflows, but also during their runtime [Boh15]. Thereby, the replacement of patterns with more low-level workflow fragments is based on the prototype presented by Gómez Sáez et al. [GSAH$^+$15].

This prototype and all its patterns and rewrite rules have been applied to each individual workflow of the running example of a bone simulation [Pie12, Boh14, Rie14, vS15b]. Furthermore, this has been the basis to assess at a conceptual level

[5]Apache Jena: `https://jena.apache.org/`
[6]Drools: `http://www.drools.org/`

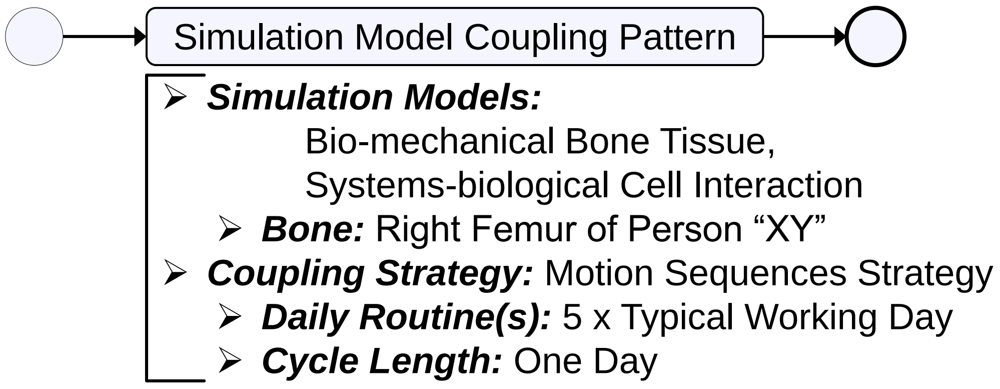

Simulation Model Coupling Pattern
Simulation Models:
Bio-mechanical Bone Tissue,
Systems-biological Cell Interaction
Bone: Right Femur of Person "XY"
Coupling Strategy: Motion Sequences Strategy
Daily Routine(s): 5 x Typical Working Day
Cycle Length: One Day

Figure 6.8: Simulation-specific process pattern to alleviate the design of the coupling workflow shown in Figures 4.6 and 5.9; cf. [RSM14a].

to what extend the patterns proposed by this thesis may alleviate workflow design for the other simulation examples listed in Section 3.1. For instance, Pietranek discusses corresponding assessment results regarding the simulations of catalysis reactions introduced by Rommel et al. [RK11, Pie12]. Nevertheless and in analogy to the reasons discussed in Section 4.3.2, the bone simulation is again well-suited to illustrate the major aspects of applying the pattern-based approach to real simulations. Section 6.1 already covers the application to the bio-mechanical workflow depicted in Figures 1.1 and 1.2. Hence, the following subsections discuss important aspects regarding the more complex coupling workflow shown in Figures 4.6 and 5.9. Firstly, Section 6.5.1 illustrates a high-level simulation-specific process pattern making the design of this workflow especially tailor-made for scientists. The focus of Section 6.5.2 is then on exemplifying the rule-based transformation of this high-level pattern into an executable workflow.

6.5.1 Simulation-specific Process Pattern

The Simulation Model Coupling Pattern shown in Figure 6.8 describes the whole coupling workflow depicted in Figures 4.6 and 5.9 in a way that provides scientists with all the benefits listed in Section 1.2.2:

- It significantly reduces the number of workflow tasks scientists have to specify from a multiplicity of tasks shown in Figure 5.9 to only one pattern.
- This pattern represents one of the main use cases scientists are interested in, i. e., coupling two mathematical simulation models. So, it constitutes a workflow building block that is particularly meaningful to scientists.

- It likewise allows for its domain-specific and thus easy parameterization. In fact, each individual pattern parameter may be specified using simulation-specific values according to the classification given in Figure 6.6. So, these parameter values correspond to terms or concepts scientists already know from their domain-specific simulation models or simulation methods.
- Finally, the use case of coupling simulation models represented by the pattern is common for the simulation domain [GZC14, Gat14, UGM14]. The pattern in itself is therefore sufficiently generic to be re-usable in various simulation examples of different scientific domains.

The first parameter of the pattern shown in Figure 6.8 determines the *simulation models* to be coupled together. These are the bio-mechanical bone tissue model and the systems-biological cell interaction model also depicted in Figure 3.2. In addition, a parameter that is specific to these two models defines the concrete *bone* to be simulated, e. g., the right femur of a certain person. The next pattern parameter points to the *coupling strategy* that indicates the concrete process how both simulation models are coupled together. The coupling strategy used in Figure 6.8 corresponds to the process shown in Figure 3.2. This coupling process firstly carries out several bio-mechanical calculations in parallel, i. e., one calculation for each typical motion sequence of the relevant person. Afterwards, ETL processes prepare the bio-mechanical simulation results for the subsequent systems-biological calculations. Thereby, the bio-mechanical results for individual motion sequences are composed to an idealized solution approximating the whole daily routines of the relevant person. The next parameter of the pattern shown in Figure 6.8 defines these *daily routines* and its composition of individual motion sequences. Moreover, the *cycle length* determines the frequency in which the coupling process re-iterates among both simulation models. The example considers a cycle length of one day and a total duration of five days, where each daily routine corresponds to the typical working day of the relevant person.

6.5.2 Rule-based Pattern Transformation

Figure 6.9 illustrates the main transformation steps to map the Simulation Model Coupling Pattern shown in Figure 6.8 onto an executable workflow. The following subsections respectively exemplify the patterns, workflow fragments, and rewrite rules used at each individual abstraction level of the pattern hierarchy depicted in Figure 6.5. More details on this example are discussed by von Steht [vS15b].

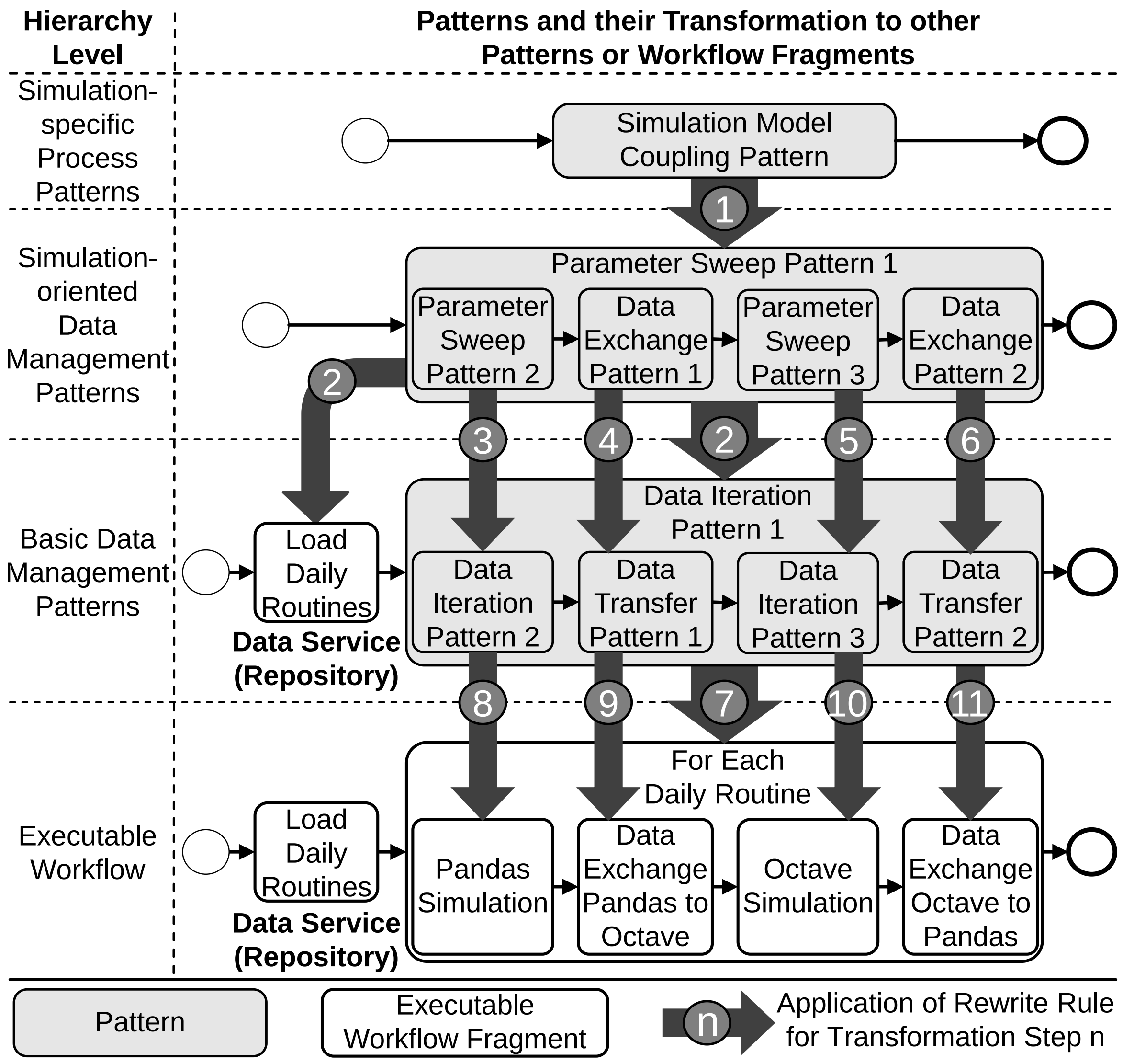

Hierarchy Level
Patterns and their Transformation to other Patterns or Workflow Fragments
Simulation-specific Process Patterns
Simulation Model Coupling Pattern
1
Simulation-oriented Data Management Patterns
Parameter Sweep Pattern 1
Parameter Sweep Pattern 2
Data Exchange Pattern 1
Parameter Sweep Pattern 3
Data Exchange Pattern 2
2
3
4
2
5
6
Basic Data Management Patterns
Load Daily Routines
Data Service (Repository)
Data Iteration Pattern 1
Data Iteration Pattern 2
Data Transfer Pattern 1
Data Iteration Pattern 3
Data Transfer Pattern 2
8
9
7
10
11
Executable Workflow
Load Daily Routines
Data Service (Repository)
For Each Daily Routine
Pandas Simulation
Data Exchange Pandas to Octave
Octave Simulation
Data Exchange Octave to Pandas
Pattern
Executable Workflow Fragment
n
Application of Rewrite Rule for Transformation Step n

Figure 6.9: Main transformation steps mapping the simulation-specific process pattern shown in Figure 6.8 onto an executable workflow; cf. [RSM14a, vS15b].

6.5.2.1 Simulation-oriented Data Management Patterns

The workflow fragment resulting from the Simulation Model Coupling Pattern after transformation step 1 consists of five simulation-oriented data management patterns. This workflow fragment and its patterns realize the coupling strategy indicated in Figure 6.8, i. e., the process depicted in Figure 3.2. The five patterns may again be parameterized completely via simulation-specific values.

The simulation-specific parameter list of *Parameter Sweep Pattern 1* corresponds to the list of daily routines that is also specified for the Simulation Model

Coupling Pattern shown in Figure 6.8, i. e., the five typical working days. Furthermore, the operation of the parameter sweep is represented by the embedded control flow of the four remaining patterns. In other words, Parameter Sweep Pattern 1 sequentially iterates over the list of daily routines and executes its four embedded patterns for each day. The control flow of these four embedded patterns thereby realizes exactly one coupling cycle depicted in Figure 3.2.

The first embedded pattern, i. e., *Parameter Sweep Pattern 2*, iterates in parallel over the list of motion sequences of the respectively current day. The pattern moreover defines the bio-mechanical simulation model as its operation. This indicates that the bio-mechanical model is to be calculated for each individual motion sequence. The subsequent *Simulation-oriented Data Exchange Pattern 1* specifies the data exchange from the bio-mechanical to the systems-biological model in a simulation-specific way. It therefore defines the relationships between their mathematical variables in terms of dependencies between the respective differential equations, i. e., as described by Krause [Kra14]. This is also indicated by the variable exchanges depicted by steps 1, 2, and 3 in Figure 2.1.

Parameter Sweep Pattern 3 analogously defines the systems-biological simulation model as its *operation*. As discussed in Section 2.1.1, this simulation model may be calculated locally and thus concurrently at each integration point of the underlying finite element grid (see also Figure 2.2b). So, the pattern specifies the list of relevant integration points as its simulation-specific parameter list. Furthermore, it iterates over this list in parallel to account for the concurrent execution of individual systems-biological calculations. Afterwards, *Simulation-oriented Data Exchange Pattern 2* abstracts from the process to exchange data from the systems-biological model back to the bio-mechanical model. Hence, the corresponding relationships between the respective mathematical variables are those indicated by steps 4 and 5 depicted in Figure 2.1 [Kra14].

6.5.2.2 Basic Data Management Patterns

After transformation steps 2 to 6, the workflow consists of one service call and five basic data management patterns as shown in Figure 6.9. At this abstraction level, the patterns are specified mainly via service-, data-, and ETL-specific parameter values according to the classification given in Figure 6.6.

The first rewrite rule maps the original Parameter Sweep Pattern 1 onto the afore-mentioned service call and onto the subsequent Data Iteration Pattern 1 (transformation step 2 in Figure 6.9). The service accesses a repository that delivers the list of daily routines including their motion sequences (see also Section 4.3.2.2). The workflow then stores this list into a local workflow variable. The subsequent *Data Iteration Pattern 1* defines this local variable as its data set, i. e., it sequentially iterates over the stored list of daily routines. For each daily routine, it carries out its operation, which is again the embedded control flow consisting of the four remaining basic data management patterns.

As discussed in Section 6.3.3 and shown in Figure 6.9, the remaining Parameter Sweep Patterns are implemented using Data Iteration Patterns (transformation steps 3 and 5). Furthermore, Data Transfer and Transformation Patterns provide more low-level details for the original Simulation-oriented Data Exchange Patterns (steps 4 and 6). The data set of *Data Iteration Pattern 2* is the list of motion sequences of the current day, which is extracted from the overall list of daily routines. The operation corresponds to a service that carries out the bio-mechanical simulation using the Pandas software. This service may be implemented via a variant of the simulation workflow shown in Figure 5.8. Thereby, the pattern carries out one parallel instance of this operation for each motion sequence. As discussed in Section 4.3.2.2, the parallel operations are moreover distributed among several computers. Therefore, the resources parameter of the pattern points to a repository service delivering a list of available computers.

Afterwards, *Data Transfer and Transformation Pattern 1* adds implementation details to the data exchange from the bio-mechanical to the systems-biological model. The source data container of this pattern corresponds to the SQL database table of Pandas, while the set of CSV-based input files of Octave represent its target. Furthermore and since the SQL database table and CSV-based files have a well-defined structure, schema mappings are good candidates to describe low-level structural and algebraic dependencies between them [vS15b].

The set of CSV-based files being the target of Data Transfer and Transformation Pattern 1 corresponds to the data set of *Data Iteration Pattern 2*. The operation is a service that implements the systems-biological calculation via Octave and therefore gets the CSV-based files as input. Furthermore, another repository service delivers a list of resources, i. e., a list of available computers among which the systems-biological calculations shall be distributed. The data split parameter of the pattern moreover specifies how to partition its data set among these computers. This is based on a data-specific partitioning mode indicating the equal distribution of rows stored in the CSV-based input files.

Finally, *Data Transfer and Transformation Pattern 2* uses data-specific and ETL-specific parameter values to describe the data exchange from the systems-biological model back to the bio-mechanical model. So, its source is the set of CSV-based output files of Octave. These files are imported into the target of the pattern, i. e., into the SQL database of Pandas. In addition, schema mappings again define the dependencies between the source and the target.

6.5.2.3 Executable Workflow Fragments

Finally, rewrite rules of the last transformation steps 7 to 11 depicted in Figure 6.9 transform each basic data management pattern into an executable workflow fragment. These workflow fragments correspond to variants of individual parts of the coupling workflow depicted in Figure 5.9. For instance, Data Iteration Pattern 1 may be implemented via the outer loop iterating over the list of daily routines that has been previously loaded into a workflow variable. The workflow fragments realizing the other four basic patterns make use of workflow tasks that are similar to those embedded in the outer loop. More details on the resulting executable workflow are given by von Steht [vS15b].

6.6 Evaluation

The prototypical implementation illustrated above has again been the basis to evaluate the proposed pattern-based approach to simulation workflow design. As discussed in Section 4.4.3, the main goal of this pattern-based approach is to offer both the first and especially the fourth missing feature of currently available workflow systems summarized in Table 4.3. Section 6.6.1 therefore discusses how it offers the first missing feature, i. e., a domain-specific registry facilitating the search for data services and data resources. Section 6.6.2 correspondingly covers evaluation results regarding the fourth missing feature of an abstraction support for workflow design that is tailor-made for scientists. Note that the need for such an abstraction support also corresponds to the challenge discussed in Section 1.2.2, which is thus likewise considered in Section 6.6.2. The subsequent Sections 6.6.3 to 6.6.6 respectively discuss benefits or drawbacks of the proposed pattern-based approach regarding the remaining challenges illustrated in Sections 1.2.1 and 1.2.3 to 1.2.5.

6.6.1 Domain-specific Registry for Data Services and Data Resources

As illustrated in Section 4.4.2.1, the simulation artifact registry shown in Figure 6.1 is the major system component covering the first missing feature summarized in Table 4.3. It offers an interface that enables scientists to search for data services or data resources that match their particular requirements. Thereby, it provides scientists with the desired domain-specific and thus tailor-made means to submit corresponding queries. This is facilitated by the metadata describing simulation models, simulation methods, simulation software, as well as their dependencies to data services and data resources [vS15a]. In other words, scientists may search for data services or data resources by actually querying the simulation models, methods, or software they are more familiar with. For instance, they may query the concrete data resources or data containers storing the data that represent the mathematical input variables of their simulation models [vS15a].

Another means to search for proper data services or data resources in a domain-specific way is provided by the patterns themselves. In fact, the pattern-based representations of workflows, e. g., as depicted in Figures 1.2 and 6.8, conform to the vision of high-level workflow sketches proposed by Cohen-Boulakia and Leser [CBL11]. Scientists may use the pattern-based approach introduced in this chapter to design such workflow sketches as high-level descriptions of the simulation process they have in mind. The rule-based pattern transformer then uses the metadata managed by the simulation artifact registry to map individual patterns to concrete workflow fragments. These workflow fragments finally access the data services and/or data resources that support the requirements that have been previously claimed by scientists via the domain-specific patterns.

6.6.2 Multiplicity and Complexity of Low-Level Data Management Operations

The following subsections discuss to what extend the proposed pattern-based approach offers the four benefits listed in Section 1.2.2. The discussion regarding the first benefit, i. e., claiming for a reduced number of tasks to be specified in workflows, is covered by Section 6.6.2.1. Afterwards, Section 6.6.2.2 jointly discusses evaluation results regarding the second and third benefit. So, this concerns the question whether the proposed patterns constitute workflow building

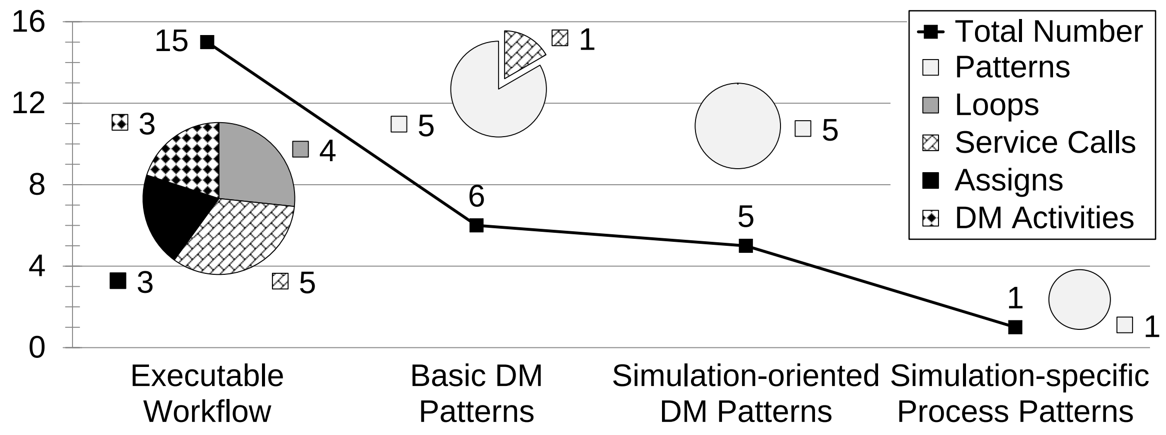


Figure 6.10: Number of workflow tasks to be specified for the coupling workflow at different abstraction levels shown in Figure 6.9; The partitioning among different types of workflow tasks is shown in pie charts; “DM” means “data management”; cf. [RSM14a].

blocks that are meaningful to scientists or to other persons involved in workflow design. Another aspect is whether the patterns allow for a domain-specific and thus easy parameterization. The focus of Section 6.6.2.3 is finally on the fourth benefit. Hence, it discusses whether the proposed patterns are sufficiently generic to be re-used in different simulation examples of various scientific domains.

