Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Studienarbeit Nr. 2448

Pattern-basierte Definition der Datenbereitstellung für Simulationen zu Strukturänderungen in Knochen

Andreas Michael Bohrn

Studiengang:	Informatik
Prüfer/in:	PD Dr. rer. nat. habil. Holger Schwarz
Betreuer/in:	Dipl.-Inf. Peter Reimann
Beginn am:	19. Dezember 2013
Beendet am:	20. Juni 2014
CR-Nummer:	H.2.5, H.4.1, I.6.7

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

1 Einleitung

Workflows haben sich im Business Bereich bereits seit langem bewährt. Nun werden sie auch im wissenschaftlichen Bereich immer häufiger eingesetzt [TDGS07]. Eines der Haupteinsatzgebiete ist die Modellierung von Simulationsabläufen [GSK+11]. Oftmals müssen dabei riesige Datensätze in unterschiedlichen Formaten für die Simulationsumgebung bereitgestellt und in passende Formate transformiert werden. Dies führt dazu, dass sich Wissenschaftler nicht mehr auf ihre Kernkompetenzen konzentrieren können und erhöht das Risiko von Fehlern bei der Umsetzung der Datenbereitstellung [RSM14].

Eine Möglichkeit dieses Problem zu lösen sind sogenannte Workflow Patterns [RSM14]. Patterns reduzieren die Menge an Eingabevariablen die ein Wissenschaftler vor der Ausführung einer Simulation bereitstellen muss und abstrahieren komplexe Datentransfer- und Transformationsschritte. Dadurch kommt es zu einer Trennung der Zuständigkeiten bei der Implementierung einer Simulation. Wissenschaftler können sich auf die eigentliche Simulation konzentrieren, während beispielsweise Datenbank Experten die nötigen Patterns für die Datenbereitstellung implementieren.

1.1 Aufgabenstellung

Im Zentrum dieser Arbeit steht die Implementierung einer Simulation der Strukturänderungen in Knochen [KSR+11]. Bei dieser Simulation kommen zwei Simulationsmodelle zum Einsatz: Ein biomechanisches Modell simuliert den Massenaustausch zwischen Festkörpern und Flüssigkeiten innerhalb des porösen Knochengewebes. Für diese Simulation wird das auf der Finite Elemente Methode (FEM) beruhende Pandas Rahmenwerk genutzt[pan]. Nachdem Pandas die Berechnungen für das biomechanische Simulationsmodell abgeschlossen hat, wird ein systembiologisches Simulationsmodell genutzt, um Veränderungen auf der Zellebene zu berechnen. Durchgeführt wird diese Simulation unter Verwendung der Rechenumgebung Gnu Octave [oct]. Anschließend werden die Ergebnisse der systembiologischen Simulation genutzt, um die Knochenkonfiguration der biomechanischen Simulation anzupassen und weitere Zeitschritte berechnen zu können.

In einer vorangegangenen Arbeit wurden für diese Simulation bereits erste Workflows erstellt: Ein biomechanischer Workflow, der das Pandas Rahmenwerk nutzt, ein systembiologischer Workflow, welcher noch die Matlab Rechenumgebung nutzte und ein Kopplungsworkflow, der im Wesentlichen die Datenintegration zwischen den anderen beiden Workflows steuert [Pie12][Dor11]. Die Datenbereitstellung in diesen Workflows wurde noch sehr rudimentär und zudem für ein spezielles Simulationsproblem umgesetzt.

Im Rahmen dieser Arbeit werden der systembiologische und der biomechanische Workflow angepasst, um generisch für unterschiedliche Knochen und dazugehörige Randbedingungen eingesetzt

werden zu können. Außerdem werden für die Datenbereitstellungsschritte der beiden Workflows entsprechende Patterns für jede Ebene der Patternhierarchie [RS13] erarbeitet und deren Transformation mittels entsprechender Abbildungsregeln und zugehöriger Metadaten bis hin zu ausführbaren Workflowfragmenten implementiert. Abschließend werden die erarbeiteten Patterns auf ihre Abstraktionsunterstützung, sowie generische Einsetzbarkeit hin bewertet.

1.2 Gliederung

Diese Arbeit ist folgendermaßen gegliedert.

Kapitel 2 Grundlagen
Dieses Kapitel beschäftigt sich mit den Grundlagen dieser Arbeit. Es werden Konzepte und Sprachen, die für das Verständnis der restlichen Arbeit nötig sind, näher erläutert.

Kapitel 3 Knochensimulation
In diesem Kapitel wird die Simulation von Strukturänderungen in Knochen genauer beschrieben.

Kapitel 4 vorangegangene Arbeiten
In diesem Kapitel wird der Stand der in diesem Bereich bereits zuvor erbrachten Arbeiten aufgeführt, um die in Zuge dieser Arbeit erbrachten Leistungen genauer abzugrenzen.

Kapitel 5 konzeptioneller Entwurf
Dieses Kapitel beschreibt den konzeptionellen Entwurf des biomechanischen, sowie des systembiologischen Workflows und der dazugehörigen Patterns.

Kapitel 6 Implementierung
In diesem Kapitel wird die konkrete Implementierung der zuvor beschriebenen Workflows und Patterns beschrieben.

Kapitel 7 Bewertung
Dieses Kapitel beschäftigt sich mit der Bewertung der zuvor beschriebenen Patterns. Als Kriterien gelten dabei zum einen der Grad der Abstraktionsunterstützung und zum anderen inwiefern sich die Patterns generisch einsetzen lassen.

Kapitel 8 Zusammenfassung
In diesem Kapitel werden die Ergebnisse der Arbeit noch einmal zusammengefasst.

2 Grundlagen

Das folgende Kapitel beschäftigt sich mit den Grundlagen, die für das Verständnis der nachfolgenden Arbeit relevant sind. Dazu gehören unter anderem XML, serviceorientierte Architektur, Datenbanken, workflow-basierte Simulation und Ontologien

2.1 XML

Die Abkürzung XML steht für ”eXstensible Markup Language” (erweiterbare Auszeichnungssprache). Mit XML können Daten in einer hierarchisch strukturierten Form dargestellt werden. XML wird heute hauptsächlich zum plattform- und implementationsunabhängigen Austausch von Daten verwendet. Die erste XML Spezifikation wurde vom World Wide Web Consortium (W3C) am 10. Februar 1998 veröffentlicht. Die momentan aktuelle fünfte Version der XML Spezifikation wurde am 26. November 2008 veröffentlicht [BPSM$^+$]. Die folgenden beiden Abschnitte basieren, soweit im Text nicht anders angegeben, auf den Quellen [Sch03] und [Seb10].

2.1.1 Aufbau einer XML-Datei

Der standardmäßige Aufbau eines XML Dokuments sieht wie folgt aus:

- Ein optionaler XML Kopf
- Eine optionale Schema Definition
- Ein Wurzelelement

Listing 2.1 zeigt ein Beispiel für einen XML Kopf. Der Kopf gibt die verwendete XML Version an sowie die Zeichenkodierung des Dokuments. Der Kopf ist zwar optional, jedoch erleichtert er die spätere Verarbeitung eines XML Dokuments. Das Schema legt die zugrundeliegende Struktur des eigentlichen Dokumentinhaltes fest und ist vor allem für den Austausch von Daten wichtig. Das Schema eines XML Dokuments kann entweder als Document Type Definition(DTD), oder in Form eines XML-Schemas erfolgen. Aufbau und Funktion einer XML Schema Datei wird in 2.1.2 beschrieben.

```
<?XML version="1.0" encoding="UTF-8"?>
```

Listing 2.1: XML Kopfzeile

Der eigentliche Inhalt des XML Dokuments wird durch das Wurzelelement beschrieben. XML Dokumente besitzen eine baumartige Struktur. Jedes Element mit Ausnahme des Wurzelelementes hat genau ein Elternelement und beliebig viele Kindelemente. Ein XML Element wird durch sogenannte ”Tags” eingeleitet.. In Listing 2.2 ist <Person> das Start Tag des Elements Person und </Person> das dazugehörige End-Tag. Alles was zwischen Start- und Endtag steht, ist der eigentliche Inhalt des Elementes. Der Inhalt eines Elementes kann entweder ein einfacher Text, weitere Kindelemente, oder auch eine Mischung aus beiden Varianten sein. Elemente können beliebig tief geschachtelt werden. Zusätzlich können Elemente noch mit Attributen versehen werden. Attribute werden innerhalb des Start Tags eines Elements definiert. Im Beispiel aus 2.2 hat das Element Geburtstag ein Attribut Alter mit dem Wert "37".

```
<Person>
  <Name>
    <Vorname> Max </Vorname>
    <Nachname> Mustermann </Nachname>
  </Name>
  <Geburtstag Alter="37"> 1.1.1977 </Geburtstag>
```

Listing 2.2: Beispiel Element

2.1.2 XML Schema und DTD

XML wird hauptsächlich zum Austausch von Daten genutzt. Für diesen Zweck ist es wichtig, dass alle Kommunikationspartner wissen, welche Struktur die ausgetauschten Daten haben. XML Schema ist eine vom World Wide Web Consortium entwickelte Beschreibungssprache, um die Struktur eines XML Dokuments festzulegen.

XML Schema unterscheidet zwischen zwei Sorten von Datentypen: Einfache Datentypen, wie z.b. Strings, Integer Zahlen, Boolean Werte, etc. und komplexe Datentypen, die aus Elementen und Attributen zusammengesetzt werden können. Listing 2.3 zeigt, wie ein Schema für einen Brief aussehen könnte. Brief ist hier ein komplexer Datentyp mit dem Attribut Versanddatum und den Unterelementen Absender, Empfänger, Adresse, Postleitzahl und Inhalt. Die Elemente eines komplexen Datentyps können auf unterschiedliche Weise miteinander verknüpft werden. In diesem Beispiel werden sie als xs:sequence definiert. Eine Sequenz ist eine sequentiele Liste, in der jedes Element entweder keinmal, genau einmal, oder beliebig oft vorkommen kann. Das Element Inhalt wurde mit dem Attribut minOccurs = ”0” versehen und muss somit nicht zwingend in der Sequenz vorkommen. Werden keine näheren Angaben gemacht, wird sowohl MinOccurs als auch MaxOccurs auf den Wert eins gesetzt, das bedeutet ein Element darf nur genau einmal in der Liste vorkommen. Mit xs:choice können die Kindelemente als Liste von Alternativen, aus der genau eine ausgewählt werden kann, definiert werden. Zuletzt gibt es noch die Möglichkeit die Kindelemente mit xs:all als eine Liste mit beliebiger Reihenfolge, in der jedes Element höchstens einmal auftreten darf, zu definieren.

```
<xs:complexType name="Brief">
  <xs:sequence>
    <xs:element name="Absender" type="xs:string"/>
    <xs:element name="Empf"anger" type="xs:string"/>
    <xs:element name="Adresse" type="xs:string"/>
    <xs:element name="Postleitzahl" type="xs:integer"/>
    <xs:element name="Inhalt"  type="xs:String" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="Versanddatum" type="xs:date"/>
</xs:complexType>
```

Listing 2.3: XML Schema Beispiel

2.2 Serviceorientierte Architektur

Der Begriff serviceorientierte Architektur (SOA) bezieht sich nicht auf eine konkrete Technologie, sondern beschreibt vielmehr einen Architekturstil aus der Informatik. Bei diesem Architekturstil steht das Konzept des Services (Dienst) im Mittelpunkt. Es gibt keine allgemeingültige Definition einer SOA, aber in [Mel10] wird die serviceorientierte Architektur auf diese Art definiert:

> ”Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuell inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform und sprachenunabhängige Nutzung und Wiederverwendbarkeit ermöglicht.”