6.6.2.1 Reduced Number of Workflow Tasks

Figure 6.10 summarizes the number of workflow tasks to be specified for the coupling workflow at different abstraction levels of the pattern hierarchy, as shown in Figure 6.9. So, this encompasses the levels of the executable workflow depicted in Figures 4.6 and 5.9, of basic data management patterns, of simulation-oriented patterns, and of the simulation-specific process pattern shown in Figure 6.8. Furthermore, pie charts in Figure 6.10 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 (1) simple *loop* constructs over (2) *service calls* and (3) BPEL `assign` activities reflecting local data processing steps to (4) complex *data management (DM) activities.*

If scientists were designing the executable coupling workflow depicted in Figures 4.6 and 5.9, they would need to define a total number of 15 workflow tasks. Furthermore, this would force them to specify many complex low-level details.

In particular, they would need to realize 6 sophisticated tasks for BPEL `assigns` and data management 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 [RSM14b].

As shown in Figure 6.10, the three pattern abstraction levels remove the burden from scientists to specify many and complex executable workflow tasks. Only the level of basic data management patterns includes one service call with moderate design complexity. Over and above, scientists may rely on a few number of abstract patterns to describe the whole coupling workflow. Especially the level of simulation-specific process patterns reduces the total number of workflow tasks from 15 original tasks to only one single pattern. Altogether and as indicated in Table 6.1, this constitutes a significantly higher reduction of the number of workflow tasks scientists have to specify than provided by related work.

6.6.2.2 Meaningful Workflow Building Blocks and their Domain-Specific Parameterization

A detailed analysis of the underlying pattern-based coupling workflows [Pie12, Rie14, vS15b] has been the basis to identify the numbers and complexities of pattern parameters to be specified at each individual abstraction level. Figure 6.11 summarizes the respective numbers of pattern parameters. Furthermore, it again uses pie charts to illustrate the partitioning of parameters according to their complexity, i. e., according to the parameter classes given in Figure 6.6. The class “others” summarizes a few parameters that are not covered by Figure 6.6. In the example, this encompasses simple parameters of Parameter Sweep Patterns or Data Iteration Patterns indicating a parallel execution of an operation.

The basic data management patterns summarized in Table 6.3 consider low-level use cases related to data transfers, data transformations, and data iterations. As discussed in Section 6.3.3 and depicted in Figure 6.5, they hence constitute workflow building blocks that are particularly meaningful to data engineers. Typically, these data engineers also have the necessary skills to cope with the high number and complexity of pattern parameters to be specified at this abstraction level. In fact, the basic patterns used to describe the coupling workflow mainly require low-level data- and ETL-specific parameter values, namely 13 of all 19. Note that scientists would usually be overwhelmed with this high number and especially with the high complexity of pattern parameters.

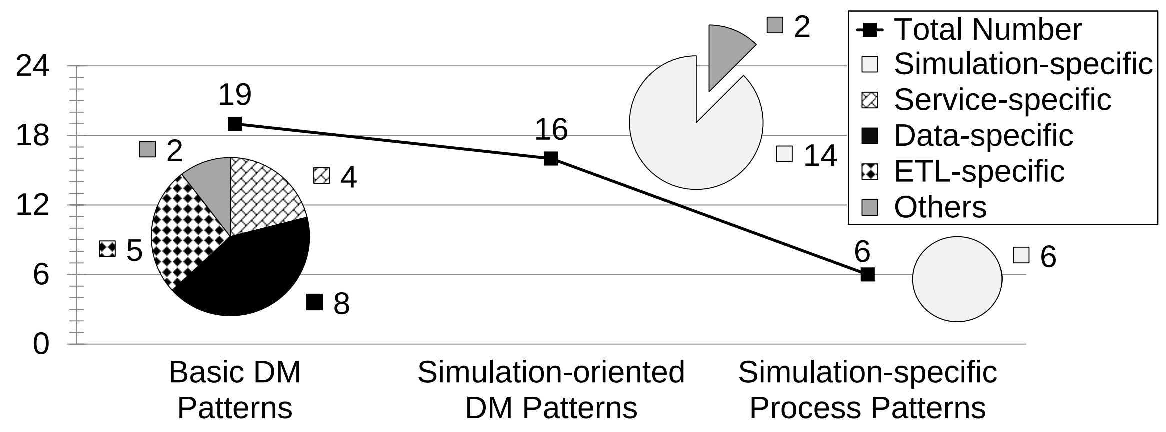


Figure 6.11: Number of pattern parameters to be specified for the coupling workflow at different abstraction levels shown in Figure 6.9; The partitioning among different classes of parameter values according to Figure 6.6 is shown in pie charts; “DM” means “data management”; cf. [RSM14a].

The simulation-oriented data management patterns summarized in Table 6.2 at least moderately reduce the total number of necessary parameters to 16. The most important advantage of these patterns is that they completely neglect low-level data- and ETL-specific parameter values. Instead, they mainly consider simulation-specific values that abstract from any implementation details and that are particularly suitable to be specified by scientists. The simulation-oriented patterns however focus on use cases related to data provisioning, data exchange, or parameter sweeps. On the one hand, these use cases are more common than the low-level use cases considered by basic data management patterns. On the other hand, simulation-oriented patterns still focus on data management issues. Hence and as shown in Figure 6.5, scientists might need help from other persons, e. g., simulation process experts, to combine these patterns in their workflows.

Simulation-specific process patterns are even more abstract and completely domain-specific. These high-level patterns are particularly meaningful to scientists conducting simulations, as they represent the main use cases they are interested in. For instance, the use cases and patterns supported by the prototype illustrated in Section 6.5 cover (1) the execution of simulation calculations based on a mathematical model, (2) the interpretation of simulation results, and (3) coupling two simulation models [Boh15, vS15b]. In the considered example, the simulation-specific process patterns even reduce the number of parameters scientists have to specify by a factor of around 3. Thereby, the remaining 6 parameters only exhibit simulation-specific values scientists are familiar with.

Especially the fact that the proposed patterns represent meaningful workflow building blocks is a unique selling point compared to related work (see Table 6.1). The simulation-specific process patterns and the accompanying reduced number and complexity of both workflow tasks and pattern parameters entail a considerable simplification for simulation workflow design. Furthermore, the distribution of these numbers and complexities among different abstraction levels of patterns complies with the skills of the respective persons associated with the levels in Figure 6.5. This finally highlights the suitability of the separation of concerns introduced by the proposed pattern hierarchy. In contrast to related work assessed in Table 6.1, this holistic separation of concerns provides a systematic framework to incorporate various skills of multiple kinds of persons. All this is also confirmed by Bohrn, who discusses similar evaluation results regarding the smaller example of the bio-mechanical simulation workflow [Boh14].

6.6.2.3 Generic Workflow Patterns

Another benefit of the approach is the generality of proposed patterns, which enables their seamless adoption in different simulation examples of various scientific domains. This generality of patterns has been investigated by conceptually applying them to the remaining examples listed in Section 3.1.

The simulation-specific process patterns depicted in Figures 1.2 and 6.8 may be used to describe the majority of process steps in all considered examples. This is because the use cases they represent are common for the simulation domain. The patterns for simulation calculations and result interpretations depicted in Figure 1.2 represent the most important high-level process steps in any simulation example. Moreover, novel applications increasingly rely on coupling different simulation models – as reflected by the pattern shown in Figure 6.8 – in order to produce more precise results [Gat14, GZC14, UGM14]. The model reduction example introduced by Fehr et al. is one of the rare applications that benefits from a new simulation-specific process pattern [FE11]. Figure 6.12 shows this pattern, its parameterization, and its transformation through the pattern hierarchy. The purpose of model reductions is to reduce the number of degrees of freedom in a numerical simulation model in order to speed up subsequent simulation calculations. So, the parameters of the pattern specify (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.

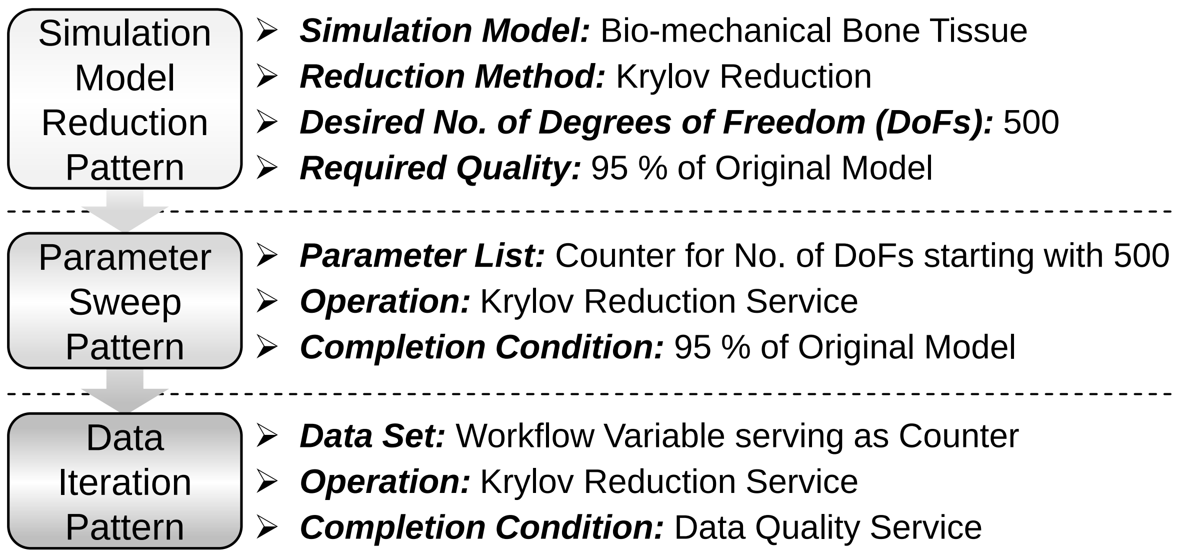

Simulation Model Reduction Pattern
Simulation Model: Bio-mechanical Bone Tissue
Reduction Method: Krylov Reduction
Desired No. of Degrees of Freedom (DoFs): 500
Required Quality: 95 % of Original Model
Parameter Sweep Pattern
Parameter List: Counter for No. of DoFs starting with 500
Operation: Krylov Reduction Service
Completion Condition: 95 % of Original Model
Data Iteration Pattern
Data Set: Workflow Variable serving as Counter
Operation: Krylov Reduction Service
Completion Condition: Data Quality Service

Figure 6.12: Patterns and their transformation to alleviate workflow design in the model reduction example described by Fehr et al. [FE11].

Both simulation-oriented data management patterns and basic data management patterns represent even more generic use cases related to data management. So, they may likewise be used in other examples listed in Section 3.1, e. g., to describe all process steps for data provisioning and data exchange. Thereby, the simulation-specific process patterns of these examples are mapped to the patterns at lower hierarchy levels in a similar way as depicted in Figures 6.3 and 6.9. This even holds for the new pattern of the model reduction example shown in Figure 6.12. Here, a Parameter Sweep Pattern and subsequently a Data Iteration Pattern may provide more low-level details of the Simulation Model Reduction Pattern. The *parameter list* of the Parameter Sweep Pattern corresponds to a counter for the number of degrees of freedom. This counter starts with the desired number specified for the Model Reduction Pattern and increments this number in each iteration. The *operation* is a service realizing the reduction method, e. g., based on Krylov subspaces [Bai02]. The *completion condition* represents the required quality of the reduced model, i. e., the reduction finishes when this quality is reached. The Data Iteration Pattern specifies the superordinate parameter list as a workflow variable serving as counter. While the operation parameter is not changed, the pattern provides more low-level details how to check the completion condition. This is based on a data quality service that validates the quality of the reduced model in each iteration [RBD+11, RBKK14]. The executable workflow finally realizing these patterns is described by Remppis [Rem11].

Simulation-oriented data management patterns and their parameters summarized in Table 6.2 are still tailored to computer-based simulations. Nevertheless, the basic data management patterns are completely generic. In fact, they may be adopted in any data provisioning or data exchange process, independent of the application area. This is because they are specified by service-, data-, or ETL-specific parameter values only (see Figure 6.6), which do not have a specific connection to any application area. This increased generality of the basic patterns especially makes it possible to adopt them in other kinds of data-intensive workflows classified in Figure 2.6 as well. Corresponding data analysis, data curation, or data integration workflows frequently comprise complex tasks for data transfers and data transformations [RLSR+06]. So, they may likewise benefit from the abstraction support provided by Data Transfer and Transformation Patterns. Moreover, data-intensive workflows often reflect data iterations. For instance and as discussed in Section 3.3.2, a Data Iteration Pattern may be used to express nearly the whole data analysis workflow illustrated by Wagner [Wag10]. Another example described by Ogasawara et al. analogously iterates over a list of files and carries out an analytical operation for each file [OdOV+11].

6.6.3 Diversity of Available Data Provisioning Techniques

The proposed pattern-based approach provides scientists with a small and reasonable set of different kinds of workflow building blocks and thus adequately meets the challenge discussed in Section 1.2.1. As depicted in Figure 6.5, these scientists usually only have to be aware of the simulation-specific process patterns and of the simulation-oriented data management patterns. The current set of patterns encompasses merely three simulation-specific patterns shown in Figures 1.2 and 6.8, as well as three simulation-oriented patterns summarized in Table 6.2. Furthermore, scientists are often quite familiar with the meanings of these patterns, as discussed above. Even the set of basic data management patterns, which are typically employed by data engineers, comprises only two different workflow building blocks as shown in Table 6.3.

Note that the pattern plug-in of the workflow design tool shown in Figure 6.1 is nevertheless designed to be seamlessly extensible by additional kinds of patterns [Pie12]. This may for instance be necessary to add support for the Simulation Model Reduction Pattern shown in Figure 6.12. However, the set of available patterns should always be concise in order not to overburden scientists.

Table 6.4: Overhead of the rule-based pattern transformation for its worst-case duration of 0.49 seconds; cf. [RSM14a]

Number of tuples	1 million	10 million	100 million
Workflow duration	140 seconds	1,410 seconds	14,095 seconds
Worst-case overhead	0.35 %	0.035 %	0.0035 %

6.6.4 Efficient Data Processing and Optimization

The pattern-based approach to simulation workflow design may influence the efficiency of data processing in two ways. Firstly and as discussed in Section 6.4.2, the transformation of patterns into executable workflows may cause an overhead when applied at runtime of workflows. Secondly, the pattern-based approach may also exploit optimization techniques to make data processing in simulation workflows more efficient. The following subsections discuss these two aspects.

6.6.4.1 Overhead Caused by Pattern Transformation

The overhead of the rule-based transformation of patterns has been evaluated by conducting a set of experiments based on the prototype illustrated in Section 6.5 [RSM14a]. Thereby, the prototype ran on a 64 bit Windows Server 2008 system with 32 GB RAM and an Intel Xeon CPU with 8 cores. The experiments have been used to measure the total duration of the eleven transformation steps depicted in Figure 6.9. This total duration has been calculated as the average among the results of 100 consecutive experiment runs. To assume a worst-case scenario, the prototypical implementation checks 100 rules for each of the eleven transformation steps before the 100th rule is actually applied. The resulting worst-case duration of the pattern transformation is 0.49 seconds.

Table 6.4 compares this worst-case duration of the pattern transformation with the typical duration of a simulation workflow. This typical duration of a simulation workflow has been determined by measuring the duration of executing the coupling workflow shown in Figure 5.9 for one daily routine. The workflow and the simulation services it calls ran on seven computers, i. e., one for the workflow system, three for the Pandas software, and three for Octave. Each computer has been equipped with the system resources described above. The workflow has been executed 10 times and for a varying number of tuples as Pandas typically stores them in its SQL database, i. e., from one million to 100 million tuples. As shown in Table 6.4, the average workflow duration correspondingly ranges from 140 to 14,095 seconds. So, the maximum overhead caused by the pattern transformation for its worst-case duration of 0.49 seconds is only 0.35 %. This constitutes a negligible overhead compared to the workflow duration.

6.6.4.2 Integration of Optimization Techniques

The rule-based approach for pattern transformation furthermore enables a seamless integration of corresponding rule-based optimization decisions [VSS+07, OdOV+11]. As discussed in Section 6.4.1, rewrite rules may this way find efficient workflow fragments realizing given patterns. For instance, Kalyoncu discusses how to integrate some of the optimization rules proposed by Vrhovnik et al. into rewrite rules used for the running example of a bone simulation [VSS+07, Kal15]. His measurements reveal that this may likewise induce significant performance improvements for simulation workflows.

On this note, rewrite rules may choose efficient realizations of patterns that rely on scalable data processing facilities, e. g., based on MapReduce or cloud computing technologies [DG08, LQ14]. This may be beneficial or even necessary in case the size of data to be processed by a workflow exceeds a certain threshold. It is for instance crucial if the number of tuples generated by Pandas goes beyond 100 million. Here, Gessler shows that a MapReduce-based implementation of parts of the coupling workflow depicted in Figure 5.9 considerably accelerates workflow execution [Ges14]. Thereby, MapReduce is especially suitable for the ETL processes depicted in Figure 3.2 that filter and aggregate data [LTP11]. These ETL processes correspond to the Data Transfer and Transformation Patterns shown in Figure 6.9, which may thus be mapped onto such efficient realizations. Furthermore, the Data Iteration Pattern shown in Figure 3.5 resembles some general ideas of MapReduce and may thus also be implemented via this approach.

6.6.5 Data Quality

In a similar way, rewrite rules transforming patterns may also consider requirements related to data quality. Thereby, the parameterizations of certain patterns may be amended by descriptions of such quality demands [RBD+11, RBKK14]. An example is the required quality of a reduced simulation model specified for the Simulation Model Coupling Pattern shown in Figure 6.12. As indicated by the Data Iteration Pattern in this figure, this quality demand is finally assured by a data quality service [RBKK12, RBKK14]. The rewrite rules used for the transformation steps shown in Figure 6.12 may choose such data quality services as parts of the final realizations of patterns.

Nevertheless, the optimization of data quality is not only limited to choosing proper data quality services. In some scenarios, a more obvious issue is to select proper data resources or data containers storing data that meet relevant quality demands. Furthermore, the operations a workflow later carries out on the selected data may influence the final data quality as well. This is especially true for operations like data filters or aggregations. Such issues may likewise be reflected by rewrite rules. Typically, this is done in rules that map simulation-oriented data management patterns onto basic data management patterns. The reason is that simulation-oriented patterns abstractly specify relevant data and operations. In contrast, basic patterns define more concrete details about data and operations, and thus also bring in characteristics related to data quality.

6.6.6 Monitoring and Provenance Support

While the proposed pattern-based 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 rule-based pattern transformation. This leads to a missing correlation between (1) patterns visible to scientists in a workflow design tool and (2) audit or provenance information captured by an execution environment [CVDK$^+$12]. So, scientists might be confused when they monitor workflow executions or try to reproduce simulation runs. Workflow systems have to be extended by mechanisms that aggregate audit and provenance information to recover their correlation to patterns.

A possible solution to this problem may be a view concept on workflows [SGK$^+$11]. After each application of a rewrite rule during the pattern transformation, the input pattern is not replaced by the resulting workflow fragment. Instead, the pattern is defined as view on the fragment. Such views directly enable the necessary aggregation of audit or provenance information for individual patterns [SGK$^+$11]. Furthermore, a corresponding view expansion may allow scientists to look into specific low-level details of data provisioning tasks if this is required for monitoring or reproducibility purposes.

6.7 Summary and Future Work

Simulation workflows frequently involve multiple, highly complex data provisioning and data exchange tasks, particularly when they are coupled across different scientific domains. In order that scientists are able to concentrate on their core issue, namely on the simulation itself, an abstraction support for this complex data management is indispensable. As summarized in Table 6.1, related work from several research areas do not comprehensively offer all four benefits listed in Section 1.2.2. These benefits are however crucial for an adequate abstraction support that is especially tailor-made for scientists conducting simulations.

The novel pattern-based approach to simulation workflow design introduced in this chapter exactly fills this gap and supports all four benefits in a consolidated fashion. Scientists only need to select a few number of abstract, simulation-specific process patterns to describe the high-level simulation process they have in mind. These simulation-specific patterns are especially meaningful to scientists and allow for their domain-specific and thus easy parameterization. A transformation approach based on a set of rewrite rules makes it possible to map such high-level process models and patterns onto executable simulation workflows. As another major contribution, a pattern hierarchy with different, clearly distinguished abstraction levels facilitates a holistic separation of concerns for workflow design. It offers a systematic framework to seamlessly incorporate specific skills of various kinds of persons, e. g., not only scientists, but also data engineers.

A prototypical implementation and its application to several simulation examples, e. g., to bio-mechanical and systems-biological problems, has served as basis to evaluate this approach. In particular, it turned out that the patterns significantly alleviate simulation workflow design and thereby make it especially suitable for scientists. The generality of patterns furthermore enables their seamless adoption in various simulation domains. Altogether, this represents the first principled approach that adequately conquers the complexity of data provisioning and data exchange inherently associated with simulation workflows.

Future work may even increase the generality of patterns and of the whole approach by applying them to other application areas than simulations, e. g., to other workflow classes shown in Figure 2.6. Another major aspect is to further investigate the potential of integrating optimization decisions into rewrite rules to increase both the efficiency of workflows and data quality. Note that this again goes hand in hand with possible future work discussed in Sections 4.5 and 5.6.

Chapter 7

An Approach to Optimize Workflow-Local Data Processing

Various optimization approaches exist to improve the performance of data-intensive workflows. Some of these approaches focus on selecting efficient services or resources realizing certain workflow steps [Ley05, WCL$^+$05]. These services or resources may furthermore be based on scalable and parallel data processing facilities, such as MapReduce-based systems or cloud computing technologies [DG08, ZBKL10, LTP11, LQ14]. Other optimization approaches represent rule-based techniques for re-structuring workflow models [BHW$^+$07, VSS$^+$07, OdOV$^+$11]. On this note, different kinds of parallelism may be reflected in workflow models, e. g., different kinds of task, data, and pipeline parallelism [PA06]. As discussed in Sections 5.5.4 and 6.6.4, most of these optimization approaches are also applicable to simulation workflows – and especially to the concepts and methods introduced by this thesis.

The majority of existing optimization approaches is targeted at huge amounts of data that are processed externally to a workflow or to a workflow system. So, they are related to the concepts of data services and data management activities depicted in Figure 4.2, but not to local data processing in workflows. Furthermore, many optimization approaches make use of dataflow-oriented descriptions of workflows, since a dataflow usually facilitates various kinds of optimizations (see also Section 2.3.2). However, optimization approaches for local data processing in workflows that are described by control-flow-oriented workflow languages have largely been neglected in previous work. This is underpinned by a related work study conducted by Müller and Wagner [Mül10, Wag11]

According to guideline 1 discussed in Section 4.3.2.1, it is sometimes inevitable or even appropriate to process data locally in the context of simulation workflows. The major scenarios calling for a local data processing are represented by the Data Iteration Patterns shown in Figures 3.5 and 3.6. For instance, the coupling workflow depicted in Figures 4.6 and 5.9 needs to load several external data sets, e. g., lists of daily routines or of available computers. It then processes these data sets locally in the workflow context to carry out sophisticated data iterations.

Moreover, Figure 2.7 shows that simulation workflows are combinations of both data-intensive and orchestration (sub-)workflows. Orchestration workflows usually rely on control flow as the main concept for workflow languages [LR00] (see Section 2.3.2). In this spirit, this thesis considers conventional control-flow-oriented workflow languages as the major kinds of languages to describe simulation workflows [GSK$^+$11]. The main reason is that this leads to a generic, standard-based approach for all phases of a simulation workflow depicted in Figure 2.7. This generic approach likewise facilitates multi-scale simulations that are coupled across different scientific domains [RSM14a]. Furthermore, it leads to a seamless workflow design environment that removes the burden from scientists to get accustomed to many diverse design tools or technologies.

As conclusion, simulation workflows require a novel optimization approach targeted at local data processing in workflows that are described by control-flow-oriented languages. This thesis fills this gap in current research by introducing a new tailored approach, as discussed in Section 1.3.3. This new approach improves both the performance and robustness of local data processing tasks in simulation workflows [RSM11]. The following sections discuss both the need for such an approach and its detailed contributions as follows:

- Section 7.1 discusses the state of the art regarding local data processing via control-flow-oriented workflow languages. It firstly illustrates the major kinds of local data processing tasks that are typically supported by such workflow languages, e. g., variable assignments or expression evaluations for control flow decisions. It then details the data-intensive protein modeling workflow also listed in Section 3.1 [Wag10, Wag11], which is used as main use case throughout the rest of this chapter.[1] Furthermore, it discusses both the means and the limitations of currently available control-flow-based workflow systems to support relevant local data processing tasks.

[1]Note that the protein modeling workflow is preferred here to the bone simulation, because its data-intensive nature is more appropriate for the relevant illustrations (see Section 3.1).

- Afterwards, Section 7.2 introduces the new approach proposed by this thesis. It therefore shows how to extend the typical architecture of control-flow-based workflow systems to transparently improve local data processing in workflows. The proposed extensions of workflow systems introduce various techniques that partition local data processing tasks to be performed during workflow execution in a smart way. Thereby, data processing tasks are either assigned to the workflow execution engine or to a tightly integrated local database engine. This allows for additionally exploiting the efficient and robust data processing capabilities of mature database systems.
- In the course of working on this thesis, the effectiveness of the introduced techniques has been evaluated by a set of experiments. These experiments are based on various test scenarios, including both smaller test workflows and the above-mentioned protein modeling workflow. Section 7.3 illustrates the prototypical implementation used for these experiments, and it discusses the corresponding evaluation results. This discussion especially comes up with possible indicators when to use either the workflow execution engine or the local database engine for particular data processing tasks. Furthermore, it shows the great potential offered by the proposed approach for improved performance and robustness of local data processing in workflows.

Finally, Section 7.4 summarizes the major aspects and lists possible future work. This chapter is a revised version of a previous author publication [RSM11].

7.1 Local Data Processing in Workflows

The local data processing in control-flow-oriented workflow languages is usually based on accessing and modifying workflow variables and their contents. For instance, the workflow language BPEL makes use of XML-based variable contents to represent the internal state of a particular workflow instance [OAS07] (see Section 2.3.3). The following Section 7.1.1 summarizes major kinds of local data processing tasks that are commonly supported by control-flow-oriented workflow languages. Section 7.1.2 then exemplifies these kinds of tasks via the data-intensive protein modeling workflow [Wag10, Wag11]. Afterwards, Section 7.1.3 discusses architectural aspects of control-flow-based workflow systems, with a special focus on how these systems usually support local data processing tasks in workflows.

7.1.1 Major Kinds of Local Data Processing Tasks

Conventional control-flow-oriented workflow languages typically support the following major kinds of local data processing tasks [LR00, OAS07, Wes12]:

- The first major kind is represented by *variable assignments*. These are certain workflow tasks that modify the contents of workflow variables, e. g., to assign initial values to a variable or to copy values between several variables. For instance, BPEL supports such scenarios via the `assign` activity. As illustrated in Section 2.3.3, this activity offers many possibilities to carry out complex variable modifications. This mainly includes several specialized built-in functions, as well as generic and powerful possibilities using expression or query languages such as XPath.
- Furthermore, workflows often access their variables to make state-based *control flow decisions*. These control flow decisions determine which of a set of succeeding control flow branches are to be executed depending on the current state of the workflow instance. This is usually based on Boolean expressions that are evaluated on the contents of relevant variables. As example, a BPEL workflow may embed XPath expressions into transition conditions of the `flow` activity. Other examples are the `if` activity or loop-based workflow constructs.
- Finally, *service calls* or other messaging tasks are the common way to receive or to load data from external partners or resources, as well as to send or to store data back. For instance, the coupling workflow depicted in Figures 4.6 and 5.9 comprises several service calls to load some data from external repositories. BPEL mainly supports this scenario via `invoke`, `receive`, or `reply` activities, and via certain event handlers.[2]

7.1.2 Data-Intensive Protein Modeling Workflow

Figure 7.1 shows the activities and their control flow of a workflow for protein modeling [Wag10, RSM11, Wag11]. This workflow is a data analysis workflow according to the classification given in Figure 2.6. It identifies or investigates certain

[2]Note that also the data management activities of SIMPL introduced in Chapter 5 may be used to load external data and to store it back. Nevertheless, the way these activities access local workflow variables is very similar to service calls or messaging tasks [HSR^{+}10, Mül10, Wag11]. So, data management activities are not covered separately in this chapter.

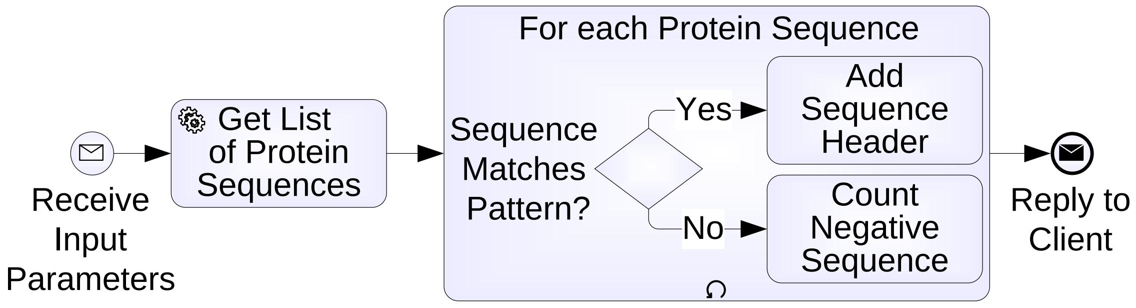


Figure 7.1: Protein modeling workflow; cf. [RSM11].

proteins that are able to solve specific chemical or biological problems [BTS07]. Therefore, it uses pattern matching to find important regions within individual protein sequences or families. For instance, it may try to find amino acids that are relevant for certain chemical reactions.

The protein modeling workflow processes most of its data directly within the workflow context, i. e., locally in workflow variables. Thereby, it comprises all major kinds of local data processing tasks listed in Section 7.1.1. Furthermore, it embeds all these tasks into a complex data iteration, which especially entails challenges regarding an efficient data processing in the workflow [Vhr11, Kal15]. The size of the locally processed data may range from about a hundred KBs to a few hundred MBs [Wag10, Wag11].

The workflow firstly gets some input parameters from the client, e. g., an identifier for the data source storing the protein sequences to be investigated. An example data source is a common protein database offering access to its data via a service interface, such as GenBank [BKML$^+$10]. The workflow accordingly uses a service call to retrieve a list of relevant protein sequences from this database and to store the list into a local workflow variable. This workflow variable is based on an XML schema, where individual protein sequences are listed as character strings [Wag10, Wag11]. The subsequent loop iterates over the list of protein sequences stored in the variable. Thereby, an embedded control flow decision searches for a certain pattern within each protein sequence, e. g., via a regular expression. If a protein sequences matches the pattern, the workflow adds the header of the sequence to a list of headers stored in another workflow variable. Otherwise, it increments a counter for negative sequences. These two local data processing tasks are realized using variable assignments. After having processed all protein sequences in this way, the workflow sends both the list of headers of positive sequences and the number of negative sequences back to the client.

7.1.3 Current Architecture for Local Data Processing in Control-Flow-Based Systems

Figure 7.2 sketches the main components of the typical architecture of control-flow-based workflow systems. For better readability, it leaves out components that are not directly relevant for the local data processing in workflows. This for instance concerns a component to deploy workflow models [LR00, GSK$^+$11].

A workflow execution environment usually not only contains a *workflow execution engine*, but also a *local database system*. Remark that this is not the type of external database system a workflow may access via data services or data management activities (see Figure 4.2). In fact, the database system shown in Figure 7.2 supports local data processing tasks embedded into workflows. For that purpose, it mainly serves as persistent data store for the contents of workflow variables – and also for metadata such as auditing information [Mül10].

However, the local database system does not process the data stored in workflow variables, e. g., during the local data processing tasks listed in Section 7.1.1. Instead, the workflow execution engine itself performs this local data processing. It therefore contains a *pool of variables* that manages all workflow variables and their contents. This pool may for instance be realized as a heap of Java objects representing XML data. Furthermore, an *expression evaluation engine* carries out expressions for variable assignments or control flow decisions, e. g., based on XPath. A *data processing logic* component implements the execution logic of different kinds of local data processing tasks, e. g., of the kinds listed in Section 7.1.1. Thereby, it also controls how the pool of variables and the expression evaluation engine are employed during individual tasks.

A *persistence manager*, amongst others, controls the persistence of the data stored in workflow variables. So, it stores and loads the variable contents into or from the local database either automatically or when the execution engine triggers it. A possible solution to such a persistence manager is a Data Access Object (DAO) layer, e. g., as offered by Hibernate[3]. To manage the contents of workflow variables, the local database contains a *pool of variables* as well. The persistence manager provides means to map the variables in the pool of the workflow execution engine between those in the pool of the database. This way, the workflow execution engine is independent of the concrete local database system, i. e., it is possible to change the database system quite easily [Mül10].

[3]Hibernate: `http://hibernate.org/`

The local *database management system* also contains a *query / expression execution engine*. However, this engine and its mature, efficient, and robust data processing capabilities are not exploited during workflow execution. The following section therefore introduces a novel approach that better makes use of this component in order to improve data-intensive local data processing tasks.

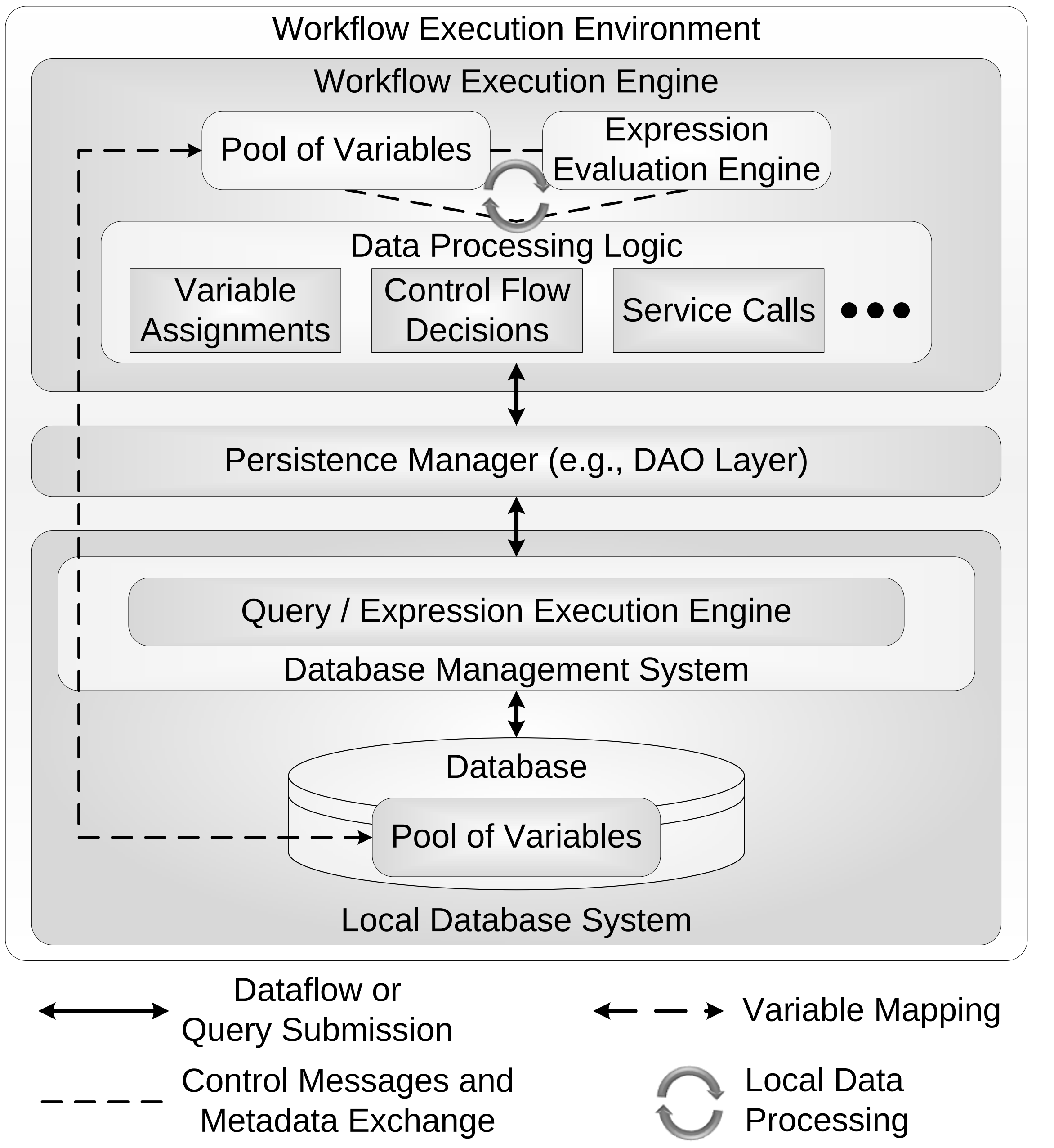

Workflow Execution Environment
Workflow Execution Engine
Pool of Variables
Expression Evaluation Engine
Data Processing Logic
Variable Assignments
Control Flow Decisions
Service Calls
Persistence Manager (e.g., DAO Layer)
Query / Expression Execution Engine
Database Management System
Database
Pool of Variables
Local Database System
Dataflow or Query Submission
Variable Mapping
Control Messages and Metadata Exchange
Local Data Processing

Figure 7.2: Common architecture for local data processing in workflow systems. The database system is often only used as persistent data store; cf. [RSM11].

7.2 Novel Approach to Improve Local Data Processing in Workflows

The following Section 7.2.1 illustrates how the optimization approach proposed by this thesis extends the architecture of a workflow system shown in Figure 7.2. Afterwards, Section 7.2.2 introduces various techniques that make use of these architectural extensions in order to improve the different kinds of local data processing tasks listed in Section 7.1.1. The purpose of Section 7.2.3 is to discuss the resulting optimization potential to increase both the performance and robustness of local data processing in workflows.

7.2.1 Extended System Architecture

Figure 7.3 highlights the proposed extensions of the architecture of a workflow system. These extensions rely on a reasoned design principle in that they only introduce two additional system components: the data processing optimizer and the query / expression interface. This entails a small implementation overhead to extent concrete workflow systems accordingly [Wag11]. The new components make use of other, already existing components in a smart way in order to ensure an efficient and robust local data processing in workflows. Thereby, local data processing tasks to be performed during workflow execution may be partitioned between the workflow execution engine and the local database system. This way, it is possible to exploit their respective capabilities as effectively as possible. Note that this kind of optimization is transparent to workflow models, i. e., it does not change these models, but only influences their execution.

The *query / expression interface* enables to push down local data processing tasks from the workflow execution engine to the local database system. So, it offers the means to exploit the efficient and robust data processing capabilities of the integrated database management system. Thereby, the execution engine only issues queries or expressions against the database system. This system likewise answers with only small data items, e. g., Boolean values needed for control flow decisions. Note that corresponding queries to carry out local data processing tasks may vary between different kinds of database systems, e. g., due to varying SQL dialects. Hence, the query / expression interface has to maintain query templates for certain kinds of database systems [Wag11]. This way, the workflow execution engine is again independent of the concrete local database system.