Diese Services stellen ihre Funktionalitäten dabei über eine öffentlich definierte Schnittstelle zur Verfügung. Eine Anwendung kann den Service über seine Schnittstelle dynamisch binden und ist mit diesem nur lose gekoppelt, d.h. Services sind prinzipiell austauschbar.

Die unterschiedlichen Rollen in einer SOA werden in Abbildung 2.1 in Form des sogenannten SOA Dreiecks dargestellt. Der Service Provider stellt eine Funktionalität in Form eines Services bereit und registriert diesen Service danach in der Service Registry. Die Registry enthält für jeden registrierten Service eine abstrakte Servicebeschreibung und die nötigen Informationen, um den Service aufrufen zu können. Der Service Client durchsucht die Service Registry nach Services, die seinem Bedürfnissen entsprechen. Über die in der Registry hinterlegten Daten kann der Client dann die Schnittstellenbeschreibung des gewählten Services beim Service Provider erfragen. Die Schnittstellenbeschreibung erlaubt es dem Client den Service zu nutzen. Dieser Aufbau erlaubt eine hohe Wiederverwertbarkeit bestehender Services und sorgt für eine große Flexibilität bei der Auswahl des Services.

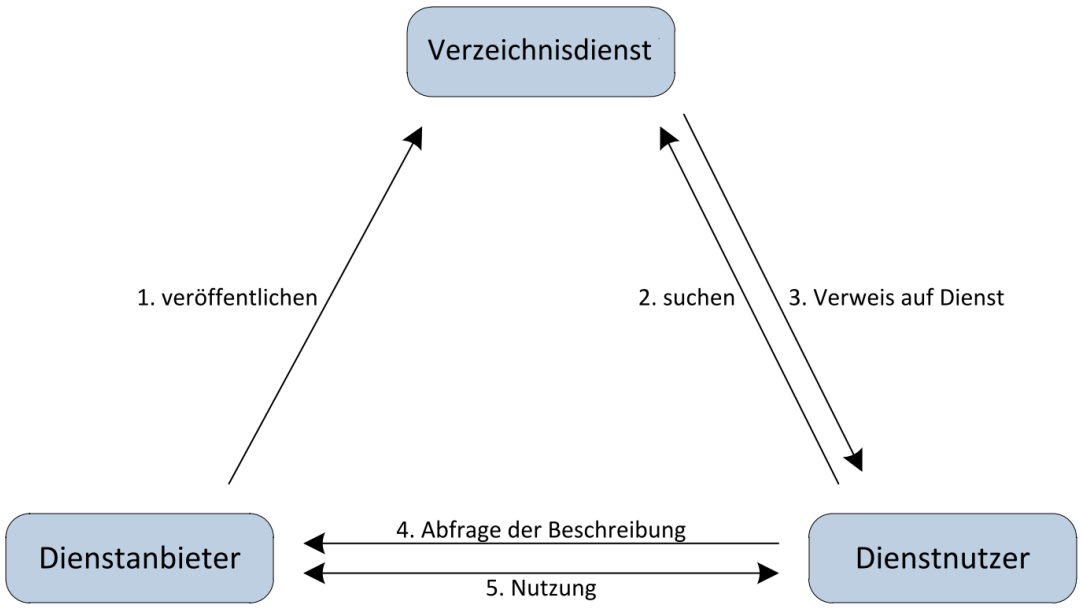


Abbildung 2.1: SOA Dreieck Quelle: [Mel10]

2.3 Webservice

Webservices sind eine mögliche Implementierung der serviceorientierten Architektur. Ein Webservice stellt eine oder mehrere Funktionalitäten an einer festen URI zur Verfügung. Wie auf diese Funktionalitäten zugegriffen werden kann, wird dabei meist durch eine Web Service Definition Language (WSDL) Datei öffentlich beschrieben. Näheres zu WSDL findet sich in Abschnitt 2.3.2. Der Nachrichtenaustausch mit einem Webservice geschieht in der Regel über das SOAP Protokoll, welches im Abschnitt 2.3.1 behandelt wird.

2.3.1 SOAP

SOAP (ursprünglich Simple Object Access Protocol) ist ein Protokoll, mit dem Daten zwischen Systemen oder Anwendungen ausgetauscht werden können. SOAP verwendet XML Schema für die Definition der ausgetauschten Daten und kann auf unterschiedlichen Transportprotokollen definiert werden. Eine SOAP Nachricht besteht aus einem Envelope, einem oder mehreren Header Feldern und einem Body Element. Listing 2.4 zeigt eine einfache Soap Nachricht.

Die Header einer SOAP Nachricht sind für die Verarbeitung auf dem Transportweg gedacht. Zwischenstationen können die Header einer SOAP Nachricht lesen und entsprechend Veränderungen an der Nachricht durchführen (z.B. Verschlüsselung der Nachricht). Die für den Empfänger relevanten Daten befinden sich innerhalb des Body Elements.

```
<?xml version="1.0"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header>
    ...
  </soapenv:Header>
  <soapenv:Body>
    ...
  <soapenv:Body>
</soapenv:Envelope>
```

Listing 2.4: Beispiel einer SOAP Nachricht

2.3.2 WSDL

Die Abkürzung WSDL steht für Web Service Description Language. WSDL ist eine auf XML basierte Beschreibungssprache für Webservices. Abbildung 2.2 zeigt eine schematische Darstellung einer WSDL Datei. Eine WSDL Datei wird in zwei Bereiche aufgeteilt: den Abstrakten Bereich, der unabhängig von Protokollen ist, und dem konkreten Bereich, in dem die Abbildung der abstrakten Definitionen auf konkrete Protokolle und Dateiformate beschrieben wird.

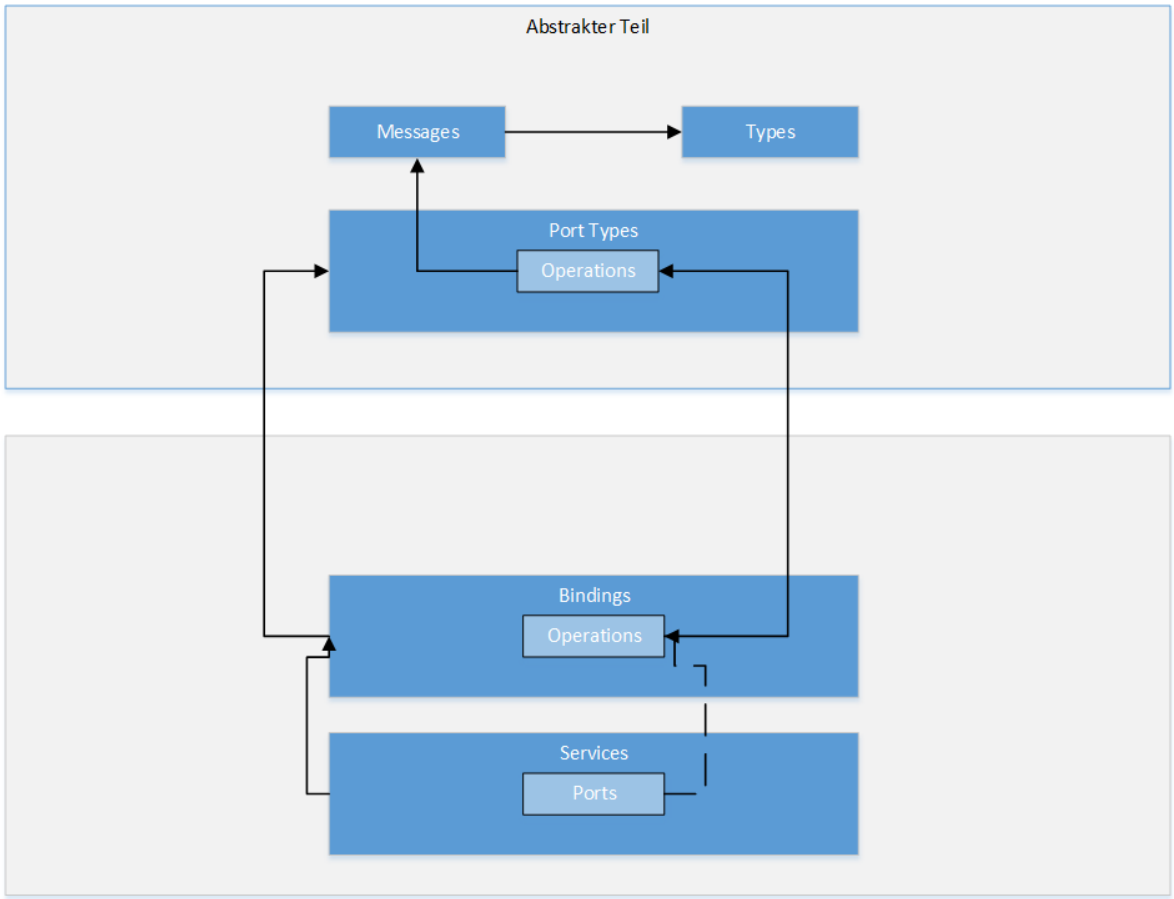


Abbildung 2.2: schematischer Aufbau einer WSDL Datei

Der Abstrakte Teil einer WSDL stellt das Service Interface des definierten Web Services dar. Jeder Web Service definiert einen oder mehrere Port Types. Ein Port Type enthält eine oder mehrere Operationen. Eine Operation beschreibt eine Funktion, die der Service bereitstellt, in Form ihrer Input und Output Nachrichten. Dabei werden komplexe XML Typen verwendet, um den Inhalt der Nachrichten zu sprezifizieren.

Im konkreten Teil der WSDL Datei werden die Service definiert. Service fassen die Menge an Ports eines Port Types zusammen. Ein Port ist dabei ein konkreter Endpunkt (meist eine URI) an dem der Webservice erreichbar ist. Jeder Port hat dabei ein Binding, welches die abstrakten Operationen, welche durch den Port Type beschrieben werden, auf konkrete Protokolle und Datenformate abbildet. Listing

2.4 Workflows

Ein Workflow (Arbeitsablauf) beschreibt die Reihenfolge in der eine Menge von Tätigkeiten abgearbeitet werden. In einem Workflow können mehrere Aktivitäten zu Komponenten zusammengefasst werden, wodurch sich die Wiederverwendbarkeit erhöht. Ihren Ursprung haben Workflows vor allem im Bereich der Unternehmen [LR00], aber sie werden zunehmend auch im wissenschaftlichen Bereich eingesetzt, z.B. für die automatische Ausführung komplexer Simulationen [GSK+11].

2.4.1 WS-BPEL

WS-Business Process Execution Language (WS-BPEL) ist eine XML-basierte Sprache zur Beschreibung von Workflow Abläufen[Jor07]. WS-BPEL bietet die Möglichkeit Webservice Aufrufe in einen Workflow einzubinden und diesen Workflow selbst ebenfalls als Webservice zur Verfügung zu stellen. Dies erhöht die Wiederverwendbarkeit derartiger Workflows, da sie für die Implementierung weiterer Workflows verwendet werden können.

Die Einbindung von Webservices in einen BPEL Prozess erfolgt durch die WSDL Datei des verwendeten Webservice. Dies wird auch als PartnerLink bezeichnet. In diesem Zusammenhang sind die sogenannten PartnerLink Types, also die Typen von verwendbaren PartnerLinks, wichtig. PartnerLink Types werden in der Regel in WSDL Dateien definiert. PartnerLink Types beschreiben, wie ein Prozess mit einem Partner interagieren kann und welche Port Types von beiden Seiten unterstützt werden. Dabei werden sogenannte Roles genutzt, um beiden Seiten eines PartnerLinks ihre Rolle in der Interaktion zuzuweisen. Listing 2.5 zeigt ein einfaches Beispiel für die konkrete Definition eines PartnerLink Types und eines dazugehörigen PartnerLinks. Der PartnerLinkType ”beispielPLT” definiert eine Rolle ”pltProvider” mit einem dazugehörigen PortType, den Web Services mit der Rolle "pltProvideränbieten. Im konkreten PartnerLink ”beispielPL” wird dem Partner des BPEL-Prozesses diese Rolle zugeweisen ("partnerRole"), d.h. der Prozess erwartet, dass der Partner Webservice den entsprechenden PortType zur Verfügung stellt. Ein BPEL Prozess kann für einen ParnterLink über myRole auch angeben, welchen Port Type der BPEL Prozess selbst dem Partner zur Verfügung stellt.

```
<partnerLinkType name="beispielPLT">
<role name="pltProvider">
<portType name="ns:beispielPT"/>
</role>
</partnerLinkType >

<partnerLink name="beispielPL" partnerLinkType ="beispielPLT"
partnerRole="pltProvider"/>
```

Listing 2.5: PartnerLink Beispiel

Für die eigentliche Definition des Workflows verwendet BPEL sogenannte Activities und Variablen. Activities lassen sich in zwei Gruppen unterteilen:

Simple Activities sind atomar, d.h. sie sind nicht aus anderen Activities aufgebaut. Zu ihnen zählen unter anderem eine Assign Activity, um Variablen einen konkreten Wert zuzuweisen, eine Invoke Activity für den Aufruf eines Webservice und eine Receive Activity um bei einer asynchronen Kommunikation mit einem Web Service Nachrichten von diesem Service empfangen zu können.

Komplexen Activities beeinhalten andere Activities. Sie werden hauptsächlich zur Beschreibung des Kontrollflusses innerhalb des Workflows genutzt. Zu den komplexen Activities gehören unter anderem die If, while und for each Activities, sowie eine sequence Activity für sequentielle Ausführung von Worflow Schritten, und eine flow Activity für die parallele Ausführung.

Außerdem gibt es noch sogenannte Scopes. Ein Scope bündelt mehrere Activities zu einer transaktionalen Einheit. Innerhalb eines Scopes können Variablen mit lokaler Sichtbarkeit definiert werden. Außerdem können einem Scope Fault Handler, für die Behandlung von Fehlern, und Compensation Handler, für die Kompensation der durchgeführten Activities, zugeordnet werden. Das Wurzelelement eines BPEL Prozesses ist stellt ebenfalls einen eigenen Scope dar.

2.5 Datenbanksysteme

Die folgenden Abschnitte basieren, soweit nicht anders angegeben, auf der Quelle [KE11]. Ein Datenbanksystem (DBS) wird im allgemeinen in zwei Komponenten aufgeteilt: Die eigentliche Datenbank, in der die Daten gespeichert werden, und dem Datenbankverwaltungssystem (engl. database management system DBMS), welches die Verwaltung und Verarbeitung der Daten in der Datenbank übernimmt.

Ein Datenbanksystem bietet eine Reihe von Vorteilen bei der Verwaltung der Daten. Zum einen verringert ein Datenbanksystem den Grad der Redundanz gespeicherter Daten im Vergleich zu einem normalen Dateisystem. Zum anderen erlaubt es einen leichteren Zugriff auf die gespeicherten Daten durch den Einsatz sogenannter Abfragesprachen.

2.5.1 SQL

SQL(structured query language) ist eine Datenbanksprache für die Definition, Bearbeitung und Abfrage von Daten in einer relationalen Datenbank. Relationale Datenbanken bestehen aus einer Sammlung verknüpfter Tabellen. Die einzelnen Zeilen(auch Tupel genannt) einer Tabelle(auch Relation genannt) stellen dabei einen Datensatz dar. Die Tupel setzen sich aus einzelnen Attributwerten, den Spalten der Tabelle, zusammen. Als Primärschlüssel bezeichnet man dabei eine Teilmenge der Attributwerte, welche ein Tupel der Tabelle eindeutig identifiziert. Ein Fremdschlüssel ist eine Menge von Attributen innerhalb einer Relation, welche auf den Primärschlüssel einer anderen (oder auch der gleichen) Relation verweist. Fremdschlüssel werden genutzt, um Beziehungen zwischen Relationen darzustellen.

SQL ist eine deklarative Abfragesprache, das heißt eine Abfrage in SQL bestimmt nur, welche Daten von der Datenbank zurückgegeben werden sollen, die eigentliche Verarbeitung übernimmt das Datenbankverwaltungssystem.

2.6 zeigt eine einfache Beispielabfrage in SQL. Dabei definiert der FROM-Abschnitt die Tabellen, die für die Anfrage relevant sind. Der SELECT-Abschnitt gibt, an welche Spalten der jeweiligen Tabellen in der Antwort auftauchen sollen, und der WHERE-Teil erlaubt es, die Ergebnisse der Anfrage auf eine oder mehrere Bedingungen zu selektieren. In diesem Beispiel liefert die Anfrage alle Spalten (*) der Tabelle Person, bei denen der Wert Gehalt größer als eintausend ist.

```
SELECT *
FROM Person p
WHERE p.Gehalt < 1000
```

Listing 2.6: SQL Abfrage

2.6 Ontologien

Der folgende Abschnitt basiert auf [Stu09]. Eine Ontologie ist eine formale und geordnete Darstellung von einer Menge an Begriffen und Konzepten aus einem abgegrenzten Themenbereich und den Beziehungen dieser Begriffe untereinander. Ontologien werden genutzt, um Wissen in einem Themengebiet in einer formalen und für maschinelle Verarbeitung geeigneten Form zu definieren. Um diese Ziele umzusetzen, wurden standardisierte Modellierungssprachen für Ontologien entwickelt.

Eine der im Bereich der Informatik am häufigsten verwendete Ontologiesprache ist die Web Ontology Language (OWL). OWL ist eine Spezifikation des World Wide Web Consortiums(W3C) und eine formale Beschreibungssprache für Ontologien, die vor allem für die Verwendung im Bereich Semantic Web entworfen wurde. OWL basiert auf dem Ressource Description Framework (RDF) Datenmodell und verwendet die XML Syntax.

Das RDF Datenmodell besteht aus binären Relationen, die als Tripel der Form Subjekt, Prädikat und Objekt dargestellt werden. Ein solches Tripel sagt aus, dass das Subjekt in einer durch das Prädikat

näher definierten Relation zu dem Objekt des Tripels steht. Mehrere solche Relationen können auch als Graph dargestellt werden.

Abbildung 2.3 zeigt ein einfaches Beispiel eines RDF-Modells. In diesem Beispiel ist ”Max Muster” Subjekt einer durch das Prädikat ”wohnt in” näher beschriebenen Relation mit dem Objekt ”Stuttgart”. Gleichzeitig zeigt das Beispiel auch, dass ein und die selbe Ressource unterschiedliche Rollen in unterschiedlichen Tripeln einnehmen kann. So ist ”Stuttgart” zum einen Objekt des vorher beschriebenen Tripels und zum anderen Subjekt eines weiteren Tripels mit Prädikat ”hat Postleitzahl” und Objekt ”70173”.

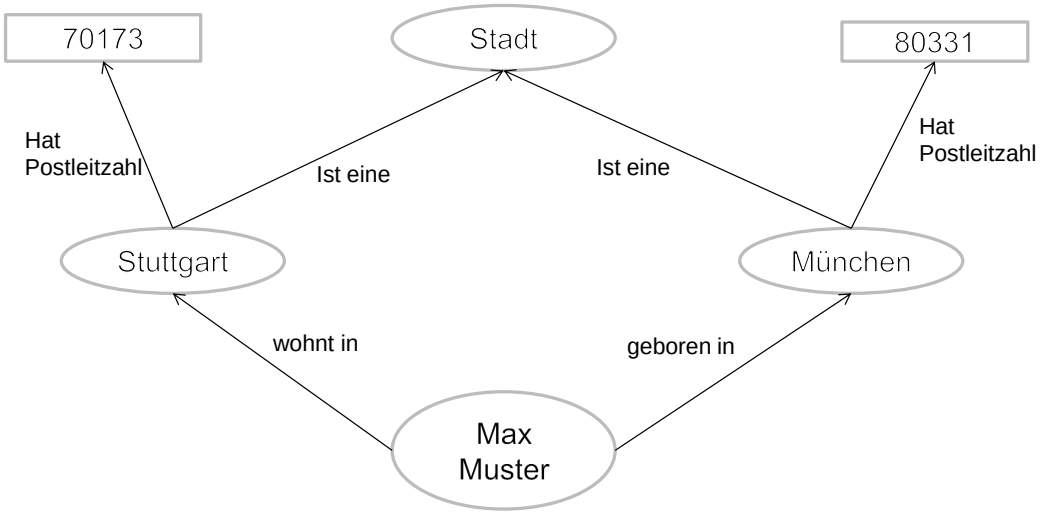


Abbildung 2.3: Einfaches Beispiel eines RDF-Modells

Im folgenden sollen die wichtigsten sprachlichen Konstrukte von OWL erklärt werden:

owl:class : Bezeichnet eine Klassendefinition.

owl:thing : Bezeichnet ein Individuumeiner bestimmten Klasse.

rdf:type : Relation, die einem Individuum eine Klasse zuordnet.

owl:objectproperties : Relationen zwischen Individuen zweier Klassen.

owl:datatypeproperties : Relationen zwischen Individuen und Literalen (Zahlen, Strings, etc.).

Das grundlegendste Element der OWL Beschreibungssprachen sind Klassen. Klassen werden durch owl:class definiert. Klassen können von anderen Klassen mittel owl:subClassof abgeleitet werden. Desweiteren können Klassen z.B. durch Vereinigung, Schnitt oder komplementärer Menge zu anderen Klassen gebildet werden. Individuen werden durch owl:thing definiert und mit der Relation rdf:type einer Klasse zugeordnet. Relationen zwischen Individuen lassen sich durch owl:opbejtproperties definieren. Außerdem können Relationen zwischen Individuen und Literalwerten wie Strings oder Integer Zahlen mit owl:datatypeproperties definiert werden.

3 Knochensimulation

In diesem Kapitel wird die für diese Arbeit relevante Simulation der Strukturänderung in Knochen näher beschrieben.

3.1 Simulation der Strukturänderungen in Knochen

Der für diese Arbeit relevante Anwendungsfall ist die Simulation der Strukturänderung in Knochen. Diese Simulation spielt vor allem bei der Untersuchung von Heilungsprozessen nach Knochenbrüchen eine Rolle [KSR+11].

Diese Simulation lässt sich in zwei Teilgebiete unterteilen: Die biomechanische Simulation simuliert Massenbewegungen zwischen porösen Medien und den darin enthaltenen Flüssigkeiten auf der Ebene des Knochengewebes . Die systembiologische Simulation bestimmt Gewebebildungsprozesse auf der Zellebene. Für die biomechanische Simulation wird das Pandas Rahmenwerk[pan] verwendet, die systembiologische Simulation wird durch die Rechenumgebung GNU Octave [oct] durchgeführt.

Abbildung 3.1 zeigt die Kopplung dieser beiden Simulationsmodelle. Die Simulation erfolgt anhand von typischen Bewegungssequenzen. Zunächst berechnet die biomechanische Simulation für jede dieser Bewegungssequenzen die makroskopischen Veränderungen der Knochenstruktur. Die systembiologische Simulation nutzt diese Berechnungen für einzelne Bewegungssequenzen und setzt deren Ergebnisse gemäß eines idealisierten Tagesablaufs zusammen. Die Ergebnisse der systembiologischen Simulation werden genutzt, um die Knochenkonfiguration der biomechanischen Simulation für den nächsten Zeitschritt, z.B. den nächsten zu betrachteten Tag, anzupassen. Dieser Vorgang wiederholt sich, solange bis alle relevanten Zeitschritte berechnet wurden.