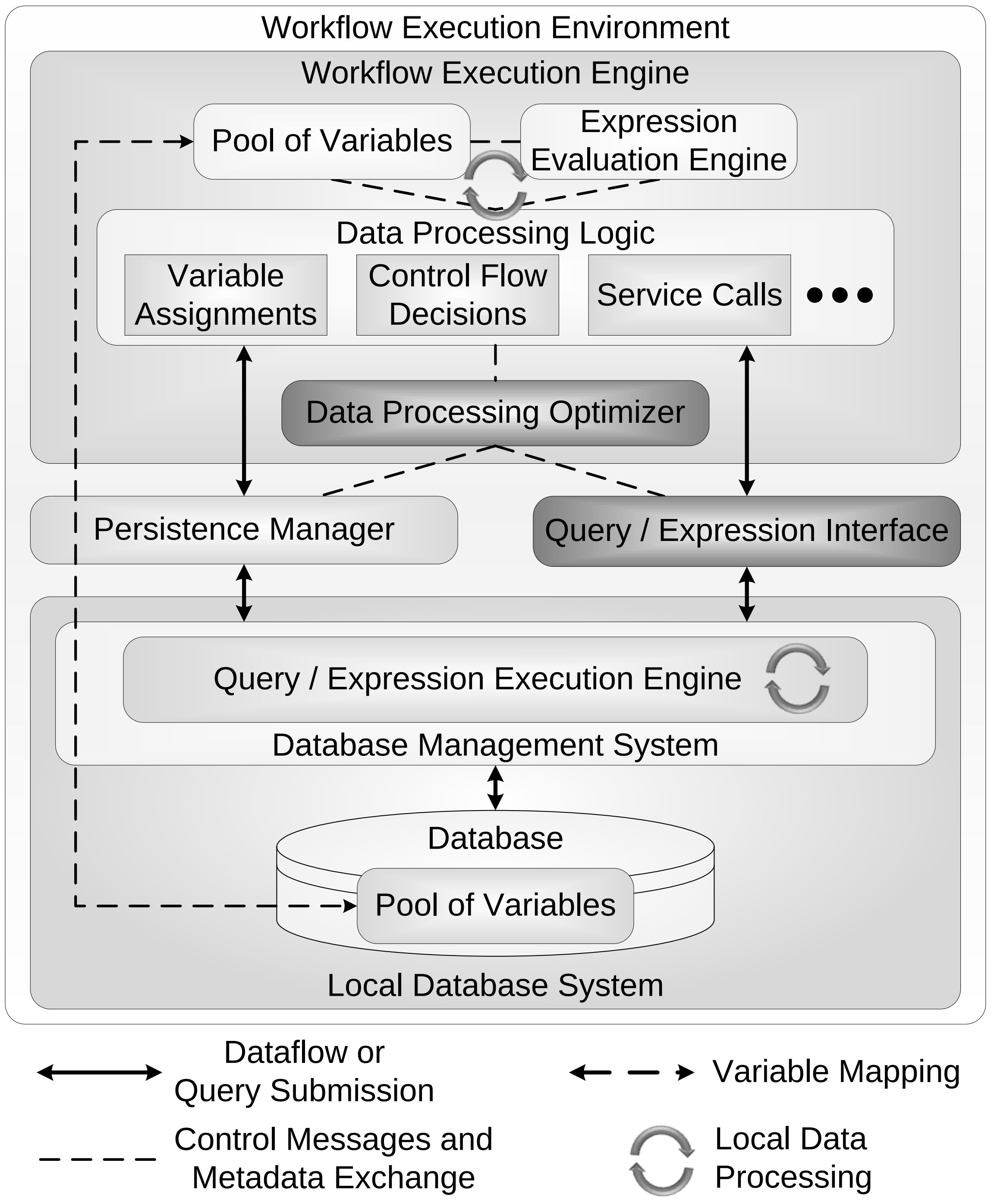

Workflow Execution Environment
Workflow Execution Engine
Pool of Variables
Expression Evaluation Engine
Data Processing Logic
Variable Assignments
Control Flow Decisions
Service Calls
Data Processing Optimizer
Persistence Manager
Query / Expression Interface
Query / Expression Execution Engine
Database Management System
Database
Pool of Variables
Local Database System
Dataflow or Query Submission
Variable Mapping
Control Messages and Metadata Exchange
Local Data Processing

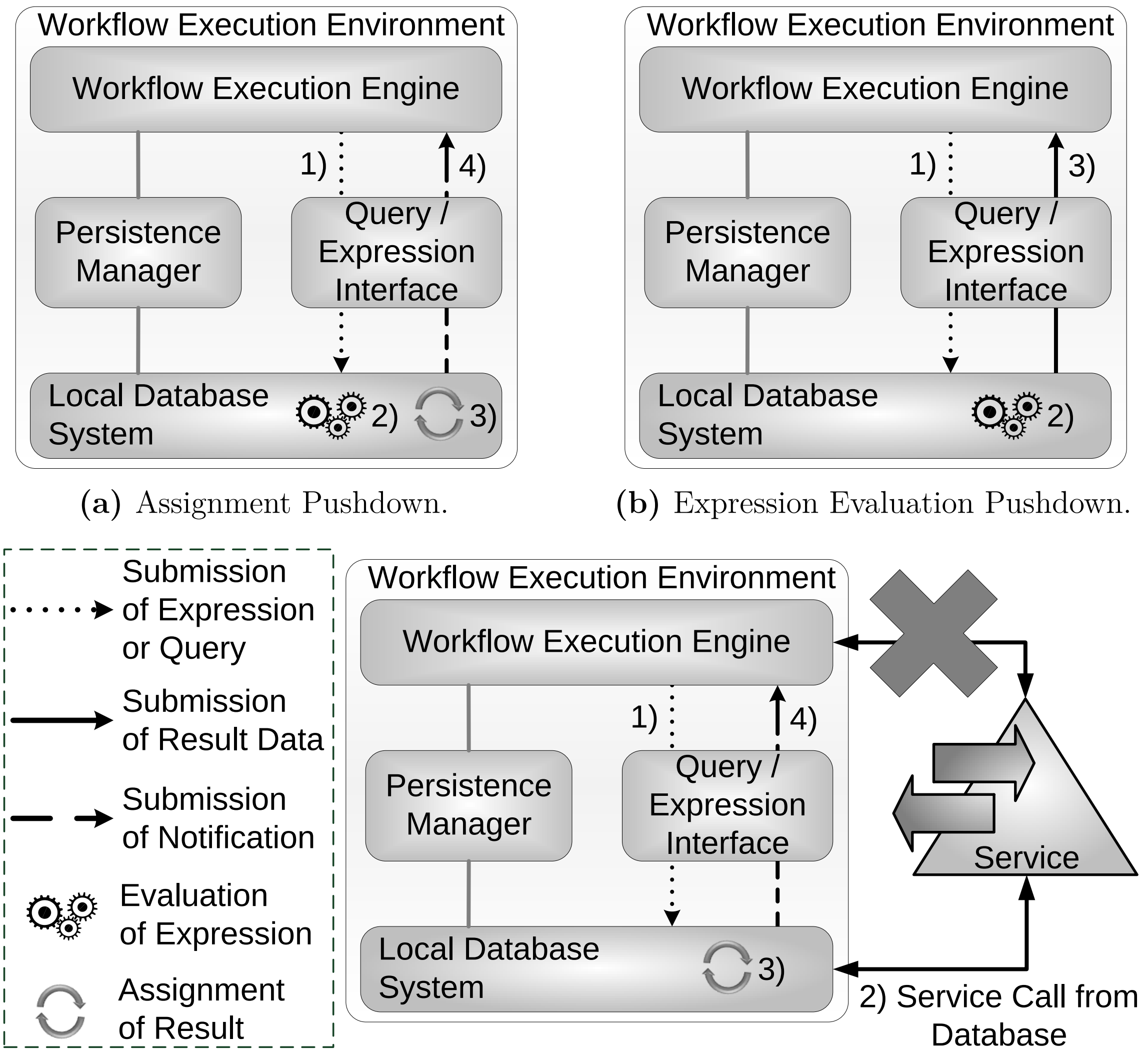
Figure 7.3: Extended architecture for an improved local data processing in workflow systems. Especially the new components *data processing optimizer* and *query / expression interface* (dark-gray) allow for exploiting the database system more intensively during local data processing tasks; cf. [RSM11].

The *data processing optimizer* is directly embedded into the workflow execution engine in order to effectively control the data processing logic. In particular, the optimizer decides whether local data processing tasks are to be executed by the workflow execution engine or by the local database system. In analogy, it determines which of both components acts as primary storage of the corresponding data of workflow variables. Thereby, the data processing optimizer not only tries to exploit the mature data processing capabilities of the database system whenever this is appropriate. It additionally aims at minimizing the amount of data to be transferred between the execution engine and the database. Its optimization decisions hence mainly depend on (1) the capabilities of the execution engine and of the database system, (2) the concrete data sizes, and (3) the complexities of involved queries or expressions [Wag11]. These characteristics are described by metadata or statistics that are provided by other system components.

7.2.2 Techniques to Improve Local Data Processing

The proposed architectural extensions support various techniques to effectively exploit the local database system during the execution of local data processing tasks. Figure 7.4 depicts the major kinds of these techniques, which also reflect the different kinds of local data processing tasks listed in Section 7.1.1 [Wag11]. The *Assignment Pushdown* depicted in Figure 7.4a shifts the responsibility for executing variable assignments from the workflow execution engine to the local database system. The execution engine firstly issues the query or expression specified for the assignment against the database (step 1 in Figure 7.4a). Afterwards, the database system evaluates this assignment expression (step 2) and assigns the expression results to the target variables stored in its own pool of variables (step 3). Finally, it returns a notification to the execution engine (step 4), which indicates whether the assignment has been executed successfully.

The *Expression Evaluation Pushdown* is targeted at the second kind of local data processing tasks listed in Section 7.1.1, i. e., at control flow decisions. As shown in Figure 7.4b, it again lets the local database system evaluate expressions that are specified for relevant control flow decisions (steps 1 and 2). In contrast to the Assignment Pushdown, the database system synchronously returns the expression results to the workflow execution engine (step 3). These expression results are however only small data items, e. g., Boolean values. The execution engine inevitably needs such results to properly make its control flow decisions.


Workflow Execution Environment
Workflow Execution Engine
1)
4)
Persistence Manager
Query / Expression Interface
Local Database System
2)
3)
Workflow Execution Environment
Workflow Execution Engine
1)
3)
Persistence Manager
Query / Expression Interface
Local Database System
2)
Submission of Expression or Query
Submission of Result Data
Submission of Notification
Evaluation of Expression
Assignment of Result
Workflow Execution Environment
Workflow Execution Engine
1)
4)
Persistence Manager
Query / Expression Interface
Local Database System
3)
Service
2) Service Call from Database

(a) Assignment Pushdown.

(b) Expression Evaluation Pushdown.

(c) Service Call Pushdown.

Figure 7.4: Techniques to improve local data processing tasks. The numbers indicate the orders of individual processing steps; cf. [RSM11, Wag11].

Using these two techniques, the local database system only returns notifications or small data items to the workflow execution engine. This may lead to performance improvements, because less amounts of data need to be transferred between both system components. However, the input data of the expressions used for variable assignments or control flow decisions should then be available in the local database system before it evaluates these expressions. Otherwise, the persistence manager would need to transfer these data from the execution engine to the database system. This may in turn lead to a performance degradation, especially since such input data may be big in case of data-intensive workflows.

In control-flow-oriented languages, service calls or other messaging tasks are the major kinds of workflow tasks receiving data that eventually has to be stored in the local database system as described above [LR00, Wes12]. So, such workflow tasks correspond to the third kind of data processing tasks listed in Section 7.1.1. This is where the *Service Call Pushdown* comes into play, as shown in Figure 7.4c. Here, the workflow execution engine does not call the relevant service. Instead, it asks the local database system to call the service on its own (steps 1 and 2), e. g., via a user-defined function (UDF) [VSS+07]. This way, the result data of the service is directly stored in the database (step 3), i. e., without the indirection over the execution engine and the persistence manager. Finally, the database system again notifies the execution engine about the successful execution of the service call (step 4). Note that the same principle can be applied for asynchronous communication, i. e., when a workflow sends data to services or when it gets data from them, without expecting any result on either side [Wag11].

7.2.3 Optimization Potential for Local Data Processing

In previous control-flow-based workflow systems, only the workflow execution engine could carry out the local data processing tasks listed in Section 7.1.1. The architectural extensions shown in Figure 7.3 add a new option to shift this kind of data processing to the local database system. This offers a great potential for improved performance and robustness of executing local data processing tasks in data-intensive workflows. One reason is that the mature database technology is typically able to deal with large amounts of data in an efficient way. Furthermore, it provides reliable means to cope with complex and repeating data management operations that are involved in data-intensive workflows. This for instance concerns sophisticated data iterations, e. g., as occurring in the protein modeling workflow shown in Figure 7.1. Moreover and as discussed above, the techniques depicted in Figure 7.4 may help to reduce the amount of data that has to be transferred between the execution engine and the local database system.

The approach proposed in this chapter also entails a benefit when multiple workflow instances or data management operations run concurrently. In this case, the execution of local data processing tasks within the workflow execution engine typically leads to an increased main memory consumption of this engine. It may thus get too busy or even overloaded. The techniques depicted in Figure 7.4 make it possible to push parts of workflow processing to the local database system.

This may lead to a more balanced utilization of all components of the workflow system. In particular, it leads to a more efficient overall main memory utilization, which in turn may entail a higher throughput of concurrent workflow instances.

Nevertheless, the workflow execution engine is more flexible regarding internal data structures to store the contents of workflow variables in main memory. It may thus rely on customized internal data structures that are especially tailored to the local data processing tasks listed in Section 7.1.1. Hence, the efficient and robust capabilities offered by the local database system often only take effect with increased data sizes and complexities of data management operations [Wag11]. For instance, most kinds of orchestration workflows (see Figure 2.6) are usually less data-intensive and work with smaller data sets. Such workflows often do not benefit from the capabilities of the local database system. In such a case, the data processing optimizer shown in Figure 7.3 may decide to leave the responsibility for local data processing to the workflow execution engine.

7.3 Prototype and Evaluation

The arguments discussed in Section 7.2.3 regarding improved performance and robustness for data-intensive workflows have been evaluated via a set of experiments. These experiments especially aimed at assessing the techniques to improve local data processing tasks depicted in Figure 7.4. The following Section 7.3.1 presents the prototypical implementation, the experimental setup, and the test scenarios used for the experiments. Afterwards, Section 7.3.2 discusses the corresponding experimental results. One goal of this discussion is to exemplify the optimization potential of the approach proposed in this chapter. Furthermore, it comes up with possible indicators when to use either the workflow execution engine or the local database system for local data processing tasks.

Note that the optimization approach proposed in this chapter is completely transparent to workflow models. More precisely, it does not require changing workflow models, but only influences their execution. Hence, there is no relation between this approach and the challenges introduced in Sections 1.2.1 and 1.2.2, which focus on workflow design issues. Dependencies to data quality or monitoring and provenance issues illustrated in Sections 1.2.4 and 1.2.5 are negligible as well. So, the evaluation covered in this section only focuses on the challenge introduced in Section 1.2.3, i. e., calling for an efficient data processing in workflows.

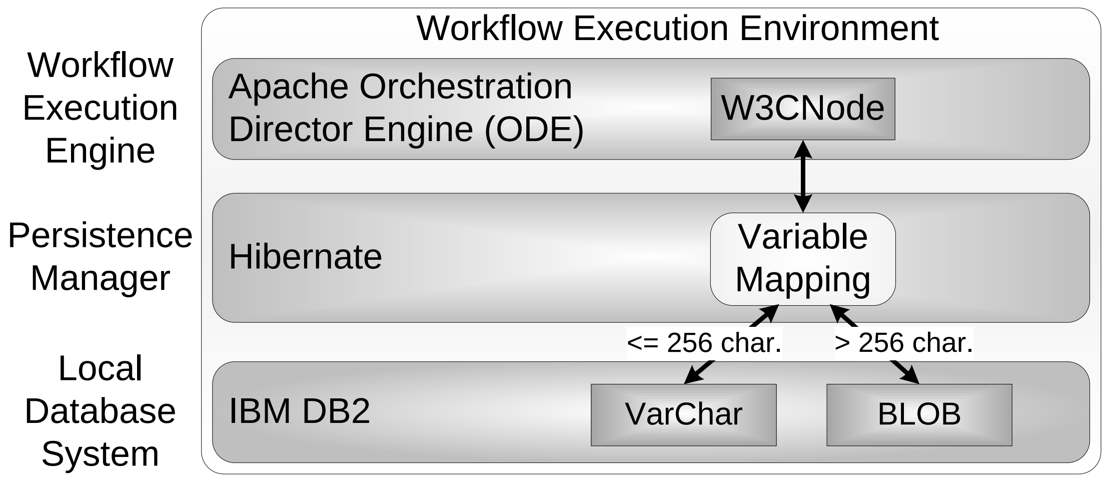
7.3.1 Prototype, Experimental Setup, and Test Scenarios

The evaluation has been backed up by two separate BPEL-based prototypes of a workflow execution environment. These prototypes have been developed in the course of several student projects that have accompanied the work on this thesis [Mül10, Wag11, Age12]. The first one depicted in Figure 7.5a reflects the current architecture of workflow systems as shown in Figure 7.2. The second prototype depicted in Figure 7.5b implements the most relevant parts of the architectural extensions illustrated in Figure 7.3. So, these two prototypes may be compared with each other in order to evaluate the proposed architectural extensions. Both rely on Apache ODE Version 1.3.4 as workflow execution engine. Furthermore, they make use of Hibernate Version 3.2.5 as persistence manager, and the local database system is an IBM DB2[4] Version 9.7.

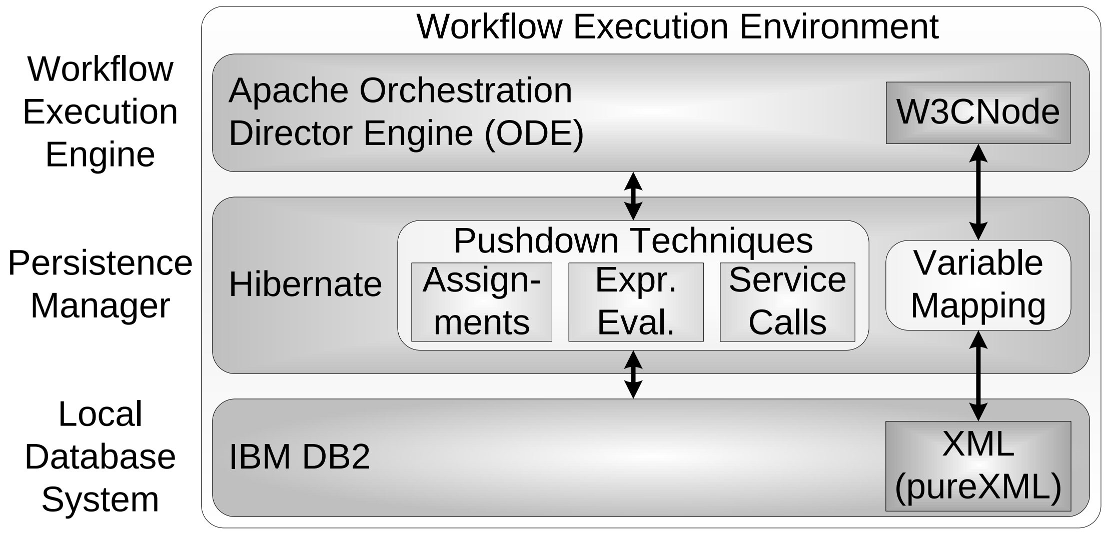
The first prototype shown in Figure 7.5a lets the workflow execution engine Apache ODE carry out local data processing tasks completely on its own. Apache ODE represents contents of workflow variables as XML documents. These documents are managed as Java objects of the type W3C Node[5]. When storing the data into the IBM DB2, Hibernate maps small XML documents to variable character strings (VarChar) and larger ones to binary large objects (BLOBs). These data structures are tailored to the original usage pattern of the local database system, i. e., it only serves as persistent data store for variable contents. In the following, this prototype is called *original ODE*.

The extended prototype depicted in Figure 7.5b abstains from the data processing optimizer shown in Figure 7.3. Instead, an extension of Hibernate implements and enforces the individual pushdown techniques summarized in Figure 7.4, which allows for evaluating the effectiveness of these techniques. So, the local IBM DB2 database system carries out the individual kinds of local data processing tasks listed in Section 7.1.1. In order that the database system is able to evaluate queries or expressions on the contents of workflow variables, the prototype changes the database-internal data structure to a native XML data type. This also makes it possible to employ the powerful XML-enabled query processing capabilities of the IBM DB2 pure XML technology [NKC09]. Altogether, this leads to a tight integration of the local database system and the workflow execution engine. This prototype is thus called *ODE-TI* (Tight Integration).

[4]IBM DB2: `https://www.ibm.com/analytics/us/en/technology/db2/`
[5]W3C Node: `http://www.w3.org/2003/01/dom2-javadoc/org/w3c/dom/Node.html`



(a) Original Apache ODE reflecting the system architecture shown in Figure 7.2.



(b) ODE-TI (Tight Integration) reflecting the system architecture shown in Figure 7.3.

Figure 7.5: Prototypes used for the experiments; cf. [RSM11].

During the experiments, all system components ran on a 32 bit Windows Server 2003 system with two Intel Xeon 3.2 GHz CPUs and 8 GB RAM. Thereby, the main focus was to investigate whether break-even points exist when ODE-TI is more efficient and robust than original ODE. Criteria for break-even points are the data size, the complexity of involved expressions, and the complexity of workflows [Wag11]. Such break-even points then provide indicators when the data processing optimizer shown in Figure 7.3 should either employ the execution engine or the database system for local data processing. Note that concrete break-even points in real scenarios are however system- and tool-dependent.

The experiments have been subdivided into two test scenarios. The first scenario comprises small test workflows, where only single workflow activities have been tested. This allows for evaluating each technique depicted in Figure 7.4 in isolation. A BPEL `assign` activity is used to evaluate the Assignment Pushdown, an `if` activity for the Expression Evaluation Pushdown, and an `invoke` activity for the Service Call Pushdown. The second test scenario makes use of the data-intensive protein modeling workflow depicted in Figure 7.1. This workflow contains all above-mentioned activities and thus all kinds of local data processing tasks listed in Section 7.1.1 [Wag11]. Furthermore, it embeds most of the activities in a loop realizing a sophisticated data iteration. This allows for evaluating all techniques shown in Figure 7.4 in combination.

The underlying data set of both test scenarios is the XML-based list of protein sequences of the protein modeling workflow [Wag11]. The tests have been carried out for five different data sizes: 100 KB, 500 KB, 4 MB, 9 MB, and 50 MB. This corresponds to 40, 199, 697, 1394, and 7695 protein sequences, respectively, as well as to the same number of iterations of the loop in the protein modeling workflow. Larger XML documents could not be tested, since original ODE generally caused main memory overloads for such data sets. Remark that this is not a severe limitation of the experimental setting. In fact, processing larger amounts of data should be outsourced to external resources or services anyway [RSM11].