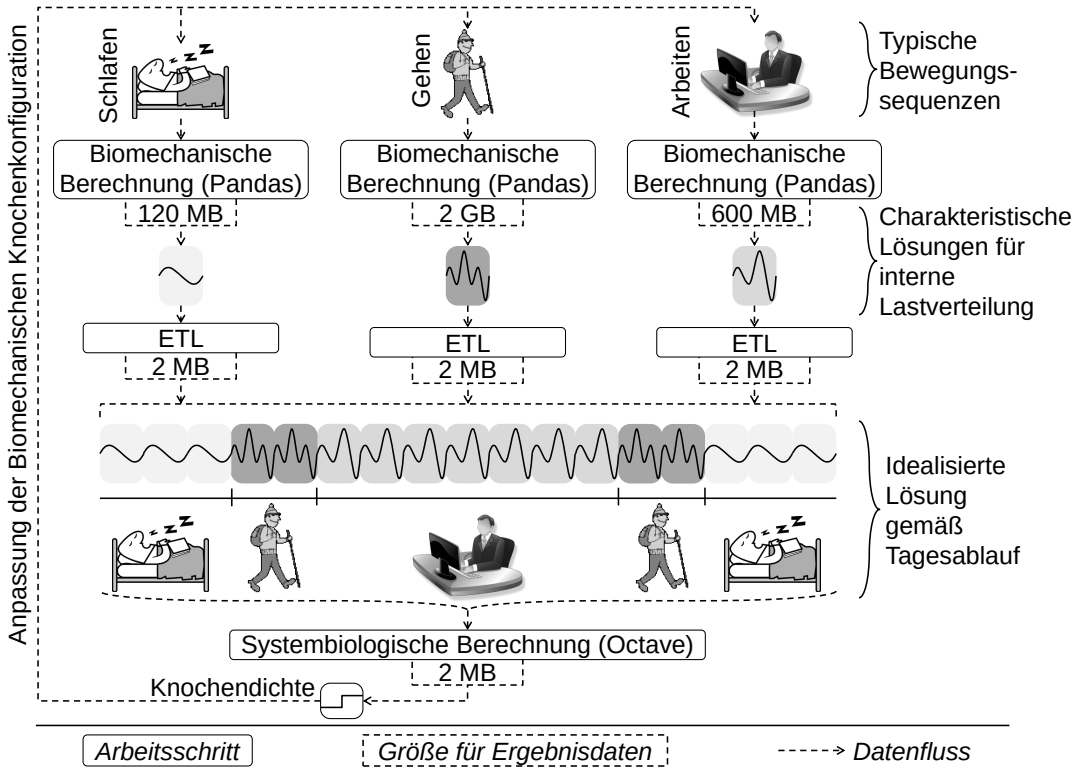


Abbildung 3.1: Kopplung der Simulationsmodelle. Quelle: [RWWS14]

Abbildung 3.2 zeigt einen Ausschnitt der konkreten Variablen der beiden Simulationsmodelle. Die von der biomechanischen Simulation errechneten Ausgabewerte lokale Zuglast und Wachstumsenergie werden als Eingabewerte der systembiologischen Simulation genutzt (Pfeile eins und zwei der Abbildung). Wenn die systembiologische Simulation mit ihrer Berechnung fertig ist, werden ihre Ausgabewerte genutzt, um die dazugehörigen Anfangsbedingungen der biomechanischen Simulation für den nächsten zu berechnenden Zeitschritt anzupassen.

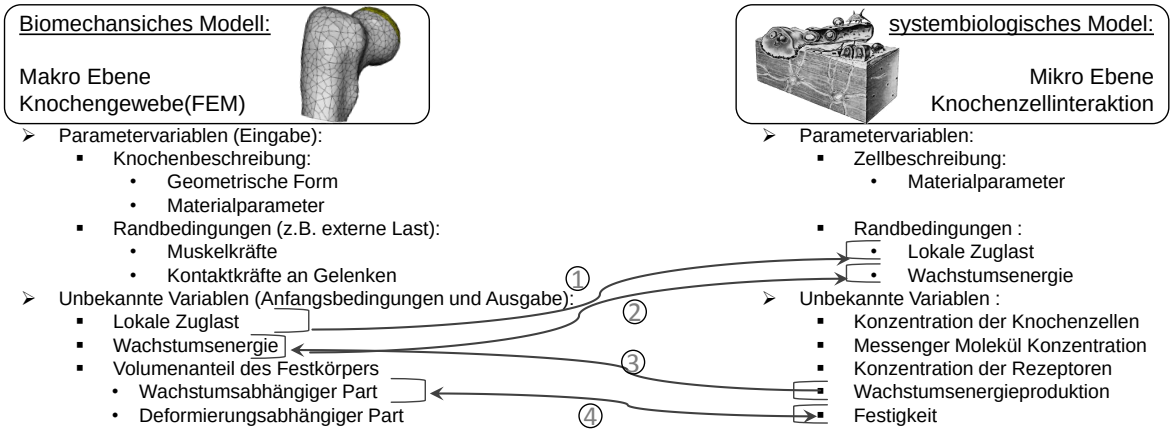


Abbildung 3.2: Simulationsmodelle und ihre Variablen. Vergleich Quelle: [RSM13]

Die Aufteilung der Simulation erfolgt in drei eigenständigen Workflows; den Kopplungsworkflow, den systembiologischen Workflow und den biomechanischen Workflow [RSM14]. Abbildung 3.3 zeigt den Aufbau des Kopplungsworkflows. Der Kopplungsworkflow teilt die verschiedenen Bewegungssequenzen auf die zur Verfügung stehenden Pandas Instanzen auf und ruft für jede Bewegungssequenz den biomechanischen Workflow auf. Der biomechanische Workflow übernimmt die Datenbereitstellung für das Pandas Rahmenwerk und ruft einen Pandas Webservice auf, der die eigentliche Simulation durchführt. Das Ergebnis dieser Simulation wird in einer SQL Datenbank gespeichert. Nachdem die Pandas Instanzen ihre Berechnung beendet haben, teilt der Kopplungsworkflow die Eingabedaten für die Octave Simulation auf die verfügbaren Octave Instanzen auf und ruft für jede Instanz den systembiologischen Workflow auf. Der systembiologische Workflow übernimmt die Datenbereitstellung für einen Octave Web Service und ruft diesen dann für die Berechnung auf.. Zuletzt nutzt der Kopplungsworkflow die Ausgabedaten der systembiologischen Simulation, um die Eingabedaten der biomechanischen Simulation für den nächsten Zeitschritt anzupassen. Zuletzt werden die Ergebnisse der gekoppelten Simulation in einem letzten Schritt an einen Visualisierungsservice übergeben, der sie dann in nach gewünschten Optionen visualisiert. Weitere Details zu diesem Workflow lassen sich in [RSM14] finden.

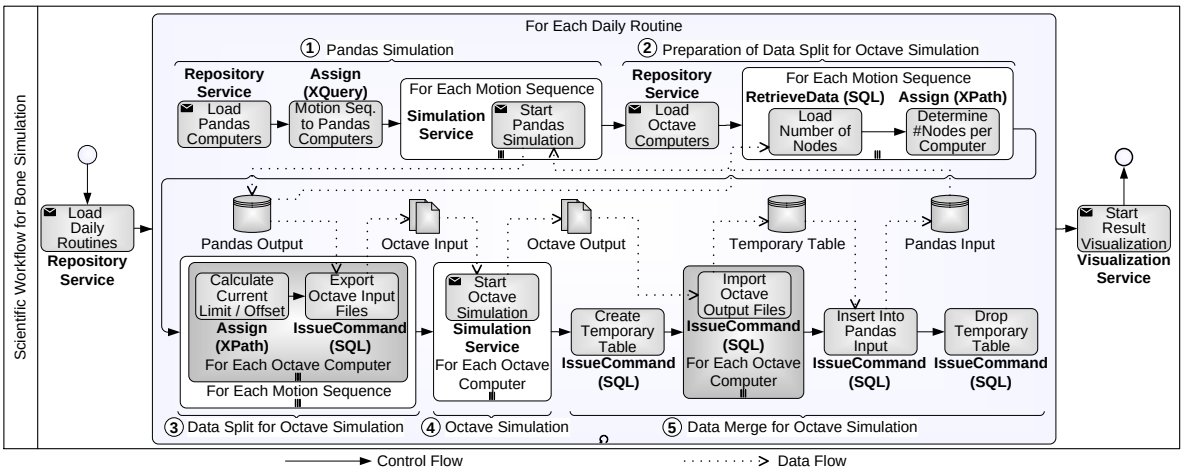


Abbildung 3.3: Aufbau des Kopplungsworkflow [RSM14]

Der Entwurf des biomechanischen und des systembiologischen Workflows werden in Kapitel 5 näher beschrieben.

4 Vorangegangene Arbeiten

Dieses Kapitel beschäftigt sich mit verschiedenen vorangegangenen Arbeiten und Entwicklungen, die für diese Arbeit relevant sind. Abschnitt 4.1 beschreibt das SIMPL Rahmenwerk, welches die Nutzung von Patterns ermöglicht. Abschnitt 4.2 beschreibt die Hierarchie der Patterns und den Ablauf der Transformation von Patterns auf ausführbare Workflowfragmente. Abschnitt 4.3 behandelt das allgemeine Datentransfer- und Transformationspattern, welches im Rahmen von [Pie12] bereits implementiert wurde. Abschnitt 4.4 beschreibt schließlich die verwendeten Webservice.

4.1 SIMPL Rahmenwerk

Der folgende Abschnitt beruht auf [RRS+11] und [RS14]. Das SIMPL-Rahmenwerk (SimTech – Information Management, Processes, and Languages) wurde in einer Zusammenarbeit des Instituts für Parallele und Verteilte Systeme (IPVS) und des Instituts für Architektur von Anwendungssystemen (IAAS) entwickelt. Das SIMPL Rahmenwerk dient dazu von einem Workflow aus auf externe Datenquellen mit unterschiedlichen Dateiformaten auf eine einheitliche Art zugreifen zu können.

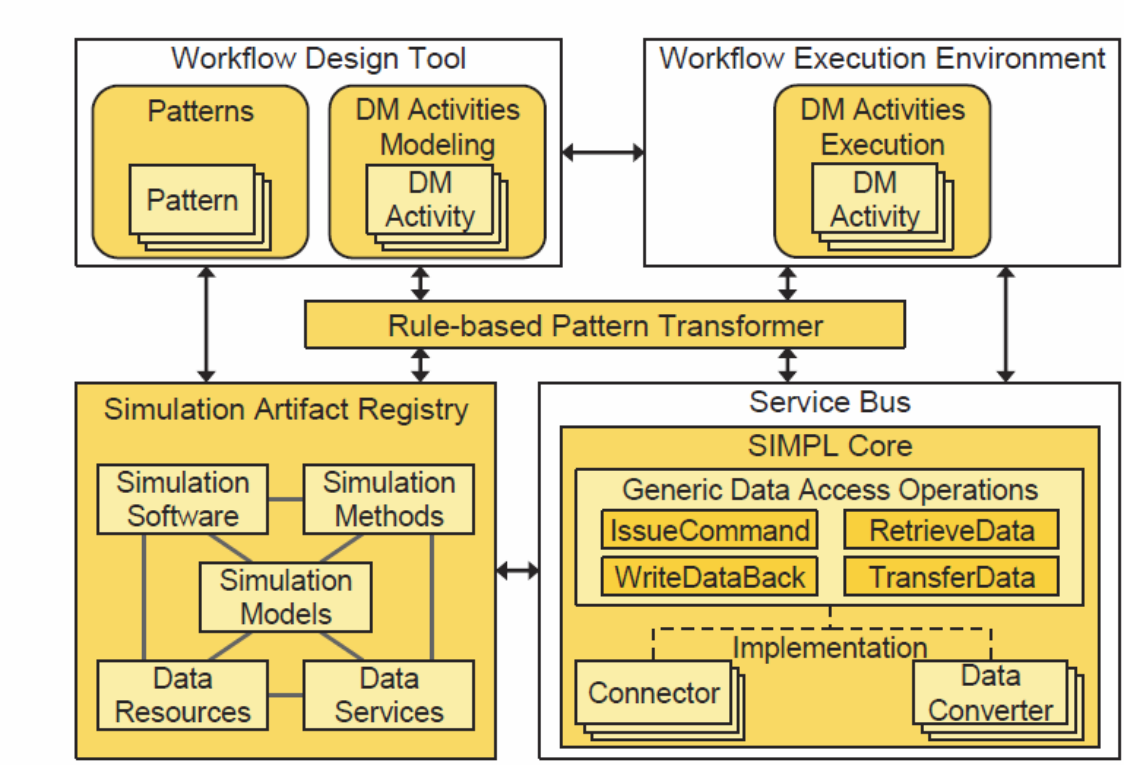


Abbildung 4.1: Aufbau des SIMPL Rahmenwerks eingebettet in ein SWfMS. Quelle: [RS14]

Abbildung 4.1 zeigt das SIMPL Rahmenwerk als Erweiterung eines Scientific Workflow Managementsystems (SWfMS). Der SIMPL Core bietet eine vereinheitlichte Schnittstelle, um auf unterschiedliche externe Datenquellen zugreifen zu können. Zusammen mit dem Ressource Management System bildet der SIMPL Core den Service Bus. Die Workflow Execution Environment wird durch das SIMPL Rahmenwerk um zusätzliche Data Management Activities erweitert. Um diese zusätzlichen Activities auch zur Modellierung von Workflows verwenden zu können, wird auch das Workflow Design Tool dahingehend erweitert. Für den Einsatz von Datenmanagementpatterns enthält das SIMPL Rahmenwerk eine regelbasierte Pattern Transformer Komponente. Diese Komponente nutzt ein Simulation Artifact Registry, um innerhalb des Workflow Design Tools verwendete Patterns in ausführbare Workflowfragmente umzuwandeln. Wie dieser Vorgang genau funktioniert wird im nächsten Abschnitt näher beschrieben.

4.2 Patternhierarchie und Patterntransformation

In [RS13] wurde eine Hierarchie für Datenmanagementpatterns beschrieben. Abbildung 4.2 zeigt diese Hierarchie. Auf der untersten Ebene dieser Hierarchie stehen ausführbare Workflowfragmente. Diese werden von Patterns der nächsthöheren Ebene - den grundlegenden Datenmanagementpatterns - genutzt um den Grad der Abstraktion zu erhöhen. Patterns erreichen dieses Ziel, indem sie

Implementierungsdetails der konkreten Workflowfragmente,oder von anderen Patterns, abstrahieren. Ein Beispiel für grundlegende Datenmanagementpatterns ist das allgemeine Datentransfer- und transformationspattern, auf das in Abschnitt 4.3 näher eingegangen wird. Dieses Pattern wurde bereits auf den Prototyp des biomechanischen Workflows angewendet [Pie12]. Die nächste Ebene der Hierarchie bilden die zusammengesetzten Datenmanagementpatterns. Ein Beispiel für Patterns dieser Ebene ist das simulationsorientierte Data Provisioning Pattern, welches in 5 beschrieben wird. Dieses Pattern wurde in dieser Arbeit auf die betrachteten Workflows angewandt. Auf der höchsten Ebene der Patternhierarchie stehen die simulationsorientierten Patterns. Ein Beispiel für Patterns dieser Stufe wird in Kapitel 5 in Form des Simulation Model Realization Patterns beschrieben. Auch dieses Pattern wurde in dieser Arbeit auf die betrachteten Workflows angewandt.

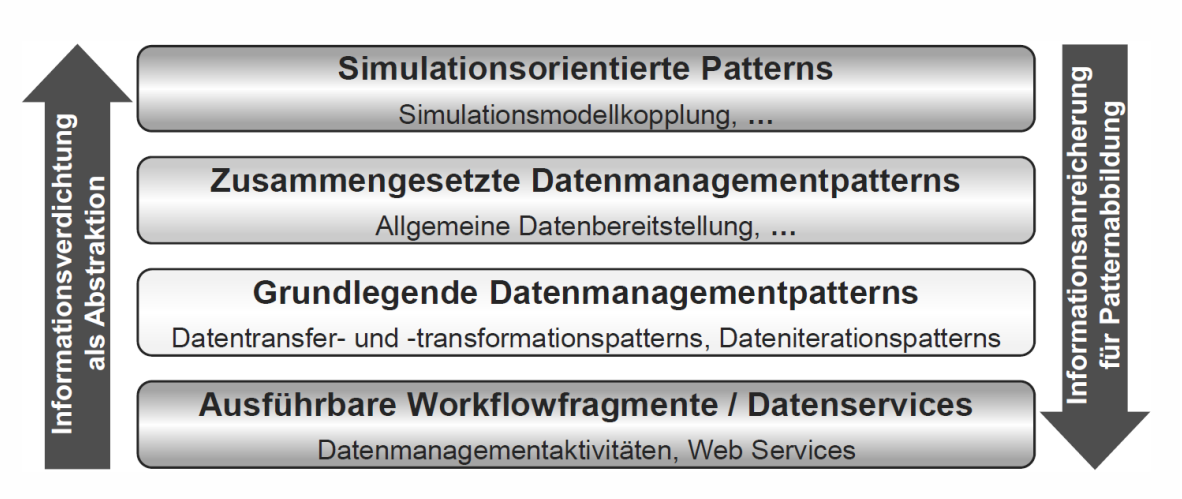


Abbildung 4.2: Hierarchie von Datenmanagementpatterns Quelle: [RS13]

Je höher die Ebene der Patternhierarchie, desto stärker werden Details der zugrundeliegenden Sprachen und Implementierungsdetails abstrahiert und desto stärker werden Informationen verdichtet. Als Folge davon benötigen Wissenschaftler immer weniger Detailwissen, um diese Patterns nutzen zu können.

Ein weiterer Vorteil ist, dass sich die Parameterwerte der Patterns mit zunehmender Hierarchieebene stärker dem zugrundeliegenden Simulationsmodell annähern. Als Beispiel sei hier das Simulation Model Realization Pattern genannt Als Parameter benötigt dieses Pattern nur ein Simulationsmodell, einen spezifischen Knochen und eine Bewegungssequenz (5). All das sind Parameter, mit denen sich ein Wissenschaftler genau auskennt, und alle weiteren Implementierungsdetails werden von dem Pattern abstrahiert.

Um die Patterns der höheren Hierarchieebene dann auch tatsächlich ausführen zu können müssen sie wieder mit Informationen angereichert werden. Zu diesem Zweck werden Patterns höherer Hierarchieebenen durch Transformationsregeln auf Patterns darunterliegender Ebenen abgebildet. Dabei werden die Transformationsregeln rekursiv angewendet, bis alle Patterns durch ausführbare Workflowfragmente, oder durch Service ersetzt wurden. Der folgende Abschnitt basiert auf [Rei11].

Abbildung 4.3 zeigt das Modell dieser Transformation von Datenmanagementpatterns. Auf der linken Seite der Abbildung wird ein Workflow mit mehreren Datenmanagementpatterns dargestellt. Um aus diesem Workflow einen ausführbaren Workflow zu machen, traversiert der Pattern Transformer durch den Workflow und sucht für jedes gefundene Pattern eine passende Transformationsregel. Transformationsregeln bestehen aus zwei Teilen: Ein Condition Part enthält Bedingungen, welche erfüllt sein müssen, damit die Transformationsregel anwendbar ist. In der Regel hängen diese Bedingungen von den Parameterwerten des jeweiligen Patterns, sowie von dazugehörigen Metadaten ab.

Ein Action Part enthält die notwendigen Schritte für die Abbildung des jeweiligen Patterns. Es existieren zwei Sorten von Abbildungsregeln: Bei Regeln vom Typ 1 werden Patterns direkt auf ausführbare Workflowfragmente abgebildet. Regeln vom Typ zwei bilden Patterns auf Worflowfragmente ab, die immer noch Datenmanagementpatterns enthalten. Aus diesem Grund müssen die Workflowfragmente der Regeln vom Typ 2 ebenfalls traversiert werden, um die Transformation in einen ausführbaren Workflow zu gewährleisten.

Zusätzlich zu den Regeln existieren noch Kontrollstrategien. Diese dienen dazu den Patterns geeignete Transformationsregeln zuzuordnen und die Reihenfolge, in der Transformationsregeln auf ihre Anwendbarkeit hin überprüft werden sollen, festzulegen.

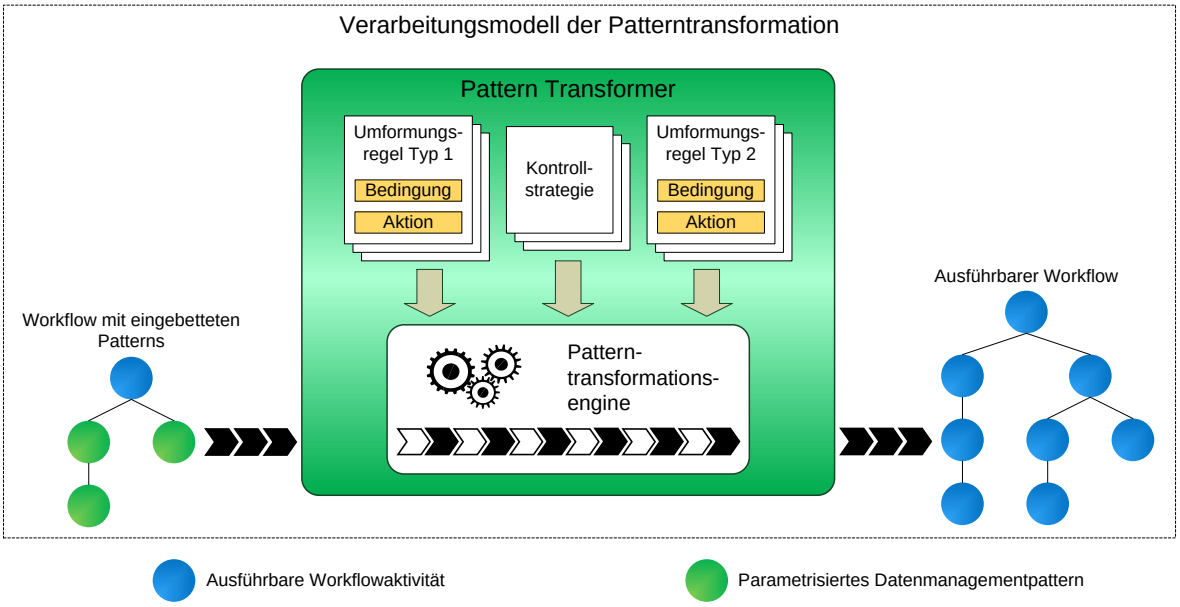


Abbildung 4.3: Modell der Transformation von Datenmanagementpatterns vgl. [Rei11]

4.3 Datentransfer- und transformationspattern

Das Datentransfer- und transformationspattern wurde in [RS13] zum ersten Mal beschrieben und im Rahmen von [Pie12] implementiert. Das Datentransfer- und -transformationspattern dient dazu Daten aus einer oder mehreren Datenquellen in eine oder mehrere Datensenken zu übertragen.

Abbildung 4.4 zeigt die allgemeine Form dieses Patterns. Daten werden aus Datencontainern (z.B. Dateien oder Datenbanktabellen) von einer oder mehreren Datenquellen (z.B. Datenbanksysteme oder Dateisysteme) extrahiert. Daten können unter anderem gefiltert und transformiert werden und werden dann in eine oder mehrere Datensenken kopiert. Damit entspricht dieses Pattern dem allgemeinen ETL-Prozess für die Extraktion, Transformation und das Laden von Daten.