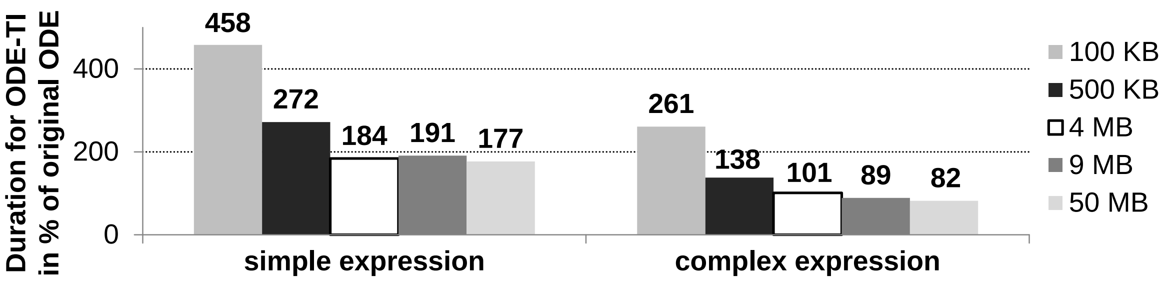
7.3.2 Discussion of Experimental Results

The following two subsections respectively discuss evaluation results regarding the two above-mentioned test scenarios reflecting (1) smaller test workflows and (2) the data-intensive protein modeling workflow.

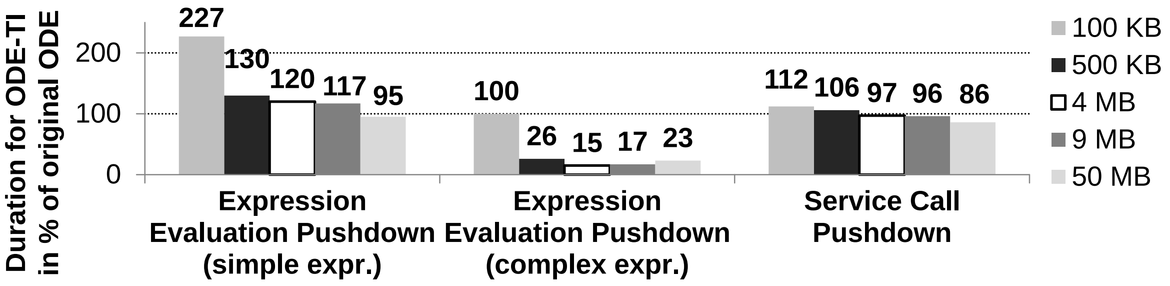
7.3.2.1 Smaller Test Workflows

Each test workflow for each technique depicted in Figure 7.4 has been executed 100 times to determine credible average durations. In Figure 7.6, the respective average durations of original ODE are taken as 100 %, i. e., the figure presents the average durations of ODE-TI as percentage of those of original ODE.

The `assign` activity used for the *Assignment Pushdown* firstly carries out a read operation to evaluate an XPath expression on the list of protein sequences. It then performs a write operation storing the expression result to another variable.



(a) Assignment Pushdown.



(b) Expression Evaluation Pushdown and Service Call Pushdown.

Figure 7.6: Effectiveness of the techniques shown in Figure 7.4; cf. [RSM11].

This pushdown technique has been evaluated with two different complexity classes of the involved XPath expression (see Figure 7.6a). Firstly, a *simple expression* selects only one protein sequence from the input list. Thereby, the Assignment Pushdown used in ODE-TI shows an average performance degradation that ranges between 358 % and 77 % for the individual data sizes. The more *complex expression* selects two protein sequences and concatenates them. Here, ODE-TI shows a performance degradation of 161 % and 38 % for 100 KB and 500 KB. Nevertheless, a break-even point where ODE-TI performs up to 18 % better than original ODE is reached when increasing the data size to more than 4 MB.

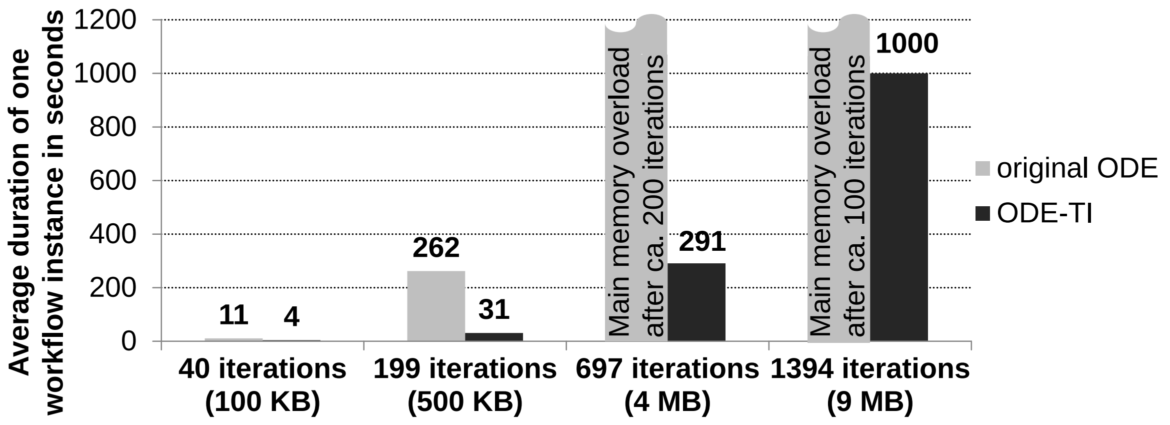
The *Expression Evaluation Pushdown* has been tested with the same two complexity classes of involved XPath expressions, as indicated in Figure 7.6b. The underlying `if` activity only performs a read operation evaluating the expression, but no subsequent write operation. For the *simple expression*, ODE-TI shows performance degradations ranging from 127 % to 17 % when reading data up to 9 MB. However, the break-even point with slightly improved performance is reached for 50 MB. The experiments for the *complex expression* reveal the potential of ODE-TI and of the approach proposed in this chapter. Here, the break-even point is already reached for the small data size of 100 KB. The bigger data sizes even lead to performance improvements between 74 % and 85 %.

The results regarding the Assignment Pushdown and Expression Evaluation Pushdown already confirm most of the arguments discussed in Section 7.2.3. In particular, the prototype ODE-TI and thus the techniques depicted in Figure 7.4 only take effect with increased data sizes and especially with complex XPath expressions. This is mainly because the local IBM DB2 database system and its efficient query processing capabilities are especially tailored to large data sets and to complex data management operations. Another observation is that the results of the Expression Evaluation Pushdown are much better than those of the Assignment Pushdown. The major difference between the underlying `if` and `assign` activities is the additional write operation of the `assign`. So, the workflow engine in original ODE appears to deal more efficiently with write operations than the local database system in ODE-TI. The reason is not only that the workflow engine may rely on tailored internal data structures, as discussed in Section 7.2.3. In addition, the database system causes an extra overhead during write operations to store log information on disk and to adapt index structures.

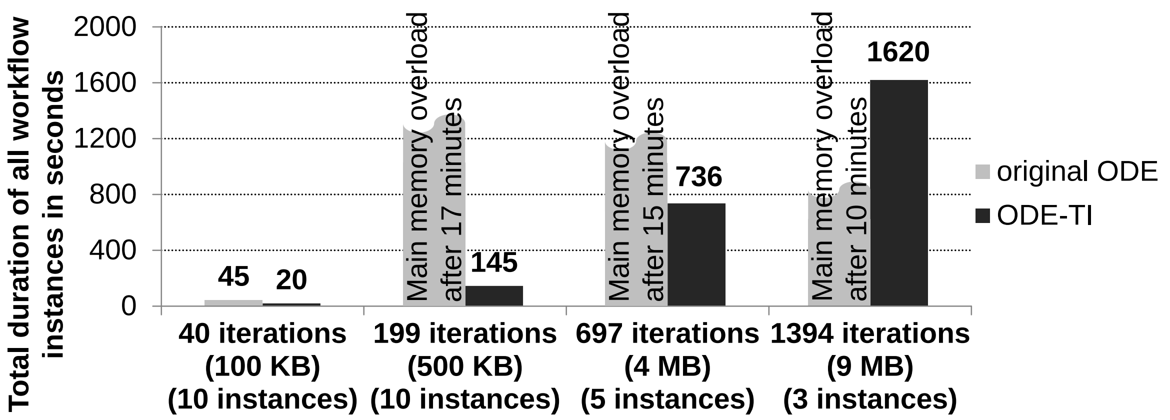
As shown in Figure 7.6b, the results regarding the *Service Call Pushdown* get likewise better with increasing data sizes. In original ODE, the underlying `invoke` activity anyway persists the result data of the service in the local database system for recovery purposes. This entails extra costs for transferring the data from the workflow execution engine over the persistence manager to the local database system. The Service Call Pushdown used in ODE-TI circumvents these extra transfer costs, as discussed in Section 7.2.2. However, ODE-TI causes a small write overhead, as it employs an XML data type instead of the more efficient BLOB type used in original ODE [Wag11] (see Figure 7.5). For smaller data sizes, this additional write overhead of ODE-TI is larger than the extra transfer costs caused by original ODE. Nevertheless, this turns into the opposite for larger data sets, which finally leads to performances improvements of up to 14 %.

7.3.2.2 Data-Intensive Protein Modeling Workflow

The second test scenario considers both the sequential and the parallel execution of several instances of the protein modeling workflow. In the case of a sequential execution, the workflow has again been carried out 100 times. Figure 7.7a presents the average durations of one workflow instance of both original ODE and ODE-TI for the data sizes of 100 KB to 9 MB. These results show the great potential of the proposed optimization approach. For a data size of 100 KB,



(a) Sequential workflow execution.



(b) Parallel workflow execution.

Figure 7.7: Performance of the protein modeling workflow; cf. [RSM11].

ODE-TI already reduces workflow duration by nearly a factor of 3 – and even by a factor of more than 8 when using 500 KB. For 4 and 9 MB, original ODE was not able to execute the entire workflow at least once. Due to main memory overloads in the workflow engine, it crashed after respectively 200 or 100 iterations of the loop. Note that original ODE was likewise not able to carry out the workflow for the larger data set of 50 MB. So, corresponding results are neglected in Figure 7.7. However, ODE-TI successfully executed all instances of the workflow for all data sizes. Altogether, these results again confirm most of the arguments discussed in Section 7.2.3. In particular, the database system is obviously able to deal with large data sets as well as with complex and repeating operations in a more efficient and robust manner than the workflow engine.

Figure 7.7b presents the results for parallel workflow execution. Here, the workflow has been executed ten times for 100 KB and 500 KB, five times for 4 MB, and three times for 9 MB. For each data size, all parallel workflow instances have been started at the same time. The figure respectively shows the total durations until the last instance has been finished. ODE-TI outperforms original ODE in any of these cases. It reduces the total duration by a factor of more than 2 already for 100 KB. When the data size grows, original ODE was not able to execute at least one of the workflow instances. After ten to 17 minutes, it aborts with a main memory overload again. ODE-TI in turn could execute all parallel instances for all data sizes within acceptable time periods. This again confirms that the techniques depicted in Figure 7.4 lead to significantly improved performance and robustness for data-intensive workflows.

7.4 Summary and Future Work

The large set of currently available optimization approaches mainly focuses on accelerating operations that process data externally to a workflow. However, approaches to optimize local data processing tasks in workflows are rarely proposed by related work. This is especially true for workflows that are described via control-flow-oriented workflow languages. This chapter therefore proposes a novel optimization approach that is targeted at this neglected setting. This approach introduces various techniques to partition relevant local data processing tasks between the components of a control-flow-based workflow system in a smart way. The techniques encompass the execution of variable assignments, expression evaluations for control flow decisions, and service calls. Previous workflow systems could only carry out such tasks within the workflow execution engine. The introduced techniques additionally allow for pushing them down to an integrated local database system. A prototypical implementation has been the basis to evaluate the effectiveness of the techniques. The test results especially demonstrate that the proposed approach significantly improves performance and robustness of the local data processing in data-intensive workflows.

Future work should mainly aim at even extending the efficiency and scalability of the ODE-TI prototype. For instance, additional index structures in the local IBM DB2 database system that are tailored to the XML data format may even increase the performance of read operations [NKC09]. The efficiency of local data processing in workflows may also be enhanced by employing main-memory-based database systems. All this may even entail that the ODE-TI prototype is already applicable to smaller data sizes and to less complex operations or workflows.

Chapter 8

Conclusion and Future Work

Scientific workflows traditionally focus on the implementation of sophisticated pipelines for data analyses [TDG07, KWK$^+$09]. Nowadays, workflow technology is increasingly adopted for computer-based simulations as well [GSK$^+$11]. Thereby, corresponding simulation workflows mainly orchestrate the interaction with usually proprietary simulation tools. These simulation tools perform long-running numerical calculations that realize mathematical simulation models, e. g., based on differential equations [Hum90, Har96]. The input data for the simulation tools often come from diverse data sources that manage their data in a multiplicity of proprietary formats. For that purpose, simulation workflows have to carry out many complex data provisioning tasks that filter and transform these heterogeneous input data in a way so that the used simulation tools can properly ingest them [RLSR$^+$06, RS14]. This results in a high effort to be spent on designing such complex data provisioning tasks [RLSR$^+$06, CBL11].

This effort to be spent on workflow design is often even increased due to one prevalent trend in simulation research. To make simulations more realistic, scientists couple mathematical simulation models from several scientific domains in simulation workflows [KSR$^+$11, GZC14]. This allows them to cover various levels of granularity in simulation calculations, which increases the potential to produce precise results [Gat14]. In such coupled simulations, the result data of one simulation model are used as input for other simulation models. Different simulation models are often implemented by different simulation tools, where each tool may rely on proprietary ways to manage its data [RSM14b]. This even increases the heterogeneity of data resources and data formats. Furthermore, the various levels of granularity involved in coupled simulation calculations make it necessary to interpolate or extrapolate data accordingly [Gat14]. All this makes the implementation of data transformations in simulation workflows even more sophisticated. These issues are highlighted by the background information introduced in Chapter 2 – and especially by the discussion of the application area of data management in simulation workflows given in Chapter 3.

Scientists typically define and execute their simulation workflows on their own. In doing so, they have to spend a considerably high effort to specify or implement the mentioned data transformations that realize the data provisioning for and the data exchange between simulation tools [RLSR$^+$06, CBL11]. This brings in additional complexity for scientists, who typically do not have adequate skills in workflow design or data management. It hinders them to concentrate on their core issues, namely the development of mathematical simulation models, the execution of simulation calculations, and the interpretation of their results.

The main contribution of this thesis is a set of novel concepts and methods that remove the burden from scientists to implement such complex data transformations. Individual concepts and methods are introduced in Chapters 4 to 7. Thereby, these chapters not only illustrate the respective approaches and their design considerations. In addition, each chapter covers comprehensive discussions of related work, and especially profound evaluations of the proposed concepts and methods. These evaluations have been backed up by elaborate prototypical implementations and by their application to several real-world simulations, e. g., to the bone simulation introduced in Section 1.1. In the following, Section 8.1 summarizes concrete contributions offered by the proposed concepts and methods. Afterwards, Section 8.2 discusses possible future work.

8.1 Summary of the Contributions

Chapter 4 deals with the issue that most existing workflow systems or other simulation tools offer a multiplicity of diverse data provisioning techniques. Examples are application-specific services or customized workflow extension activities [LAB$^+$06, VSRM08, GSK$^+$11]. Scientists are often overwhelmed with the diversity and complexity of available data provisioning techniques, which induces them not to leverage the techniques at all [CBL11]. This thesis conquers this diversity and complexity by a systematic classification of available techniques into representative data provisioning concepts. The resulting concepts are (1) data services encapsulating access to data resources, (2) data management activities that allow for a seamless access to resources, and (3) local data processing within workflows. Chapter 4 then discusses results of evaluating these concepts with respect to data management patterns that typically occur in simulation workflows and also by considering relevant non-functional issues. The evaluation results are furthermore used to derive and assess a set of guidelines that assist scientists in choosing data provisioning techniques for their workflows. Another outcome of all these discussions is a list of essential missing features existing workflow systems do not support (see Table 4.3). Hence, the last contribution introduced in Chapter 4 is an extended simulation workflow system that offers all missing features in a holistic way. Most extensions covered by this simulation workflow system correspond to the contributions introduced in succeeding chapters.

One missing feature of existing workflow systems is that they lack a generic solution to data provisioning in simulation workflows. More precisely, they are often restricted to certain data resources, data formats, and/or data management operations. Among related work, only ETL technology is sufficiently generic to support all kinds of resources, formats, and operations that are commonly required by computer-based simulations, as summarized in Tables 2.1 and 3.1 [KRRT98]. However, corresponding ETL tools have a decisive drawback that prevents their adoption for simulations. They offer a diversity of complex ETL operators, which may overwhelm scientists when designing data provisioning pipelines.

Chapter 5 introduces and assesses a set of extensions to conventional workflow languages that transfer the general ideas of ETL technology into an ETL workflow approach [MMLW05, RRS+11]. These extensions offer the necessary generic solution, but they do not have the decisive drawback of ETL technology. In fact, they correspond to only four reasonable types of generic data management activities. So, scientists are not overwhelmed with a vast amount of diverse workflow building blocks. Nevertheless, the proposed data management activities allow for specifying a broad range of data management operations for any kind of data resource or data format. Thereby, they support any operation that may be specified via the command languages offered by involved data resources, e. g., SQL statements or shell commands. A prototypical implementation and its evaluation have confirmed that the activities are sufficient to design the data provisioning for virtually all simulation examples of various scientific domains.

Using the proposed data management activities, scientists still have to specify many complex data management operations via low-level command languages. In some scenarios, the high complexity of these operations even hinders scientists to specify them at all. This is for instance the case for the sophisticated data exchange between the bio-mechanical and systems-biological calculations of the bone simulation, as discussed in Section 3.2.2. It is a common issue for multi-scale simulations that are coupled across different scientific domains, since such simulations require many low-level operations, e. g., to filter, aggregate, interpolate, or extrapolate data [Gat14, RSM14b].

Chapter 6 proposes a novel pattern-based approach that significantly enhances the abstraction support for simulation workflow design [RS13b, RS14, RSM14a, RSM14b]. It thereby completely removes the burden from scientists to specify any low-level operations in their workflows. In contrast to related work from several research areas, this pattern-based approach provides scientists with all essential benefits listed in Section 1.2.2. Scientists only need to select a few number of abstract patterns to describe the high-level simulation process they have in mind. These abstract patterns are especially meaningful to scientists, as they represent use cases they are interested in, e. g., coupling simulation models. On this note, the patterns allow for their domain-specific and thus easy parameterization. In fact, all pattern parameters correspond to terms or concepts scientists already know from their simulation models or simulation methods. A rule-based transformation approach makes it possible to map such high-level process models and patterns onto executable simulation workflows. As another major feature, a pattern hierarchy with clearly distinguished abstraction levels of patterns facilitates a holistic separation of concerns. It is a key enabler to incorporate different kinds of persons and their specific skills into workflow design, e. g., not only scientists, but also data engineers. Altogether, this conquers the data complexity associated with simulation workflows, which allows scientists to concentrate on their core issue again, namely on the simulation itself.

Finally, *Chapter 7* deals with optimization approaches to increase the efficiency of data-intensive workflows [RSM11]. Most of related work focuses on methods to accelerate operations that process data externally to a workflow. Furthermore, existing approaches are often tailored to workflows that are described via dataflow-oriented languages. However, the question how to optimize local data processing tasks within workflows that are governed by control-flow-oriented languages has largely been neglected in previous work [Mül10, Wag11]. This thesis introduces a complementary optimization approach that is targeted at this neglected setting. This approach introduces various techniques that partition relevant local data processing tasks between the components of a workflow system in a smart way.

Thereby, such tasks are either assigned to the workflow execution engine or to a tightly integrated local database system. A set of experiments revealed the full potential of this novel approach. In particular, the experimental results demonstrate that the introduced techniques significantly improve performance and robustness of the local data processing in data-intensive workflows.

Altogether, the concepts and methods proposed by this thesis fill several gaps in current research regarding data provisioning in simulation workflows. In particular, they provide a holistic abstraction for the complex design of such workflows and this way make simulation workflow design especially tailor-made for scientists. This has also been confirmed by the results of the profound evaluations of individual proposed concepts and methods.