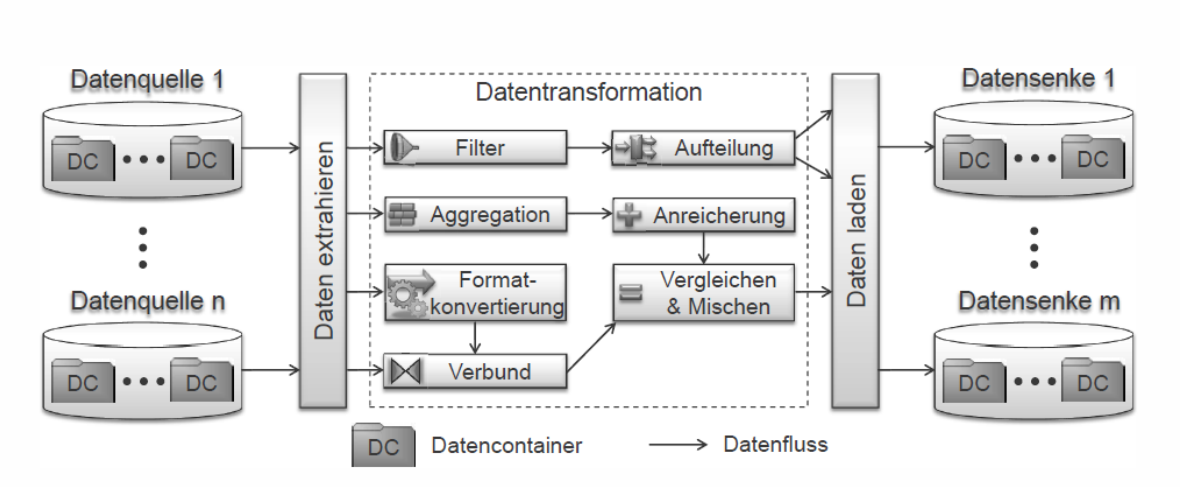


Abbildung 4.4: Das allgemeine Datentransfer- und -transformationspattern. Quelle: [RS13]

4.4 Webservices

In diesem Abschnitt werden die beiden Webservices beschrieben, die in den beiden Simulationsworkflows aufgerufen werden.

4.4.1 Pandas Webservice

Der in dieser Arbeit verwendete Pandas Webservice wurde ursprünglich von Raymond Dormien [Dor11] entwickelt und in Zusammenhang mit der Arbeit [KSR+11] von Robert Krause angepasst. Diese Anpassung umfasste auch einen Java Client, der die grundlegende Prozesslogik des biomechansichen Worflows beinhaltete. Außerdem wurde im Zuge von [Dor11] auch ein erster Prototyp des biomechansichen Workflows implementiert, der aber noch nicht für die Simulation von Strukturänderungen in Knochen genutzt wurde.

Im Rahmen dieser Studienarbeit wurde die Prozesslogik des obengenannten Java Clients mit dem von Raymond Dormien entwickelten Prototyp des biomechanischen Workflows verschmolzen und an die konkrete Simulation angepasst.

4.4.2 Octave Webservice

Für die systembiologische Simulation wird eine Octave Webservice verwendet. Dieser wurde ursprünglich im Rahmen der Arbeit [Zou12] erstellt. Der Webservice bietet die folgenden Methoden:

Create Directory : Erzeugt eine neues Verzeichnis auf dem Server.

Eingabeparameter :

DirectoryName String : Name des zu erzeugenden Verzeichnis.

Ausgabeparameter :

createDirectoryReturn String : Ergebnis der Methode.

copyFile : Kopiert eine Datei in das Directory der Zielinstanz.

Eingabeparameter :

IPAdress String : IP-Adresse des Zielsystems.

Username String : Benutzername für die Anmeldung auf dem Zielsystem.

Password String : Passwort für die Anmeldung auf dem Zielsystem.

DirectoryName String : Verzeichnis, in das die Datei kopiert werden soll.

Ausgabeparameter :

copyFileReturn String : Ergebnis der Methode.

PrepareSimulation : Erzeugt eine neue SimID und ein entsprechendes Verzeichnis

Eingabeparameter :

Start String : String für die Kommandozeilen Ausgabe.

Ausgabeparameter :

PrepareSimulationReturn String : Ergebnis der Methode.

StartProgram : Die angegebene Simulation wird ausgeführt.

Eingabeparameter :

DirectoryName String : Verzeichnis der auszuführenden Simulation.

ProgramName String : Auszuführendes Programm.

Ausgabeparameter :

StartProgramReturn String : Ergebnis der Methode.

removeFile : Die angegebene Datei wird entfernt

Eingabeparameter :

FileName String : Name der zu löschenden Datei.

Ausgabeparameter :

RemoveFileReturn String : Ergebnis der Methode.

setPlottingPath : Setzt den Pfad für die Plot Ausgabe der Simulation auf den angegebenen Pfad.

Eingabeparameter :

PlottingPath String : Zielverzeichnis.

Ausgabeparameter :

setPlottingPathReturn String : Ergebnis der Methode.

getPlottingPath : Liefert den Pfad für die Plot Ausgabe der Simulation.

Eingabeparameter :

FileName String : Name der Simulation deren Plot Pfad ausgegeben werden soll.

Ausgabeparameter :

getPlottingReturn String : Plot Pfad der gewählten Simulation.

5 Konzeptioneller Entwurf

In diesem Kapitel werden der konzeptionelle Entwurf der Workflows, die im Zuge dieser Arbeit verändert oder neu entworfen wurden, und der neu eingeführten Patterns beschrieben. Abschnitt 5.1 beschreibt die Änderungen an dem biomechanischen Workflow. In Abschnitt 5.2 wird der neue systembiologische Workflow beschrieben. Abschnitt 5.3 beschreibt letztlich die beiden im Rahmen dieser Arbeit implementierten Patterns.

5.1 biomechanischer Workflow

Für die Umsetzung dieses Workflows muss zuerst Prozesslogik aus dem von Robert Krause entwickelten Java Client für den Pandas Webservice in einen eigenen Webservice gekapselt werden. Innerhalb des Java Clients wurden für die Simulation zwei Pandas Instanzen parallel ausgeführt. Um den neuen Webservice möglichst generisch zu halten wurden die Methoden so gewählt, dass Pandas Instanzen innerhalb eines Workflows einzeln erzeugt werden können. Die entsprechende parallele Erzeugung zweier Instanzen wird dadurch in den Workflow verlagert.

Der neue Webservice hat folgende Methoden:

Plattform Provisioning : Erzeugt eine neue Simulationsinstanz

Ausgabeparameter :

Return SimIDFile : SimId und Verzeichnispfad der neu erzeugten Instanz.

Prepare Pandas Source : Kopiert Pandas Source Dateien in das Simulationsverzeichnis.

Eingabeparameter :

SimID SimIDFILE : SimID der Zielinstanz

Source String : Pfad an dem die Source Dateien gespeichert sind.

Ausgabeparameter :

Return String : SimID der veränderten Instanz.

Compile Pandas : Kompiliert die Pandas Instanz.

Eingabeparameter :

SimID SimIDFile : SimID der zu kompilierenden Instanz

Start Pandas : Die angegebene Pandas Instanz wird gestartet.

Eingabeparameter :

SimID SimIDFile : SimID der zu startenden Pandas Instanz

run cmd file : Führt die Cmd File der Pandas Instanzen aus.

Eingabeparameter :

SimID SimIDFILE : SimID der Zielinstanz.

Ausgabeparameter :

Stop Pandas : Beendet die Pandas Instanz.

Eingabeparameter :

SimID SImIDFile : SimID der Zielinstanz.

Im Rahmen dieser Studienarbeit wurde der bestehende biomechanische Workflow aus [Pie12] grundlegend verändert. Abbildung 5.1 zeigt den neuen Aufbau des biomechanischen Workflows.

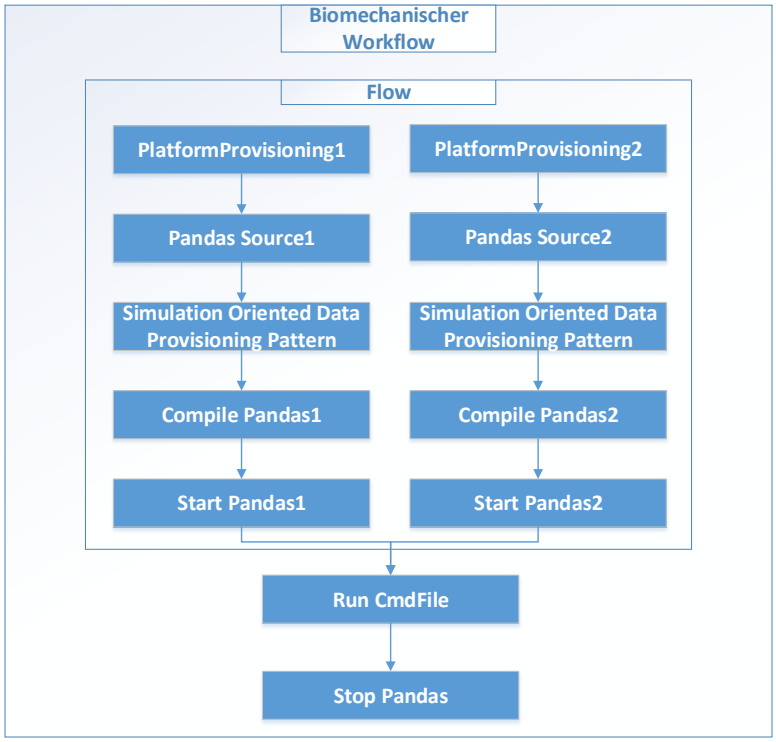


Abbildung 5.1: Aufbau des biomechanischen Workflows

Dieser Workflow wurde speziell dafür entworfen, von dem in Abbildung 3.3 gezeigten Kopplungsworkflow aufgerufen zu werden. Entsprechend entfallen einige Datenbereitstellungsschritte am Ende dieses Workflows, da diese zu dem in dieser Arbeit nicht betrachteten Kopplungsworkflow gehören.

Zu Beginn des Workflows werden parallel zwei neue Simulationsinstanzen erzeugt. Dafür wird die Webservice Methode PlatformProvisioning aufgerufen. Diese Methode liefert für jede Instanz eine SimID als Rückgabe. Nach dem PlatformProvisioning muss für jede Pandas Instanz der Pandas Sourcecode bereitgestellt werden. Dies erfolgt über den Methodenaufruf PreparePandasSource. Erst nach diesen Aufrufen kann die eigentliche Datenbereitstellung durch die Patterns erfolgen. Die für die Simulation notwendigen Eingabedaten werden für beide Simulationsinstanzen durch ein simulationsorientiertes Data Provisioning Pattern bereitgestellt(siehe 5.3). Nach der Datenbereitstellung erfolgt durch den Aufruf CompilePandas die Kompilierung des Pandas Sourcecodes. Anschließend werden beide Instanzen durch den Aufruf StartPandas gestartet. Die folgenden Methodenaufrufe erfolgen nun nicht mehr parallel. Über den Aufruf RunCmd werden die Commandfiles der beiden Simulationsinstanzen ausgeführt. Pandas speichert das Ergebnis der Simulation automatisch in die dafür bereitgestellte SQL Datenbank. Nach Abschluss der Simulation werden die Pandas Instanzen durch den Aufruf von Stop Pandas beendet und der biomechanische Workflow gibt eine Rückmeldung über den korrekten Ablauf an den aufrufenden Workflow.