8.2 Future Work

The presented thesis and its contributions summarized in Section 8.1 constitute a good basis for future research opportunities. Firstly, the generic data management activities proposed in Chapter 5 facilitate various kinds of optimizations to increase the efficiency of simulation workflows. The main reason is that these activities treat descriptions of data management operations as integral parts of workflow models. This in turn enables optimizations over the whole spectrum from the workflow to the data level [MMLW05, VSRM08]. For instance, Vrhovnik et al. propose a rule-based approach to re-structure workflow models with embedded SQL statements [VSS+07]. Kalyoncu discusses how to apply this approach to simulation workflows that make use of the generic data management activities proposed in Chapter 5 [Kal15]. On this note, the performance of simulation workflows may also be increased by employing scalable and parallel data processing facilities. In particular, Gessler and Dehghanipour demonstrate that MapReduce-based systems and cloud computing technologies are good candidates to accelerate computer-based simulations [Ges14, Deh15]. Future work may even further investigate this optimization potential of the data management activities proposed in Chapter 5. This should especially be underpinned by more in-depth evaluations of the mentioned or of even other optimization approaches.

The pattern-based approach to simulation workflow design introduced in Chapter 6 likewise facilitates various ways to optimize workflows. In particular, future work may investigate how to integrate rule-based optimization decisions into rewrite rules that transform patterns into executable workflows [Kal15]. Thereby, the major goal is to find efficient workflow fragments realizing given patterns. These workflow fragments may again be based on scalable data processing facilities relying on MapReduce or on cloud computing technologies [DG08, LQ14].

The approach to optimize local data processing tasks in workflows introduced in Chapter 7 offers opportunities for additional research as well. One goal of corresponding future work may be to increase the efficiency and scalability of this approach and of the underlying ODE-TI prototype. This should also be underpinned by further experimental tests. An interesting question is whether XML-native index structures in the local database system may accelerate read operations [NKC09]. Another possibility to further increase the efficiency of local data processing tasks is to employ main-memory-based database systems.

The extended simulation workflow system proposed in Section 4.4.2 and depicted in Figure 4.7 includes a provenance framework that integrates and analyzes different provenance information sources. This provenance framework is however not yet implemented in the current prototype of the extended workflow system. Hence, this implementation and the evaluation of the resulting prototype is the next obvious opportunity for future research. Thereby, one focus of this future evaluation is how the provenance framework may enhance the reproducibility of simulations and of their outcome [HTT09]. Another question is to what extent it may support a holistic optimization of the data processing in workflows. For instance, it is important to discuss how the above-mentioned optimization approaches may use the additional information derived by the provenance framework in order to make better optimization decisions [RM09].

Finally, but not less relevant, future work may even further enhance the already high generality of the contributions introduced in this thesis. In particular, the proposed concepts and methods should be applied to other application areas than simulations, e. g., to other workflow classes shown in Figure 2.6. According to Hirmer et al. [HRWM15], the pattern-based approach introduced in Chapter 6 is especially suited to foster and ease workflow design in other application areas. Note that enhancing the generality of this pattern-based approach may also require to develop new patterns, e. g., representing complex analytical operations for scientific workflows [TDG07, KWK$^+$09]. This may go hand in hand with extending the pattern hierarchy shown in Figure 6.5 by additional abstraction levels. Thereby, all this offers a great potential to even enhance and further exploit the accompanying separation of concerns for workflow design.

Author Publications

- Peter Reimann, Michael Reiter, Holger Schwarz, Dimka Karastoyanova, and Frank Leymann. SIMPL – A Framework for Accessing External Data in Simulation Workflows. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2011)*, pages 534–553, Kaiserslautern, Germany, 2011.
- Peter Reimann, Holger Schwarz, and Bernhard Mitschang. Design, Implementation, and Evaluation of a Tight Integration of Database and Workflow Engines. *Journal of Information and Data Management*, 2(3):353–368, 2011.
- Peter Reimann and Holger Schwarz. Datenmanagementpatterns in Simulationsworkflows. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2013)*, pages 279–293, Magdeburg, Germany, 2013.
- Peter Reimann, Holger Schwarz, and Bernhard Mitschang. 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*, Aalborg, Denmark, 2014.
- Peter Reimann and Holger Schwarz. Simulation Workflow Design Tailor-Made for Scientists. In *Proc. of the 26th International Conference on Scientific and Statistical Database Management*, Aalborg, Denmark, 2014.
- Peter Reimann, Tim Waizenegger, Matthias Wieland, and Holger Schwarz. Datenmanagement in der Cloud für den Bereich Simulationen und Wissenschaftliches Rechnen. In *Proc. of the 2nd Workshop Data Management in the Cloud (DMC 2014) in conjunction with the 44th Jahrestagung der Gesellschaft für Informatik (GI)*, Stuttgart, Germany, 2014.
- Peter Reimann, Holger Schwarz, and Bernhard Mitschang. A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design. In

R. Meersman et al., editor, *Proc. of the OnTheMove Federated Conferences and Workshops (OTM 2014), 22nd International Conference on Cooperative Information Systems (CoopIS 2014)*, volume 8841 of *LNCS*, pages 21–38, Amantea, Italy, 2014. Springer-Verlag, Berlin Heidelberg.

- Jorge Minguez, Peter Reimann, and Sema Zor. Event-driven Business Process Management in Engineer-to-Order Supply Chains. In *Proc. of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2011)*, pages 624–631, Lausanne, Switzerland, 2011.
- Stefan Silcher, Jan Königsberger, Peter Reimann, and Bernhard Mitschang. Cooperative Service Registries for the Service-based Product Lifecycle Management Architecture. In *Proc. of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD 2013)*, pages 439–446, Whistler, BC, Canada, 2013.
- Robert Krause, Frank Allgöwer, Dimka Karastoyanova, Frank Leymann, Bernd Markert, Bernhard Mitschang, Peter Reimann, Michael Reiter, Daniella Schittler, Syn Schmitt, Steffen Waldherr, and Wolfgang Ehlers. Scientific Workflows for Bone Remodelling Simulations. *Applied Mathematics and Mechanics*, 13(1), 2013.
- Pascal Hirmer, Peter Reimann, Matthias Wieland, and Bernhard Mitschang. Extended Techniques for Flexible Modeling and Execution of Data Mashups. In Markus Helfert, Andreas Holzinger, Orlando Belo, and Chiara Francalanci, editors, *Proc. of the 4th International Conference on Data Management Technologies and Applications (DATA)*, pages 111–122, Colmar, France, 2015. SciTePress.

Bibliography

[ABJF06] Ilkay Altintas, Oscar Barney, and Efrat Jaeger-Frank. Provenance Collection Support in the Kepler Scientific Workflow System. In *Proc. of the 2006 International Conference on Provenance and Annotation of Data*, IPAW'06, pages 118–132, Chicago, IL, USA, 2006. Springer-Verlag, Berlin Heidelberg, Germany.

[ABLM14] Marcelo Arenas, Pablo Barcel, Leonid Libkin, and Filip Murlak. *Foundations of Data Exchange.* Cambridge University Press, New York, NY, USA, 2014.

[Ace12] Miguel F. Acevedo. *Simulation of Ecological and Environmental Models.* CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2012.

[Age12] Christian Ageu. *Erweiterung und Evaluation eines Prototyps für eine enge Integration zwischen Datenbank- und Workflow-Engines.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2012.

[Alb96] Peter Albrecht. The Runge-Kutta Theory in a Nutshell. *SIAM Journal on Numerical Analysis*, 33(5):1712–1735, 1996.

[AMA06] Asif Akram, David Meredith, and Rob Allan. Evaluation of BPEL to Scientific Workflows. In *Proc. of the 6th International Symposium on Cluster Computing and the Grid*, CCGRID '06, pages 269–274, Singapore, Malaysia, 2006. IEEE.

[AP97] Uri M. Ascher and Linda R. Petzold. *Computer Method for Ordinary Differential Equations and Differential Algebraic Equations.* SIAM, 1997.

[Ari12] Stavros Aristidou. *Abstraktionsunterstützung für die Definition des Datenmanagements in Simulationsworkflows.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2012.

[ARR13] Raghu Anantharangachar, Srinivasan Ramani, and S. Rajagopalan. Ontology Guided Information Extraction from Unstructured Text. *International Journal of Web & Semantic Technology (IJWesT)*, 4(1):19–36, 2013.

[Bai02] Zhaojun Bai. Krylov Subspace Techniques for Reduced-Order Modeling of Large-Scale Dynamical Systems. *Applied Numerical Mathematics*, 43:9–44, 2002.

[BAJ$^+$10] Derik Barseghian, Ilkay Altintas, Matthew B. Jones, Daniel Crawl, Nathan Potter, James Gallagher, Peter Cornillon, Mark Schildhauer, Elizabeth T. Borer, Eric W. Seabloom, and Parviez R. Hosseini. Workflows and Extensions to the Kepler Scientific Workflow System to Support Environmental Sensor Data Access and Analysis. *Ecological Informatics*, 5(1):42–50, 2010.

[BBBD12] Wouter Buytaert, Selene Baez, Macarena Bustamante, and Art Dewulf. Web-Based Environmental Simulation: Bridging the Gap between Scientific Modeling and Decision-Making. *Environmental Science and Technology*, 46(4):1971–1976, 2012.

[BBF$^+$09] Stefan Bauer, Jochen Boy, Arnulf Fröhlich, Matthias Grau, Karl Gruber, Uwe Krempels, Thomas Merkt, Katja von Merten, Wolfgang Schlüter, and Stefan Sebrantke. Automotive CAE Integration – Requirements and Evaluation of Interfaces. Technical report, AUDI AG, BMW AG, Daimler AG, Dr. Ing. h. c. F. Porsche AG, Volkswagen AG, PROSTEP AG, 2009.

[BBH$^+$13] Tobias Binz, Uwe Breitenbücher, Florian Haupt, Oliver Kopp, Frank Leymann, Alexander Nowak, and Sebastian Wagner. OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In *Proc. of 11th International Conference on Service-Oriented Computing (ICSOC'13)*, volume 8274 of *LNCS*, pages 692–695, Berlin, Germany, 2013. Springer-Verlag, Berlin Heidelberg.

[BBK+14] Uwe Breitenbücher, Tobias Binz, Kálmán Képes, Oliver Kopp, Frank Leymann, and Johannes Wettinger. Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA. In *Proc. of the IEEE International Conference on Cloud Engineering (IC2E)*, pages 87–96. IEEE Computer Society, 2014.

[BDH11] Paul Blanchard, Robert L. Devaney, and Glen R. Hall. *Differential Equations*. Brooks/Cole Thomson Learning, Boston, MA, USA, 2011.

[BHW+07] Matthias Böhm, Dirk Habich, Uwe Wloka, Jüurgen Bittner, and Wolfgang Lehner. Towards Self-Optimization of Message Transformation Processes. In *Communications of the 11th East-European Conference on Advances in Databases and Information Systems (ADBIS)*, Varna, Bulgaria, 2007.

[BJA+08] Roger Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan. The Trident Scientific Workflow Workbench. In *Proc. of the 4th International Conference on e-Science*, pages 317–318, Indianapolis, IN, USA, 2008.

[BKLW99] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, and Herbert Weber. Federated Information Systems: Concepts, Terminology and Architectures. Technical report, Department of Computer Science at the Technische Universität Berlin, 1999.

[BKML+10] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. Genbank. *Nucleic Acids Research*, 38(suppl 1):D46–D51, 2010.

[BKR11] Andreas Benzing, Boris Koldehofe, and Kurt Rothermel. Efficient Support for Multi-Resolution Queries in Global Sensor Networks. In *Proc.of the 5th International Conference on Communication System Software and Middleware*, Verona, Italy, 2011. ACM.

[BL05] Shawn Bowers and Bertram Ludäscher. Actor-Oriented Design of Scientific Workflows. In *24th International Conference on Conceptual Modeling*, pages 369–384, Klagenfurt, Austria, 2005. Springer-Verlag Berlin Heidelberg.

[Boh14] Andreas Bohrn. *Pattern-basierte Definition der Datenbereitstellung für Simulationen zu Strukturänderungen in Knochen.* Student thesis, University of Stuttgart, Institute of Parallel and

Distributed Systems, 2014.

[Boh15] Andreas Bohrn. *Abbildung von Datenmanagementpatterns auf ausführbare Workflowfragmente.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2015.

[BS00] Ed F. Begley and Charles P. Sturrock. MatML: An XML for Standardizing Web-based Materials Property Data. *Journal of the Minerals, Metals and Materials Society*, 52(7):56–56, 2000.

[BTS07] Jeremy M. Berg, John L. Tymoczko, and Lubert Stryer. *Biochemistry.* W. H. Freeman and Co., New York City, NY, USA, 2007.

[CBL11] Sarah Cohen-Boulakia and Ulf Leser. Search, Adapt, and Reuse: The Future of Scientific Workflows. *SIGMOD Record*, 40(2):6–16, 2011.

[CEB$^+$09] Nazario Cipriani, Mike Eissele, Andreas Brodt, Matthias Großmann, and Bernhard Mitschang. NexusDS: A Flexible and Extensible Middleware for Distributed Stream Processing. In *Proc. of the 2009 International Symposium on Database Engineering and Applications*, IDEAS '09, pages 152–161. ACM, 2009.

[CGH$^+$06] David Churches, Gabor Gombas, Andrew Harrison, Jason Maassen, Craig Robinson, Matthew Shields, Ian Taylor, and Ian Wang. Programming Scientific and Distributed Workflow with Triana Services. *Concurrency and Computation: Practice and Experience*, 18(10):1021–1037, 2006.

[CGP00] Oscar Corcho and Asunción Gómez-Pérez. Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages. In *Proc. of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods*, Berlin, Germany, 2000.

[CH09] David Cohn and Richard Hull. Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. *IEEE Data Engineering Bulletin*, 32(3):3–9, 2009.

[CJLP03] Zhimin Chen, H. V. Jagadish, Laks V. S. Lakshmanan, and Stelios Paparizos. From Tree Patterns to Generalized Tree Patterns: On Efficient Evaluation of XQuery. In *Proc. of the 29th*

International Conference on Very Large Data Bases, VLDB ’03, pages 237–248, Berlin, Germany, 2003. VLDB Endowment.

[CMJNG08] Daniel L. Cook, Jose L. V. Mejino Jr., Maxwell L. Neal, and John H. Gennari. Bridging Biological Ontologies and Biosimulation: The Ontology of Physics for Biology. In *Proc. of the AMIA Annual Symposium*, pages 136–140, Washington, DC, USA, 2008.

[CMPW07] Robert D. Cook, David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and Applications of Finite Element Analysis.* John Wiley & Sons, New York, NY, USA, 2007.

[Cod70] Edgar F. Codd. A Relational Model of Data for Large Shared Data Banks. *Commununications of the ACM*, 13(6):377–387, 1970.

[COdO$^+$10] Fábio Coutinho, Eduardo Ogasawara, Daniel de Oliveira, Vanessa Braganholo, Alexandre Lima, Alberto Dávila, and Marta Mattoso. Data Parallelism in Bioinformatics Workflows using Hydra. In *Proc. of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC 2010)*, Chicago, IL, USA, 2010.

[COJ$^+$10] Yifeng Cui, Kim B. Olsen, Thomas H. Jordan, Kwangyoon Lee, Jun Zhou, Patrick Small, Daniel Roten, Geoffrey Ely, Dhabaleswar K. Panda, Amit Chourasia, John Levesque, Steven M. Day, and Philip Maechling. Scalable Earthquake Simulation on Petascale Supercomputers. In *Proc. of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, SC ’10, pages 1–20, New Orleans, LA, USA, 2010. IEEE Computer Society.

[CVDK$^+$12] Víctor Cuevas-Vicenttín, Saumen Dey, Sven Köhler, Sean Riddle, and Bertram Ludäscher. Scientific Workflows and Provenance: Introduction and Research Opportunities. *Datenbank-Spektrum*, 12(3):193–203, 2012.

[CWGN11] Nazario Cipriani, Matthias Wieland, Matthias Grossmann, and Daniela Nicklas. Tool Support for the Design and Management of Context Models. *Information Systems*, 36(1):99–114, 2011.

[dBS73] Carl de Boor and Blair Swartz. Collocation at Gaussian Points. *SIAM Journal on Numerical Analysis*, 10(4):582–606, 1973.

[DC08] Ewa Deelman and Ann Chervenak. Data Management Challenges of Data-Intensive Scientific Workflows. In *Proc. of the 8th International Symposium on Cluster Computing and the Grid*, CCGRID '08, Lyon, France, 2008. IEEE.

[DCBS+10] Sergio M. S. Da Cruz, Vanessa Batista, Edno Silva, Frederico Tosta, Clarissa Vilela, Rafael Cuadrat, Diogo Tschoeke, Alberto M. R. Davila, Maria L. M. Campos, and Marta Mattoso. Detecting Distant Homologies on Protozoans Metabolic Pathways using Scientific Workflows. *International Journal of Data Mining and Bioinformatics (IJDMB)*, 4(3):256–280, 2010.

[DCM+12] L. Dou, G. Cao, Paul J. Morris, R. A. Morris, Bertram Ludäscher, James A. Macklin, and James Hanken. Kurator: A Kepler Package for Data Curation Workflows. In *Proc. of the International Conference on Computational Science*, ICCS 2012, pages 1614–1619, Omaha, NE, USA, 2012.

[Deh15] Marzieh Dehghanipour. *Design and Implementation of TOSCA Service Templates for Provisioning and Executing Bone Simulations in Cloud Environments.* Master thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2015.

[DF08] Susan B. Davidson and Juliana Freire. Provenance and Scientific Workflows: Challenges and Opportunities. In *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*, pages 1345–1350, Vancouver, Canada, 2008. ACM.

[DG08] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[DHW+08] Stefan Dessloch, Mauricio A. Hernández, Ryan Wisnesky, Ahmed Radwan, and Jindan Zhou. Orchid: Integrating Schema Mapping and ETL. In *Proc. of the 24th International Conference on Data Engineering*, ICDE '08, pages 1307–1316, Cancún, México, 2008. IEEE.

[DM14] Florian Daniel and Maristella Matera. *Mashups: Concepts, Models and Architectures.* Springer-Verlag, Berlin Heidelberg, Germany, 2014.

[DMHBT07] Tien-Tuan Dao, Frédéric Marin, and Marie Christine Ho Ba Tho. Ontology of the Musculo-Skeletal System of the Lower Limbs. In *Proc. of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 386–389, Lyon, France, 2007. IEEE.

[dOCT$^+$09] Daniel de Oliveira, Luiz Cunha, Luiz Tomaz, Vinicius Pereira, and Marta Mattoso. Using Ontologies to Support Deep Water Oil Exploration Scientific Workflows. In *Proc. of the 2009 Congress on Services – I*, pages 364–367, Los Angeles, CA, USA, 2009. IEEE.

[dOOD$^+$12] Daniel de Oliveira, Eduardo Ogasawara, Jonas Dias, Fernanda Baiao, and Marta Mattoso. Ontology-based Semi-automatic Workflow Composition. *Journal of Information and Data Management*, 3(1):61–72, 2012.

[Dor11] Raymond Dormien. *Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen*. Diploma thesis, University of Stuttgart, Institute of Architecture of Application Systems, 2011.

[DP10] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. John Wiley and Sons, Chichester, UK, 2010.

[DSS$^+$05] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed Systems. *Scientific Programming*, 13(3):219–237, 2005.

[Ehl09] Wolfgang Ehlers. Challenges of Porous Media Models in Geo- and Biomechanical Engineering including Electro-Chemically Active Polymers and Gels. *International Journal of Advances in Engineering Sciences and Applied Mathematics*, 1(1):1–24, 2009.

[EKM11] Wolfgang Ehlers, Robert Krause, and Bernd Markert. Modelling and Remodelling of Biological Tissue in the Framework of Continuum Biomechanics. *Applied Mathematics and Mechanics*, 11(1):35–38, 2011.

[EL96] Johann Eder and Walter Liebhart. Workflow Recovery. In *Proc. of the 1st International Conference on Cooperative Information Systems (CoopIS 1996)*, pages 124–134, Brussels, Belgium, 1996. IEEE.

[FDP15] Fabian Franzelin, Patrick Diehl, and Dirk Pflüger. Non-intrusive Uncertainty Quantification with Sparse Grids for Multivariate Peridynamic Simulations. In Michael Griebel and Marc Alexander Schweitzer, editors, *Meshfree Methods for Partial Differential Equations VII*, volume 100 of *Lecture Notes in Computational Science and Engineering*, pages 115–143. Springer International Publishing, 2015.

[FE11] Jörg Fehr and Peter Eberhard. Simulation Process of Flexible Multibody Systems with Non-modal Model Order Reduction Techniques. *Multibody System Dynamics*, 25(3):313–334, 2011.

[FHS09] Christian Fritz, Richard Hull, and Jianwen Su. Automatic Construction of Simple Artifact-based Business Processes. In *Proc. of the 12th International Conference on Database Theory*, ICDT '09, pages 225–238, St. Petersburg, Russia, 2009. ACM.

[FKSS08] Juliana Freire, David Koop, Emanuele Santos, and Cláudio T. Silva. Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering*, 10(3):11–21, 2008.

[FSC+06] Juliana Freire, Cláudio T. Silva, Steven P. Callahan, Emanuele Santos, Carlos E. Scheidegger, and Huy T. Vo. Managing Rapidly-Evolving Scientific Workflows. In *Proc. of the 2006 International Conference on Provenance and Annotation of Data*, IPAW'06, pages 10–18, Chicago, IL, USA, 2006. Springer-Verlag.

[Gat14] Bernhard Gatzhammer. *Efficient and Flexible Partitioned Simulation of Fluid-Structure Interactions*. PhD thesis, Technische Universität München, Chair of Scientific Computing at the Department of Informatics, 2014.

[GCMS12] Katarina Grolinger, Miriam A. M. Capretz, José R. Marti, and Krishan Srivastava. Ontology–based Representation of Simulation Models. In *Proc. of the 24th International Conference on Software Engineering and Knowledge Engineering*, SEKE 2012, pages 432–437, Redwood City, CA, USA, 2012.

[GDE+07] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the Challenges of Scientific Workflows. *IEEE Computer*, 40(12):24–32, 2007.

[Ges14] Alexander Gessler. *MapReduce to Couple a Bio-mechanical and a Systems-biological Simulation*. Bachelor thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2014.

[GGCFEl07] Fernando García-García, Gerardo Cisneros, Agustín Fernández-Eguiarte, and Román Álvarez. *Numerical Simulations in the Environmental and Earth Sciences*. Cambridge University Press, Cambridge, UK, 2007.

[GHCM09] Thilina Gunarathne, Chathura Herath, Eran Chinthaka, and Suresh Marru. Experience with Adapting a WS-BPEL Runtime for eScience Workflows. In *Proc. of the 5th Grid Computing Environments Workshop*, Portland, OR, USA, 2009.

[GRS07] Christian Grossmann, Hans-Görg Roos, and Martin Stynes. *Numerical Treatment of Partial Differential Equations*. Universitext. Springer-Verlag, Berlin, Heidelberg, New York, 2007.

[GSAH+15] Santiago Gómez Sáez, Vasilios Andrikopoulos, Michael Hahn, Dimka Karastoyanova, and Andreas Weiß. Enabling Reusable and Adaptive Modeling, Provisioning & Execution of BPEL Processes. In *Proc. of the 8th International Conference on Service-Oriented Computing and Applications*, SOCA'15, Rome, Italy, 2015. IEEE.

[GSK+11] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. Conventional Workflow Technology for Scientific Simulation. In Xiaoyu Yang and Lizhe Wang, editors, *Guide to e-Science*. Springer-Verlag, London, UK, 2011.

[GT04] Torsten Grust and Jens Teubner. Relational Algebra: Mother Tongue – XQuery: Fluent. In Vojkan Mihajlovic and Djoerd Hiemstra, editors, *Proc. of the 1st Twente Data Management Workshop*, CTIT Workshop Proceedings Series, pages 9–16, Enschede, The Netherlands, 2004. Centre for Telematics and Information Technology (CTIT), University of Twente.

[GZC14] Derek Groen, Stefan J. Zasada, and Peter V. Coveney. Survey of Multiscale and Multiphysics Applications and Communities. *Computing in Science and Engineering*, 16(2):34–43, 2014.

[Har96] Stephan Hartmann. The World as a Process: Simulations in the Natural and Social Sciences. In Rainer Hegselmann, Ulrich Mueller, and Klaus Troitzsch, editors, *Simulation and Modelling in the Social Sciences from the Philosophy of Science Point of View, Theory and Decision Library*, pages 77–100. Kluwer: Dordrecht, 1996.

[Hau81] Eberhard Haug. Engineering Safety Analysis via Destructive Numerical Experiments. *Engineering Transactions*, 29(1):39–49, 1981.

[HG05] Gerd Heber and Jim Gray. Supporting Finite Element Analysis with a Relational Database Backend Part I: There is Life beyond Files. Technical Report MSR-TR-2005-49, Microsoft Research, April 2005.

[HG06] Gerd Heber and Jim Gray. Supporting Finite Element Analysis with a Relational Database Backend Part II: Database Design and Access. Technical Report MSR-TR-2006-21, Microsoft Research, February 2006.

[HI08] Muneo Hori and Tsuyoshi Ichimura. Current State of Integrated Earthquake Simulation for Earthquake Hazard and Disaster. *Journal of Seismology*, 12(2):307–321, 2008.

[HKR13] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in Cloud Computing: What it is, and What it is Not. In *Proc. of the 10th International Conference on Autonomic Computing*, (ICAC 2013), pages 24–28, San Jose, CA, USA, 2013.

[HPD$^+$05] Gerd Heber, Chris Pelkie, Andrew Dolgert, Jim Gray, and David Thompson. Supporting Finite Element Analysis with a Relational Database Backend; Part III: OpenDX — Where the Numbers Come Alive. Technical Report MSR-TR-2005-151, Microsoft Research, November 2005.

[HRWM15] Pascal Hirmer, Peter Reimann, Matthias Wieland, and Bernhard Mitschang. Extended Techniques for Flexible Modeling and Execution of Data Mashups. In Markus Helfert, Andreas Holzinger, Orlando Belo, and Chiara Francalanci, editors, *Proc. of the 4th International Conference on Data Management Technologies and Applications (DATA)*, pages 111–122, Colmar, France, 2015. SciTePress.

[HSR$^+$10] Wolfgang Hüttig, Michael Schneidt, René Rehn, Michael Hahn, Firas Zoabi, Daniel Brüderle, and Xi Tu. *Studienprojekt SIMPL – Spezifikation.* Internal report of the study project SIMPL, University of Stuttgart, Institute of Architecture of Application Systems and Institute of Parallel and Distributed Systems, 2010.

[HTT09] Tony Hey, Stewart Tansley, and Kristin Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, Redmond, WA, USA, 2009.

[Hul08] Richard Hull. Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In *Proc. of the On-TheMove Federated Conferences and Workshops (OTM 2008), 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008)*, LNCS, pages 1152–1163, Monterrey, Mexico, 2008. Springer-Verlag, Berlin Heidelberg.

[Hum90] Paul Humphreys. Computer Simulations. In *Proc. of the Biennial Meeting of the Philosophy of Science Association (PSA)*, pages 497–506, 1990.

[Hun12] William Hunter. *Recent Advances and Issues in Environmental Science.* Apple Academic Press Inc., Toronto, New York, 2012.

[IH88] Ronald L. Iman and Jon C. Helton. An Investigation of Uncertainty and Sensitivity Analysis Techniques for Computer Models. *Risk Analysis*, 8(1):71–90, 1988.

[ISO11a] ISO: International Organization for Standardization. *SQL Multimedia and Application Packages – Part 3: Spatial, ISO/IEC 13249-3:2011*, 2011.

[ISO11b] ISO: International Organization for Standardization. *Structured Query Language (SQL) – Part 1: Framework (SQL/Framework), ISO/IEC 9075-1:2011*, 2011.

[ISO11c] ISO: International Organization for Standardization. *Structured Query Language (SQL) – Part 14: XML-Related Specifications*

(SQL/XML), ISO/IEC 9075-14:2011, 2011.

[ISO11d] ISO: International Organization for Standardization. *Structured Query Language (SQL) – Part 2: Foundation (SQL/Foundation), ISO/IEC 9075-2:2011*, 2011.

[ISO11e] ISO: International Organization for Standardization. *Structured Query Language (SQL) – Part 4: Persistent Stored Modules (SQL/PSM)), ISO/IEC 9075-4:2011*, 2011.

[JHM04] Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. Advances in Dataflow Programming Languages. *ACM Computing Surveys*, 36(1):1–34, 2004.

[JMG11] Philipp Janowski, Bernhard Mitschang, and Andreas Gollmann. Issues and Characteristics of Testing as Part of the Design Process in Mechanical Engineering. In *Proc. of the 15th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Lausanne, Switzerland, 2011.

[JPT14] Michael Johnson, Jorge Pérez, and James F. Terwilliger. What Can Programming Languages Say About Data Exchange? In *Proc. of the 17th International Conference on Extending Database Technology*, EDBT 2014, pages 223–228, Athens, Greece, 2014.

[KAK$^+$13] Robert Krause, Frank Allgöwer, Dimka Karastoyanova, Frank Leymann, Bernd Markert, Bernhard Mitschang, Peter Reimann, Michael Reiter, Daniella Schittler, Syn Schmitt, Steffen Waldherr, and Wolfgang Ehlers. Scientific Workflows for Bone Remodelling Simulations. *Applied Mathematics and Mechanics*, 13(1), 2013.

[Kal15] Savas Kalyoncu. *Optimierung der Datenverarbeitung in Simulationsworkflows.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2015.

[KFPI11] Vangelis Karkaletsis, Pavlina Fragkou, Georgios Petasis, and Elias Iosif. Ontology Based Information Extraction from Text. In Georgios Paliouras, Constantine D. Spyropoulos, and George Tsatsaronis, editors, *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution: Bridging the Semantic Gap*, pages 89–109. Springer-Verlag, Berlin, Heidelberg, Germany, 2011.

[KGK+11] Oliver Kopp, Katharina Görlach, Dimka Karastoyanova, Frank Leymann, Michael Reiter, David Schumm, Mirko Sonntag, Steve Strauch, Tobias Unger, Matthias Wieland, and Rania Khalaf. A Classification of BPEL Extensions. *Journal of Systems Integration*, 2(4):2–28, 2011.

[KKL+05] Matthias Kloppmann, Dieter König, Frank Leymann, Gerhard Pfau, Alan Rickayzen, Claus von Riegen, Patrick Schmidt, and Ivana Trickovic. *WS-BPEL Extension for Sub-processes – BPEL-SPE*, 2005.

[KKL07] Rania Khalaf, Dimka Karastoyanova, and Frank Leymann. Pluggable Framework for Enabling the Execution of Extended BPEL Behavior. In *Proc. of the 3rd International Workshop on Engineering Service-Oriented Applications: Analysis, Design and Composition*, WESOA'2007, Vienna, Austria, 2007.

[KKL08a] Rania Khalaf, Oliver Kopp, and Frank Leymann. Maintaining Data Dependencies Across BPEL Process Fragments. *International Journal of Cooperative Information Systems (IJCIS)*, 17(3):259–282, 2008.

[KKL08b] Oliver Kopp, Rania Khalaf, and Frank Leymann. Deriving Explicit Data Links in WS-BPEL Processes. In *Proc. of the International Conference on Services Computing*, SCC 2008, pages 367–376, Honolulu, Hawaii, USA, 2008. IEEE Computer Society Press.

[KME10] Robert Krause, Bernd Markert, and Wolfgang Ehlers. A Porous Media Model for the Description of Adaptive Bone Remodelling. *Applied Mathematics and Mechanics*, 10(1):79–80, 2010.

[Kol05] Phokion G. Kolaitis. Schema Mappings, Data Exchange, and Metadata Management. In *Proc. of the 24th SIGMOD Symposium on Principles of Database Systems*, PODS '05, pages 61–75, Baltimore, Maryland, USA, 2005. ACM.

[KR11] Vera Künzle and Manfred Reichert. PHILharmonicFlows: Towards a Framework for Object-aware Process Management. *Journal of Software Maintenance and Evolution: Research and Practice*, 23(4):205–244, 2011.

[Kra14] Robert Krause. *Growth, Modelling and Remodelling of Biological Tissue*. PhD thesis, University of Stuttgart, Institute of Applied Mechanics (Continuum Mechanics), 2014.

[KRRT98] Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing and Deploying Data Warehouses*. John Wiley & Sons, Inc., New York City, NY, USA, 1st edition, 1998.

[KSB$^+$10] David Koop, Emanuele Santos, Bela Bauer, Matthias Troyer, Juliana Freire, and Cláudio T. Silva. Bridging Workflow and Data Provenance Using Strong Links. In *Proc. of the 22nd International Conference on Scientific and Statistical Database Management*, SSDBM'10, pages 397–415, Heidelberg, Germany, 2010. Springer-Verlag.

[KSR$^+$11] Robert Krause, Daniella Schittler, Michael Reiter, Steffen Waldherr, Frank Allgöwer, Dimka Karastoyanova, Frank Leymann, Bernd Markert, and Wolfgang Ehlers. Bone Remodelling: A Combined Biomechanical and Systems-Biological Challenge. *Applied Mathematics and Mechanics*, 11(1):99–100, 2011.

[KSW$^+$12] Robert Krause, Daniella Schittler, Steffen Waldherr, Frank Allgöwer, Bernd Markert, and Wolfgang Ehlers. Remodelling Processes in Bones: A Biphasic Porous Media Model. *Applied Mathematics and Mechanics*, 12(1):131–132, 2012.

[KTJT03] Dimitri Komatitsch, Seiji Tsuboi, Chen Ji, and Jeroen Tromp. A 14.6 Billion Degrees of Freedom, 5 Teraflops, 2.5 Terabyte Earthquake Simulation on the Earth Simulator. In *Proc. of the 2003 ACM/IEEE Conference on Supercomputing*, SC '03, Phoenix, AZ, USA, 2003. ACM.

[KWK$^+$09] Chandrika Kamath, Nikil Wale, George Karypis, Gaurav Pandey, Vipin Kumar, Krishna Rajan, Nagiza F. Samatova, Paul Breimyer, Guruprasad Kora, Chongle Pan, and Srikanth Yoginath. Scientific Data Analysis. In Arie Shoshani and Doron Rotem, editors, *Scientific Data Management: Challenges, Technology, and Deployment*, Computational Science Series, chapter 8, pages 281–323. Chapman & Hall, Boca Raton, FL, USA, 2009.

[KWvL+07] Dimka Karastoyanova, Branimir Wetzstein, Tammo van Lessen, Daniel Wutke, Jörg Nitzsche, and Frank Leymann. Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. In *Proc. of the 2nd International ICDE Workshop on Service Engineering*, SEIW 2007, pages 347–354, Istanbul, Turkey, 2007.

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[LAB+09] Bertram Ludäscher, Ilkay Altintas, Shawn Bowers, Julian Cummings, Terence Critchlow, Ewa Deelman, David De Roure, Juliana Freire, Carole Goble, Matthew Jones, Scott Klasky, Timothy McPhillips, Norbert Podhorszki, Claudio Silva, Ian Taylor, and Mladen Vouk. Scientific Process Automation and Workflow Management. In Arie Shoshani and Doron Rotem, editors, *Scientific Data Management: Challenges, Technology, and Deployment*, Computational Science Series, chapter 13, pages 467–508. Chapman & Hall, Boca Raton, FL, USA, 2009.

[LAG03] Bertram Ludäscher, Ilkay Altintas, and Amarnath Gupta. Compiling Abstract Scientific Workflows into Web Service Workflows. In *Proc. of the 15th International Conference on Scientific and Statistical Database Management (SSDBM 2003)*, pages 251–254, Cambridge, MA, USA, 2003. IEEE.

[Lar01] Tobias Larsson. *Multibody Dynamic Simulation in Product Development.* PhD thesis, Lulea University of Technology, Division of Computer Aided Design, Department of Mechanical Engineering, 2001.

[Ley95] Frank Leymann. Supporting Business Transactions Via Partial Backward Recovery In Workflow Management Systems. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW 1995)*, pages 51–70, Dresden, Germany, 1995. Springer-Verlag.

[Ley03] Frank Leymann. Web Services: Distributed Applications without Limits – an Outline. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2003)*, pages 2–23, Leipzig, Germany, 2003.

[Ley05] Frank Leymann. The (Service) Bus: Services Penetrate Everyday Life. In *Proc. of the 3rd International Conference on Service Oriented Computing (ICSOC)*, volume 3826 of *LNCS*, pages 12–20, Amsterdam, The Netherlands, 2005. Springer-Verlag.

[LMK10] Olivier P. Le Maître and Omar M. Knio. *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics.* Scientific Computation. Springer-Verlag, Dordrecht, Heidelberg, London, New York City, 2010.

[LMLG04] Philip Lord, Alison Macdonald, Liz Lyon, and David Giaretta. From Data Deluge to Data Curation. Technical report, The Digital Archiving Consultancy Limited and the Digital Curation Centre, 2004.

[LMP+05] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler. TinyOS: An Operating System for Sensor Networks. In Werner Weber, Jan M. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, pages 115–148. Springer-Verlag, Berlin Heidelberg New York, 2005.

[LN07] Ulf Leser and Felix Naumann. *Informationsintegration – Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen.* dpunkt.verlag, 2007.

[LQ14] Xiaolin Li and Judy Qiu. *Cloud Computing for Data-Intensive Applications.* Springer-Verlag, New York City, NY, USA, 2014.

[LR00] Frank Leymann and Dieter Roller. *Production Workflow: Concepts and Techniques.* Prentice Hall, Englewood Cliffs, NJ, USA, 2000.

[LSS01] Laks V. S. Lakshmanan, Fereidoon Sadri, and Subbu N. Subramanian. SchemaSQL: An Extension to SQL for Multidatabase Interoperability. *ACM Transactions on Database Systems (TODS)*, 26(4):476–519, 2001.

[LTP11] Xiufeng Liu, Christian Thomsen, and Torben Bach Pedersen. ETLMR: A Highly Scalable Dimensional ETL Framework Based on MapReduce. In *Proc. of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'11)*, LNCS, pages 96–111, Toulouse, France, 2011. Springer-Verlag, Berlin, Heidelberg.

[LWMB09] Bertram Ludäscher, Mathias Weske, Timothy McPhillips, and Shawn Bowers. Scientific Workflows: Business as Usual? In *Proc. of the 7th International Conference on Business Process Management (BPM 2009)*, Ulm, Germany, 2009.

[Lyn08] Peter Lynch. The Origins of Computer Weather Prediction and Climate Modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.

[MB08] Kenton McHenry and Peter Bajcsy. An Overview of 3D Data Content, File Formats and Viewers. Technical Report isda08-002, National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign, 2008.

[MBBL15] Timothy McPhillips, Shawn Bowers, Khalid Belhajjame, and Bertram Ludäscher. Retrospective Provenance Without a Run-time Provenance Recorder. In *Proc. of the 7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)*, Edinburgh, Scotland, UK, 2015. USENIX Association.

[MBZL09] Timothy McPhillips, Shawn Bowers, Daniel Zinn, and Bertram Ludäscher. Scientific Workflow Design for Mere Mortals. *Future Generation Computer Systems*, 25(5):541–551, 2009.

[MCD$^+$05] Philip Maechling, Hans Chalupsky, Maureen Dougherty, Ewa Deelman, Yolanda Gil, Sridhar Gullapalli, Vipin Gupta, Carl Kesselman, Jihic Kim, Gaurang Mehta, Brian Mendenhall, Thomas Russ, Gurmeet Singh, Marc Spraragen, Garrick Staples, and Karan Vahi. Simplifying Construction of Complex Workflows for Non-expert Users of the Southern California Earthquake Center Community Modeling Environment. *ACM SIGMOD Record*, 34(3):24–30, 2005.

[MFHH05] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An Acquisitional Query Processing

System for Sensor Networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[MG11] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing. National Institute of Standards and Technology (NIST)*, 2011.

[MGLRtH11] Sara Migliorini, Mauro Gambini, Marcello La Rosa, and Arthur H. M. ter Hofstede. *Pattern-Based Evaluation of Scientific Workflow Management Systems.* Queensland University of Technology (QUT) Technical Report 39935, 2011.

[MHM04] Norman May, Sven Helmer, and Guido Moerkotte. Nested Queries and Quantifiers in an Ordered Context. In *Proc. of the 20th International Conference on Data Engineering*, ICDE '04, pages 239–250, Washington, DC, USA, 2004. IEEE Computer Society.

[MMLW05] Albert Maier, Bernhard Mitschang, Frank Leymann, and Dan Wolfson. On Combining Business Process Integration and ETL Technologies. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2005)*, pages 533–546, Karlsruhe, Germany, 2005.

[MMZ15] Ryan Mork, Paul Martin, and Zhiming Zhao. Contemporary Challenges for Data-intensive Scientific Workflow Management Systems. In *Proc. of the 10th Workshop on Workflows in Support of Large-Scale Science*, WORKS '15, pages 4:1–4:11, Austin, TX, USA, 2015. ACM.

[Mor11] J. Paul Morrison. *Flow-Based Programming: A New Approach to Application Development.* CreateSpace, 2011.

[MSK+15] Timothy McPhillips, Tianhong Song, Tyler Kolisnik, Steve Aulenbach, Khalid Belhajjame, Kyle Bocinsky, Yang Cao, Fernando Chirigati, Saumen C. Dey, Juliana Freire, Deborah N. Huntzinger, Christopher Jones, David Koop, Paolo Missier, Mark Schildhauer, Christopher R. Schwalm, Yaxing Wei, James Cheney, Mark Bieda, and Bertram Ludäscher. YesWorkflow: A User-Oriented, Language-Independent Tool for Recovering Workflow Information from Scripts. *International Journal of Digital Curation*, 10:298–313, 2015.