5.2 systembiologischer Workflow

Auch dieser Workflow wurde speziell dafür entworfen, um von dem Kopplungsworkflow aufgerufen zu werden. Der systembiologische Workflow wurde im Rahmen dieser Studienarbeit komplett neu erstellt. Dieser Workflow verwendet den Octave Webservice der in [Zou12] beschrieben wird und in Abschnitt 4.4.2 der vorliegenden Arbeit zusammengefasst wird. Abbildung 5.2 zeigt den Aufbau des systembiologischen Workflows. Im ersten Webservice Aufruf, dem Prepare Simulation Schritt, erzeugt der Webservice eine neue Simulationsinstanz. Danach erfolgt durch das simulation oriented Data provisioning Pattern die Bereitstellung der Simulationsdaten. Der Aufruf von StartProgram startet die Octave Simulation. Nach Abschluss der Simulation werden die Ausgabedaten an den Kopplungsworkflow zurückgegeben.

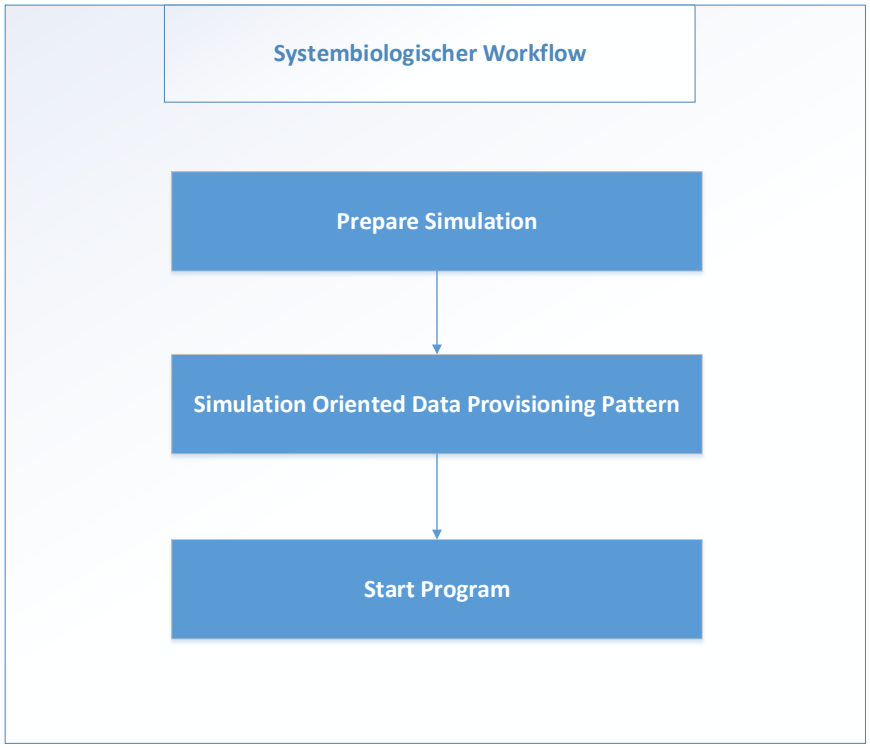


Abbildung 5.2: Aufbau des systembiologischen Workflows

5.3 Patterns

Dieser Abschnitt beschreibt die im Rahmen dieser Arbeit entworfenen Patterns für die Datenbereitstellung.

5.3.1 Simulationsorientiertes Data Provisioning Pattern

Das simulationsorientierte Data Provisioning Pattern steht innerhalb der Pattern Hierarchie (4.2) auf der Ebene der zusammengesetzten Datenmanagementpatterns. Bei der Patterntransformation wird es auf das Datentransfer- und Transformationspattern abgebildet, das bereits in [Pie12] umgesetzt wurde.

Als Eingabeparameter erhält dieses Pattern zum einen das Simulationsmodell, die von dem jeweiligen Modell benötigten mathematischen Variablen und die Ziel- Softwareinstanz, auf der die Simulation durchgeführt wird. Die Simulationsmodelle und ihre jeweiligen mathematischen Variablen werden in Form von Ontologien definiert. Bei der Transformation des simulationsorientierte Data Provisioning Patterns wird die ausgewählte Ontologie genutzt, um die mathematischen Variablen auf konkrete

Referenzen auf Datencontainer umzuwandeln. Die im Pattern angegebene Ziel- Softwareinstanz wird bei der Transformation verwendet, um das Zielverzeichnis, in welches die Eingabedaten bereitgestellt werden sollen, zu finden.

Um die benötigten mathematischen Variablen innerhalb der Entwicklungsumgebung spezifizieren zu können, wurden die beiden Ontologien BoneCell und BoneTissue erweitert. Den in der Ontologie definierten mathematischen Variablen wurden nun konkrete Instanzen zugeordnet, welche die eigentlichen Eingabedateien auf abstrakte Weise repräsentieren. Diese Instanzen haben jeweils eine Property mit dem jeweiligen konkreten Dateipfad der definierten Eingabedatei.

Zuletzt muss noch der Datencontainer der Zielinstanz ausgewählt werden. Eigentlich sollte die Zielinstanz aus dem Ressource Artifact Registry abgerufen werden 4.1. Allerdings werden die Simulationsinstanzen erst zur Laufzeit erzeugt und zudem noch nicht in der Simulation Artifact Registry automatisch registriert. Daher wäre für diese Form der Implementierung eine Patterntransformation zur Laufzeit nötig. Für den Moment wird die entsprechende Datencontainer Variable daher manuell im Workflow deklariert und bei der Patterntransformation diese Variable ausgewählt.

Für die Transformation des simulationsorientierten Data Provisioning Patterns musste zunächst ein entsprechendes Workflow Fragment für die spätere Transformation implementiert werden. Listing 5.1 zeigt dieses Workflow Fragment. Das Fragment besteht aus einem Scope, in dem mehrere Container Referenz Variablen (<bpel:containerReferenceVariables>) deklariert werden und der ein einzelnes Data Transfer und Transformationspattern enthält. Bei der Pattern Transformation dieses Patterns müssen Container Referenz Variablen für als Parameter des Patterns ausgewählten mathematischen Variablen deklariert werden, daher ist der Scope nötig. Der Token ?variables ist ein Platzhalter, der während der Transformation durch die Variablendeklarationen ersetzt wird. Ebenso wird bei der Transformation ?containerReferences durch eine Liste der zuvor deklarierten Referenzvariablen ersetzt. ?targetContainer wird durch den als Parameter des Patterns gewählte Datencontainer der Ziel- Softwareinstanz ersetzt.

```
<bpel:scope name="scope?randomNumber">
 <bpel:containerReferenceVariables>
   ?variables
 </bpel:containerReferenceVariables>
 <bpel:extensionActivity>
         <simpl:dataTransferAndTransformationPattern containerReferences="?containerReferences"
              targetContainer="?targetContainer"
              name="DataTransferAndTransformationPattern"></simpl:dataTransferAndTransformationPattern>
 </bpel:extensionActivity>
 </bpel:scope>
```

Listing 5.1: Workflow Fragment des simulationsorientierten Data Provisioning Patterns

Als nächsten Schritt wurde das Eclipse Plugin für die Pattern Transformation um die zusätzliche Transformationsregel SimulationOrientedDataProvisioning.java erweitert. Der Condition Part der Transformationsregel sieht so aus:

Bedingung1 : Der Parameter mathematical Variables darf nicht null sein.

Bedingung2 : Der Parameter targetContainer entspricht einer Data cContainer Reference Variable bzw. deren Namen.

Sind diese Bedingungen erfüllt kann der Action Part der Transformationsregel ausgeführt werden:

Schritt 1 : Das Workflow Fragment(siehe Listing 5.1) wird geladen.

Schritt 2 : Für jede mathematische Variable wird aus der Ontologie der entsprechende Dateipfad geladen. Um dies zu ermöglichen, wurde die Java Klasse Ontologyutils des Models Plugins um eine entsprechende Methode erweitert. Anschließend wird der Platzhalter ?variable durch die entsprechenden Container Referenzen der mathematischen Variablen ersetzt.

Schritt 3 : Der Platzhalter ?containerReferences wird durch die Namen der in Schritt 2 erzeugten Container Referenzen ersetzt.

Schritt 4 : Der Platzhalter ?targetContainer wird durch den Parameter Zielinstanz ersetzt.

Schritt 5 : Der Platzhalter ?randomNumber wird durch eine Zufallszahl ersetzt, um dem Scope einen eindeutigen Namen zu geben.

Für die anschließende Transformation des Datentransfer- und Transformationspatterns konnte die Implementierung von Henrik Pietranek aus [Pie12] ohne Änderungen genutzt werden.

5.3.2 Simulationmodel Realization Pattern

Das Simulationmodel Realization Pattern gehört zu der höchsten Ebene der Patternhierarchie.

Dieses Pattern hat für den betrachteten biomechanischen Anwendungsfall einer Simulation zu Strukturänderungen in Knochen folgende Eingabeparameter:

Simulationsmodell : Das Simulationsmodell, welches durch dieses Pattern realisiert werden soll.

Knochen : Ein Knochen, der simuliert werden soll.

Bewegungssequenz : Die Bewegungssequenz, unter der der Knochen simuliert werden soll.

Bei der Transformation wird dieses Pattern auf ein Workflow Fragment der in 5.1 und 5.2 beschriebenen Simulationsworkflows abgebildet. Um zusätzliche Simulationsmodelle zu unterstützen, müssen entsprechende Workflowfragmente implementiert werden. Der ausgewählte Knochen und die Bewegungssequenz werden genutzt, um die Eingabevariablen der jeweiligen Simulation auszuwählen.

Für die Implementierung des Simulation Model Realization Patterns dienen die beiden Simulationsworkflows als Workflowfragmente. Dabei wurden die Parameterwerte des jeweiligen simulationsorientierten Data Provisioning Patterns in diesen Simulationsworkflows durch Platzhalter ersetzt.

Um aus den beiden Parametern Bone und Motion Sequence auf die entsprechenden Parameter der Datenbereitstellung abbilden zu können, mussten die beiden Ontologien erneut erweitert werden. Jede Instanz eines Knochens hat jetzt Properties, welche dem Knochen Instanzen seiner mathematischen Variablen zuordnen. Außerdem wurden den Instanzen der mathematischen Variablen für die Randbedingungen jeweils eine Bewegungssequenz zugeordnet. Durch diese Änderungen ist es

möglich, einer Bewegungssequenz und einem Knochen, in einem Simulation-oriented Data Provisioning Pattern, einen konkreten Satz an Eingabedateien, in Form der Instanzen von mathematischen Variablen, zuzuordnen.

Die Transformationsregeln wurden um die Regel SimulationModelRealization.java erweitert. Diese Regel hat folgende Vorbedingungen:

Bedingung1 : Der Parameter Bone darf nicht null sein.

Bedingung2 : Der Parameter Motion Sequence darf nicht null sein.

Sind diese Bedingungen erfüllt, kann der Action Part der Transformationsregel ausgeführt werden:

Schritt 1 : Abhängig von dem ausgewählten Simulationsmodell wird der entsprechende Workflow geladen. Wird das BoneTissue Modell gewählt, so wird der biomechanische Workflow geladen. Wird das BoneCell Simulationsmodell ausgewählt, wird der systembiologische Workflow geladen.

Schritt 2 : Anhand der ausgewählten Bewegungssequenz und des gewählten Knochens werden die Platzhalter für die konkreten Datencontainer der Eingabeparameter innerhalb des Workflow Fragments ersetzt.

6 Implementierung

Dieses Kapitel beschäftigt sich mit der konkreten Implementierung der im vorigen Kapitel beschriebenen Workflows und Patterns. Abschnitt 6.1.1 beschreibt die Implementierung des biomechanischen Workflows. Abschnitt 6.1.2 die des systembiologischen Workflows. Zuletzt beschreibt Abschnitt 6.2.1 die Implementierung des simulationsorientierten Data Provisioning Patterns und des Simulation Model Realization Patterns.