[Mül10] Christoph Marian Müller. *Development of an Integrated Database Architecture for a Runtime Environment for Simulation Workflows.* Diploma thesis, University of Stuttgart, Institute of Architecture of Application Systems, 2010.

[NC03] Anil Nigam and Nathan S. Caswell. Business Artifacts: An Approach to Operational Specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[NKC09] Matthias Nicola and Pav Kumar-Chatterjee. *DB2 pureXML Cookbook: Master the Power of the IBM Hybrid Data Server.* Pearson Education Inc., IBM Press, Boston, MA, USA, 2009.

[OAS07] OASIS: Organization for the Advancement of Structured Information Standards. *Web Services Business Process Execution Language Version 2.0*, 2007.

[OAS10] OASIS: Organization for the Advancement of Structured Information Standards. *WS-BPEL Extension for People (BPEL4People) Specification Version 1.1*, 2010.

[OAS13] OASIS: Organization for the Advancement of Structured Information Standards. *Topology and Orchestration Specification for Cloud Applications Version 1.0*, 2013.

[OdOV$^+$11] Eduardo S. Ogasawara, Daniel de Oliveira, Patrick Valduriez, Jonas Dias, Fabio Porto, and Marta Mattoso. An Algebraic Approach for Data-Centric Scientific Workflows. In *Proc. of the 37th International Conference on Very Large Data Bases (VLDB 2011)*, pages 1328–1339, Seattle, WA, USA, 2011.

[OGA$^+$06] Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: Lessons in Creating a Workflow Environment for the Life Sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.

[OR11] Luís M. L. Oliveira and Joel J. P. C. Rodrigues. Wireless Sensor Networks: a Survey on Environmental Monitoring. *Journal of Communications*, 6(2):143–151, 2011.

[PA06] Cesare Pautasso and Gustavo Alonso. Parallel Computing Patterns for Grid Workflows. In *Proc. of the 1st Workshop on Workflows in Support of Large-Scale Science*, WORKS '06, Paris, France, 2006. IEEE.

[Pie11] Henrik Andreas Pietranek. *Erweiterung von SIMPL und BPEL-DM zur Unterstützung weiterer Typen von Datenquellen.* Student thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2011.

[Pie12] Henrik Andreas Pietranek. *Datenmanagementpatterns in multiskalaren Simulationsworkflows.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2012.

[PZL08] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In *Proc. of the 17th International Conference on World Wide Web*, WWW '08, pages 805–814, Beijing, China, 2008. ACM.

[RB01] Erhard Rahm and Philip A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *International Journal on Very Large Data Bases (VLDB Journal)*, 10(4):334–350, 2001.

[RBD+11] Michael Reiter, Uwe Breitenbücher, Schahram Dustdar, Dimka Karastoyanova, Frank Leymann, and Hong-Linh Truong. A Novel Framework for Monitoring and Analyzing Quality of Data in Simulation Workflows. In *Proc. of the 7th IEEE International Conference on e-Science*, pages 105–112, Stockholm, Sweden, 2011.

[RBKK12] Michael Reiter, Uwe Breitenbücher, Oliver Kopp, and Dimka Karastoyanova. Quality-of-Data-Driven Simulation Workflows. In *Proc. of the 8th IEEE International Conference on e-Science*, Chicago, IL, USA, 2012.

[RBKK14] Michael Reiter, Uwe Breitenbücher, Oliver Kopp, and Dimka Karastoyanova. Quality of Data Driven Simulation Workflows. *Journal of Systems Integration*, 5(1):3–29, 2014.

[RD00] Erhard Rahm and Hong Hai Do. Data Cleaning: Problems and Current Approaches. *IEEE Data Engineering Bulletin*, 23, 2000.

[Rem11] Simon Remppis. *Ausführung einer Modellreduktion für Simulationen auf Basis der Workflow-Technologie.* Student thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2011.

[Rie14] Victor Riempp. *Pattern-basierte Kopplung eines biomechanischen und eines systembiologischen Simulationsmodells.* Student thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2014.

[RK11] Judith B. Rommel and Johannes Kästner. The Fragmentation-Recombination Mechanism of the Enzyme Glutamate Mutase Studied by QM/MM Simulations. *Journal of the American Chemical Society*, 26(133):10195–10203, 2011.

[RKH$^+$10] Oliver Röhrle, Nils Karajan, Thomas Heidlauf, Michael Sprenger, Tille Rupp, Syn Schmitt, and Wolfgang Ehlers. Towards a Better Understanding of Skeletal Muscle Models on Lumbar Spine Mechanics. In *Proc. of the World Congress on Biomechanics*, Singapore, 2010.

[RLSR$^+$06] Uwe Radetzki, Ulf Leser, Svenja Schulze-Rauschenbach, Jörg Zimmermann, Jens Lüssem, Thomas Bode, and Armin B. Cremers. Adapters, Shims, and Glue – Service Interoperability for in Silico Experiments. *Bioinformatics*, 22(9):1137–1143, 2006.

[RM09] Sylvia Radeschütz and Bernhard Mitschang. Extended Analysis Techniques for a Comprehensive Business Process Optimization. In *Proc. of the International Conference on Knowledge Management and Information Sharing (KMIS 2009)*, Madeira, Portugal, 2009.

[RRS$^+$11] Peter Reimann, Michael Reiter, Holger Schwarz, Dimka Karastoyanova, and Frank Leymann. SIMPL – A Framework for Accessing External Data in Simulation Workflows. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2011)*, pages 534–553, Kaiserslautern, Germany, 2011.

[RS13a] Benny Raphael and Ian F. C. Smith. *Engineering Informatics: Fundamentals of Computer-Aided Engineering.* John Wiley & Sons, Chichester, UK, 2013.

[RS13b] Peter Reimann and Holger Schwarz. Datenmanagementpatterns in Simulationsworkflows. In Gesellschaft für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2013)*, pages 279–293, Magdeburg, Germany, 2013.

[RS14] Peter Reimann and Holger Schwarz. Simulation Workflow Design Tailor-Made for Scientists. In *Proc. of the 26th International Conference on Scientific and Statistical Database Management*, Aalborg, Denmark, 2014.

[RSF06] Christopher Ré, Jérôme Siméon, and Mary Fernández. A Complete and Efficient Algebraic Compiler for XQuery. In *Proc. of the 22nd International Conference on Data Engineering*, ICDE '06, Washington, DC, USA, 2006. IEEE Computer Society.

[RSM11] Peter Reimann, Holger Schwarz, and Bernhard Mitschang. Design, Implementation, and Evaluation of a Tight Integration of Database and Workflow Engines. *Journal of Information and Data Management*, 2(3):353–368, 2011.

[RSM14a] Peter Reimann, Holger Schwarz, and Bernhard Mitschang. A Pattern Approach to Conquer the Data Complexity in Simulation Workflow Design. In R. Meersman et al., editor, *Proc. of the OnTheMove Federated Conferences and Workshops (OTM 2014), 22nd International Conference on Cooperative Information Systems (CoopIS 2014)*, volume 8841 of *LNCS*, pages 21–38, Amantea, Italy, 2014. Springer-Verlag, Berlin Heidelberg.

[RSM14b] Peter Reimann, Holger Schwarz, and Bernhard Mitschang. 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*, Aalborg, Denmark, 2014.

[RTD+12] Michael Reiter, Hong-Linh Truong, Schahram Dustdar, Dimka Karastoyanova, Robert Krause, Frank Leymann, and Dieter Pahr. On Analyzing Quality of Data Influences on Performance of Finite Elements driven Computational Simulations. In *Proc. of the International European Conference on Parallel and Distributed Computing (Euro-Par 2012)*, LNCS, pages 793–804, Rhodes Island, Greece, 2012. Springer-Verlag, Berlin, Heidelberg.

[RtHEvdA05a] Nick Russel, Arthur H. M. ter Hofstede, David Edmond, and Wil M. P. van der Aalst. Workflow Data Patterns: Identification, Representation and Tool Support. In *Proc. of the 24th International Conference on Conceptual Modeling*, ER 2005, pages 353–368, Klagenfurt, Austria, 2005. Springer-Verlag, Berlin, Germany.

[RtHEvdA05b] Nick Russel, Arthur H. M. ter Hofstede, David Edmond, and Wil M. P. van der Aalst. Workflow Resource Patterns: Identification, Representation and Tool Support. In *Proc. of the 17th Conference on Advanced Information Systems Engineering*, (CAiSE'05), pages 216–232, Porto, Portugal, 2005. Springer-Verlag, Berlin, Germany.

[RtHvdAM06] Nick Russel, Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, and Nataliya Mulyar. *Workflow Control-Flow Patterns: A Revised View.* BPM Center Report BPM-06-22, 2006.

[RvdAtH06] Nick Russel, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. *Exception Handling Patterns in Process-Aware Information Systems.* (CAiSE'06). Springer-Verlag, Berlin, Germany, Luxembourg, 2006.

[RWWS14] Peter Reimann, Tim Waizenegger, Matthias Wieland, and Holger Schwarz. Datenmanagement in der Cloud für den Bereich Simulationen und Wissenschaftliches Rechnen. In *Proc. of the 2nd Workshop Data Management in the Cloud (DMC 2014), in conjunction with the 44th Jahrestagung der Gesellschaft für Informatik (GI)*, Stuttgart, Germany, 2014.

[SA05] Stewart Silling and E. Askari. A Meshfree Method based on the Peridynamic Model of Solid Mechanics. *Computers & Structures*, 83(17-18):1526–1535, 2005.

[Sad13] Shazia Sadiq. *Handbook of Data Quality: Research and Practice.* Springer-Verlag, Berlin Heidelberg, Germany, 2013.

[SAKVH15] Steve Strauch, Vasilios Andrikopoulos, Dimka Karastoyanova, and Karolina Vukojevic-Haupt. *Migrating eScience Applications to the Cloud: Methodology and Evaluation.* Cloud Computing with E-science Applications. CRC Press, Taylor & Francis, 2015.

[Sch14] Holger Schwichtenberg. *Windows PowerShell 4.0: Das Praxisbuch.* Carl Hanser Verlag, Munich, Germany, 2014.

[SGK$^+$11] Mirko Sonntag, Katharina Görlach, Dimka Karastoyanova, Frank Leymann, Polina Malets, and David Schumm. Views on Scientific Workflows. In *Proc. of the 10th International Conference on Perspectives in Business Informatics Research*, pages 321–335, Riga, Latvia, 2011. Springer-Verlag, Berlin Heidelberg, Germany.

[She99] Amit P. Sheth. Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In Michael Goodchild, Max Egenhofer, Robin Fegeas, and Cliff Kottman, editors, *Interoperating Geographic Information Systems*, pages 5–29. Springer US, Boston, MA, USA, 1999.

[SK10] Mirko Sonntag and Dimka Karastoyanova. 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.

[SK11] Mirko Sonntag and Dimka Karastoyanova. Concurrent Workflow Evolution. *Electronic Communications of the EASST*, vol. 37, 2011.

[SK12] Mirko Sonntag and Dimka Karastoyanova. Ad hoc Iteration and Re-execution of Activities in Workflows. *International Journal On Advances in Software*, 5(1 & 2):91–109, 2012.

[SK13] Mirko Sonntag and Dimka Karastoyanova. Model-as-you-go: An Approach for an Advanced Infrastructure for Scientific Workflows. *Jorunal of Grid Computing*, 11(3):553–583, 2013.

[SKK$^+$11] David Schumm, Dimka Karastoyanova, Oliver Kopp, Frank Leymann, Mirko Sonntag, and Steve Strauch. Process Fragment Libraries for Easier and Faster Development of Process-based Applications. *Journal of Systems Integration*, 2(1):39–55, 2011.

[SKRM13] Stefan Silcher, Jan Königsberger, Peter Reimann, and Bernhard Mitschang. Cooperative Service Registries for the Service-based Product Lifecycle Management Architecture. In *Proc. of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD 2013)*, pages 439–446, Whistler, BC, Canada, 2013.

[SL90] Amit P. Sheth and James A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[Slo07] Alexander Slominski. Adapting BPEL to Scientific Workflows. In Ian Taylor, Ewa Deelman, and Dennis Gannon, editors, *Workflows for e-Science – Scientific Workflows for Grids*, chapter 14. Springer, London, UK, 2007.

[Son15] Tianhong Song. Provenance-Driven Data Curation Workflow Analysis. In *Proc. of the 2015 SIGMOD PhD Symposium*, pages 45–50, Melbourne, Victoria, Australia, 2015. ACM.

[SR09] Arie Shoshani and Doron Rotem. *Scientific Data Management: Challenges, Technology, and Deployment*. Computational Science Series. Chapman & Hall, Boca Raton, FL, USA, 2009.

[SSH$^+$10] Amruta Shiroor, John Springer, Thomas Hacker, Brandeis Marshall, and Jeffrey Brewer. Scientific Workflow Management Systems and Workflow Patterns. *International Journal of Business Process Integration and Management*, 5(1):63–78, 2010.

[SSS09] Dimitrios Skoutas, Alkis Simitsis, and Timos Sellis. Ontology-Driven Conceptual Design of ETL Processes Using Graph Transformations. In Stefano Spaccapietra, Esteban Zimányi, and Il-Yeol Song, editors, *Journal on Data Semantics XIII*, pages 120–146. Springer-Verlag, Berlin, Heidelberg, Germany, 2009.

[SSWY14] Yutian Sun, Jianwen Su, Budan Wuy, and Jian Yang. Modeling Data for Business Processes. In *Proc. of the 30th International Conference on Data Engineering*, ICDE 2014, Chicago, IL, USA, 2014. IEE.

[Sta15] John Stark. *Product Lifecycle Management – Volume 1: 21st Century Paradigm for Product Realisation*. Springer International Publishing, 2015.

[SWCD12] Alkis Simitsis, Kevin Wilkinson, Malu Castellanos, and Umeshwar Dayal. Optimizing Analytic Data Flows for Multiple Execution Engines. In *Proc. of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pages 829–840, Scottsdale, AZ, USA, 2012. ACM.

[SWLG04] Robert Stevens, Chris Wroe, Phillip Lord, and Carole Goble. Ontologies in Bioinformatics. In *Handbook on Ontologies*, pages 635–658. Springer-Verlag, Berlin Heidelberg, Germany, 2004.

[TDG07] Ian Taylor, Ewa Deelman, and Dennis Gannon. *Workflows for e-Science – Scientific Workflows for Grids.* Springer, London, UK, 2007.

[Tre10] Kevin E. Trenberth. *Climate System Modeling.* Cambridge University Press, Cambridge, UK, 2010.

[UGM14] Benjamin Ükermann, Bernhard Gatzhammer, and Miriam Mehl. Coupling Algorithms for Partitioned Multi-Physics Simulations. In *Proc. of the 1st Workshop on Simulation Technology: Systems for Data Intensive Simulations, in conjunction with the 44th Jahrestagung der Gesellschaft für Informatik (GI)*, Stuttgart, Germany, 2014.

[vdAtHKB03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, B. Kiepuszewski, and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, 2003.

[VHKL13] Karolina Vukojevic-Haupt, Dimka Karastoyanova, and Frank Leymann. On-demand Provisioning of Infrastructure, Middleware and Services for Simulation Workflows. In *Proc. of the 6th IEEE International Conference on Service Oriented Computing and Applications (SOCA)*, pages 91–98, Kauai, HI, USA, 2013.

[Vhr11] Marko Vhrovnik. *Optimierung datenintensiver Workflows: Konzepte und Realisierung eines heuristischen, regelbasierten Optimierers.* PhD thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2011.

[vS15a] Patrick von Steht. *Ontologiebasierte Beschreibung der Eingabedaten einer Knochensimulation.* Research thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2015.

[vS15b] Patrick von Steht. *Pattern-basierter Datenaustausch zwischen Simulationsmodellen.* Research thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2015.

[vS16] Patrick von Steht. *Datenintegration und Datenanalysen zur Unterstützung von CAE-Prozessen.* Master thesis, University of

Stuttgart, Institute of Parallel and Distributed Systems, 2016.

[VSRM08] Marko Vrhovnik, Holger Schwarz, Sylvia Radeschütz, and Bernhard Mitschang. An Overview of SQL Support in Workflow Products. In *Proc. of the 24th International Conference on Data Engineering (ICDE 2008)*, pages 1287–1296, Cancùn, México, 2008.

[VSS+07] Marko Vrhovnik, Holger Schwarz, Oliver Suhre, Bernhard Mitschang, Volker Markl, Albert Maier, and Tobias Kraft. An Approach to Optimize Data Processing in Business Processes. In *Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007)*, pages 615–626, Vienna, Austria, 2007.

[VZ14] Alejandro Vaisman and Esteban Zimányi. *Data Warehouse Systems: Design and Implementation.* Springer-Verlag, Berlin Heidelberg, Germany, 2014.

[W3C99] W3C: World Wide Web Consortium. *XML Path Language (XPath) Version 1.0*, 1999.

[W3C01] W3C: World Wide Web Consortium. *Web Services Description Language (WSDL) 1.1*, 2001.

[W3C04a] W3C: World Wide Web Consortium. *Web Services Addressing (WS-Addressing)*, 2004.

[W3C04b] W3C: World Wide Web Consortium. *XML Schema Part 0: Primer Second Edition*, 2004.

[W3C04c] W3C: World Wide Web Consortium. *XML Schema Part 1: Structures Second Edition*, 2004.

[W3C04d] W3C: World Wide Web Consortium. *XML Schema Part 2: Datatypes Second Edition*, 2004.

[W3C07a] W3C: World Wide Web Consortium. *Simple Object Access Protocol (SOAP) Version 2.0*, 2007.

[W3C07b] W3C: World Wide Web Consortium. *Web Services Description Language (WSDL) Version 2.0*, 2007.

[W3C07c] W3C: World Wide Web Consortium. *XSL Transformations (XSLT) Version 2.0*, 2007.

[W3C10a] W3C: World Wide Web Consortium. *XML Path Language (XPath) 2.0 (Second Edition)*, 2010.

[W3C10b] W3C: World Wide Web Consortium. *XQuery 1.0: An XML Query Language (Second Edition)*, 2010.

[W3C11] W3C: World Wide Web Consortium. *XQuery Update Facility 1.0*, 2011.

[W3C14a] W3C: World Wide Web Consortium. *XML Path Language (XPath) 3.0*, 2014.

[W3C14b] W3C: World Wide Web Consortium. *XQuery 3.0: An XML Query Language)*, 2014.

[W3C15] W3C: World Wide Web Consortium. *Extensible Markup Language (XML)*, 2015.

[Wag10] Florian Bernd Dominic Wagner. *Webservice und Workflow-Technologie für Proteinmodellierung.* Student thesis, University of Stuttgart, Institute of Architecture of Application Systems, 2010.

[Wag11] Florian Bernd Dominic Wagner. *Nutzung einer integrierten Datenbank zur effizienten Ausführung von Workflows.* Diploma thesis, University of Stuttgart, Institute of Parallel and Distributed Systems, 2011.

[WAH$^+$07] Katy Wolstencroft, Pinar Alper, Duncan Hull, Chris J. Wroe, Phillip W. Lord, Robert D. Stevens, and Carol A. Goble. The myGrid Ontology: Bioinformatics Service Discovery. *International Journal of Bioinformatics Research and Applications*, 3(3):303–325, 2007.

[WCL$^+$05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey, and Donald F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

[WD10] Daya C. Wimalasuriya and Dejing Dou. Ontology-based Information Extraction: An Introduction and a Survey of Current Approaches. *Journal of Information Science*, 36(3):306–323, 2010.

[Wes12] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures.* Springer-Verlag, Berlin Heidelberg, Germany, 2012.

[WVV+01] Holger Wache, Thomas Vögele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-Based Integration of Information – A Survey of Existing Approaches. In *Proc. of the IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, Seattle, WA, USA, 2001.

[YGN09] Ustun Yildiz, Adnene Guabtni, and Anne H. H. Ngu. Towards Scientific Workflow Patterns. In *Proc. of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, Portland, OR, USA, 2009. ACM.

[ZBKL10] Daniel Zinn, Shawn Bowers, Sven Köhler, and Bertram Ludäscher. Parallelizing XML Data-Streaming Workflows via MapReduce. *Journal of Computer and System Sciences*, 76(6):447–463, 2010.

[ZBML09] Daniel Zinn, Shawn Bowers, Timothy McPhillips, and Bertram Ludäscher. Scientific Workflow Design with Data Assembly Lines. In *Proc. of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS'09, pages 14:1–14:10, Portland, OR, USA, 2009. ACM.

[ZTZ13] Olgierd Cecil Zienkiewicz, Robert L. Taylor, and J.Z. Zhu. *The Finite Element Method: its Basis and Fundamentals.* Butterworth-Heinemann, Elsevier, Oxford, UK, 7th edition, 2013.

All links in this bibliography and throughout the rest of this document have last been visited and found working on December 20, 2016.

List of Figures

List of Tables