6.1 Workflows

Die Umsetzung der beiden Workflows erfolgte in der Worflowsprache BPEL [Jor07]. Für die Implementieurng wurde Eclipse [1] und das Eclipse BPEL Projekt [2] verwendet.

6.1.1 biomechanischer Workflow

Bei der Implementierung des biomechanischen Workflows mussten, wie bereits erwähnt, aufgrund des neuen Pandas Webservices einige Anpassungen vorgenommen werden. Konkret wurden Teile des von Robert Krause entworfene Java Client des Pandas Webservice unter Verwendung von Eclipse und dem Apache Axis2 Service Archive Generator in einen Webservice umgewandelt. Dafür wurde der Code des Clients in einzelne Methoden des neuen Webservice aufgeteilt.

6.1.2 systembiologischer Workflow

Der systembiologische Workflow wurde von Grund auf neu implementiert. Als in diesem Workflow eingebunden Web Service Webservice wird der GNU Octave Webservice von [Zou12] verwendet.

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

6.2 Patterns

6.2.1 Simulationsorientiertes Data Provisioning Pattern

Für die Implementierung des simulationsorientierten Data Provisioning Patterns mussten die Eclipse Plugins des SIMPL Rahmenwerks entsprechend angepasst werden. Eine neue Transformationsregel wurde in das SIMPL UI Plugin innerhalb des Pakets org.eclipse.bpel.simpl.ui.pattern.transformation.transformation.rules eingefügt. Desweiteren wurde eine neue Kontrollstrategie speziell für dieses Pattern in das Paket org.eclipse.bpel.simpl.ui.pattern.transformation.controll.strategies eingefügt. Die Kontrollstrategie umfasst die eine zuvor definierte Transformationsregel.

Um die Benutzung des Patterns innerhalb der Entwicklungsumgebung zu gewährleisten wurde der Property Bereich des Pattern verändert. Abbildung 6.1 zeigt den Property Bereich des simulationsorientierten Data Provisioning Patterns innerhalb der Bpel-Eclipse Umgebung. Wie bereits zuvor beschrieben (5.3) benötigt das Pattern als Eingabeparameter ein Simulationsmodell, mehrere mathematische Variablen und eine Zielinstanz. Über den Button Browse kann aus einer Liste von Ontologien das gewünschte Modell ausgewählt werden.

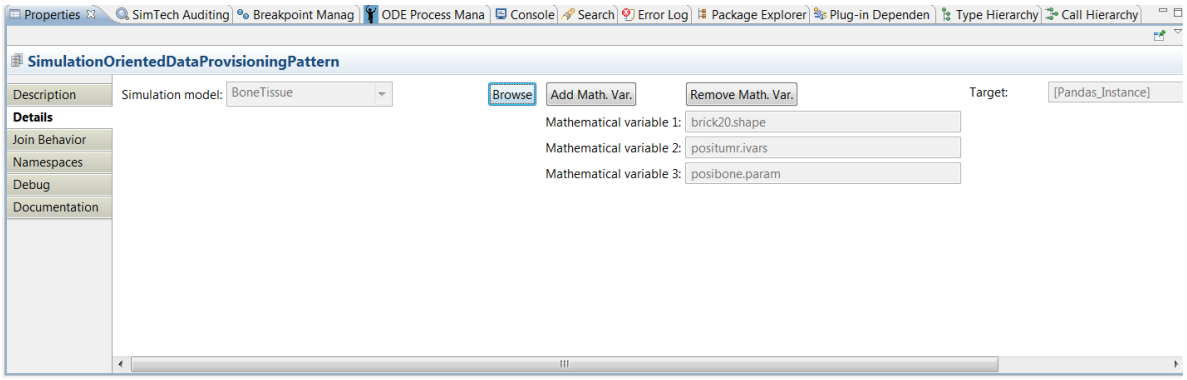


Abbildung 6.1: Property Bereich des simulationsorientierten Data Provisioning Patterns

Abbildung 6.2 zeigt das entsprechend angepasste Popup zur Auswahl der mathematischen Variablen. Dieses Popup wurde dahingehend verändert, dass nun zu der ausgewählten mathematischen Variable auf der rechten Seite eine dazugehörige konkrete Instanz dieser Variable ausgewählt werden kann.

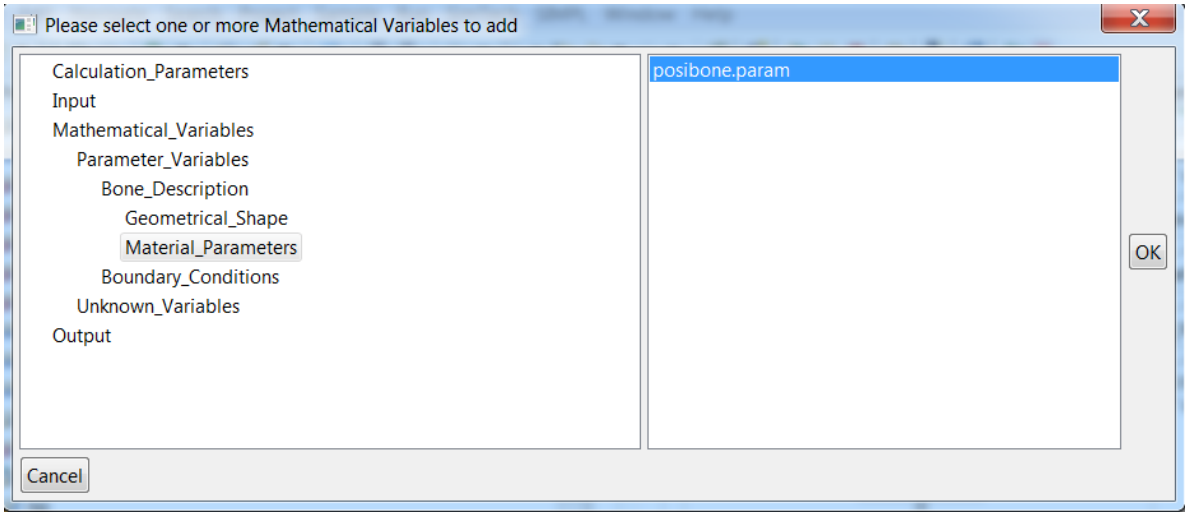


Abbildung 6.2: Popup zur Auswahl der mathematischen Variablen

Um die Transformation zu vervollständigen wurde das im vorigen Kapitel erwähnte Workflowfragment in das Fragmento Repository [3], aus dem alle Workflowfragmente geladen werden, eingefügt.

6.2.2 Simulation Model Realization

Wie bereits im vorigen Kapitel beschrieben mussten für die Implementierung dieses Patterns die beiden Ontologien BoneTissue und BoneCell angepasst werden. Instanzen der mathematischen Variablen wurden über eine objectproperty mit einem konkreten Bewegungsmuster verknüpft. Auf diese Weise kann bei der Patterntransformation aus den Parameterwerten für den Knochen und das Bewegungsmuster die entsprechenden mathematischen Variablen aus der Ontologie entnommen werden.

Wie bei dem simulationsorientierten Data Provisioning Pattern wurde eine Tranformationsregel und eine Kontrollstrategie implementiert.

[3] http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm

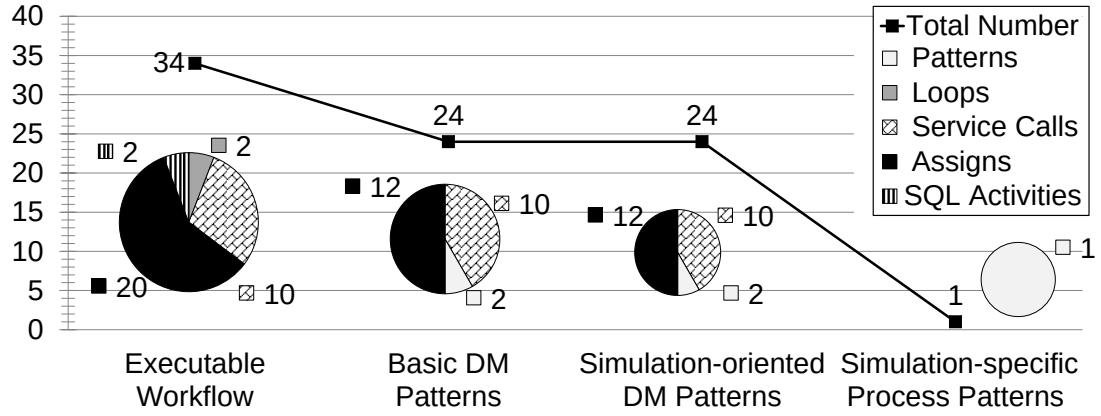
7 Bewertung

Dieses Kapitel beschäftigt sich mit der Bewertung der neu implementierten Patterns. Als Kriterien für diese Bewertung soll zum einen die Abstraktionsunterstützung des jeweiligen Patterns dienen, zum anderen inwiefern sich das jeweilige Pattern generisch für unterschiedliche Eingabeparameter und Simulationsmodelle einsetzen lässt. In Abschnitt 7.1 wird die Abstraktionsunterstützung der Patterns bewertet. Abschnitt 7.2 beschäftigt sich mit der generischen Einsetzbarkeit der Patterns.

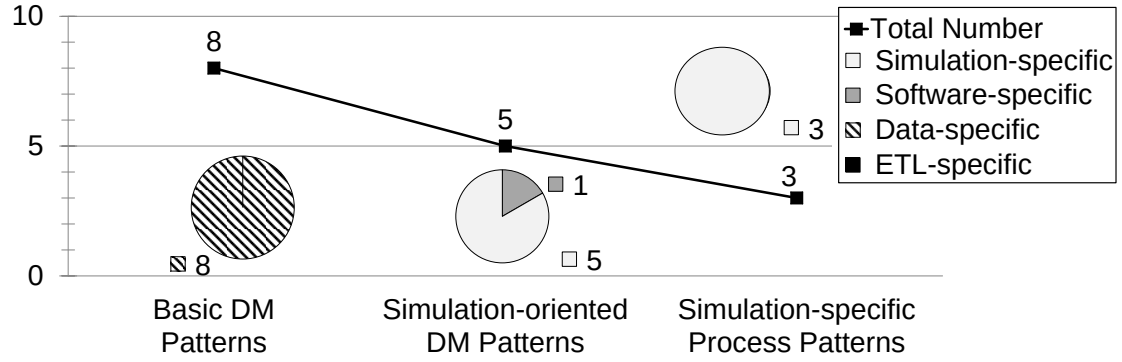
7.1 Abstraktion

Eines der Ziele beim Einsatz von Patterns ist es den Entwurf von Simulationen zu vereinfachen, indem Implementierungsdetails nicht explizit von Wissenschaftlern angegeben werden müssen. Im folgenden soll bewertet werden inwiefern die neu implementierten Patterns diese Abstraktion unterstützen. Dabei wird vor allem auf die neu implementierten Datenmanagementpatterns eingegangen.

Abbildung 7.1 zeigt eine graphische Representation der Anzahl der Workflowschritte (a)) und der Anzahl der Parameter (b)), die auf jeder Abstraktionsebene der Patternhierarchie (4.2) angegeben werden müssen. Als Basis zur Erstellung dieses Schaubilds wurde dabei der neu implementierte biomechanische Workflow genutzt.



a) Anzahl der Workflowschritte



b) Anzahl der Parameter der Patterns

Abbildung 7.1: Anzahl der Workflowschritte und Parameter, die für den biomechanischen Workflow auf den verschiedenen Abstraktionsebenen spezifiziert werden müssen.

Wie in Teilabbildung a) zu sehen ist, führt das neu eingeführte simulationsorientierte Data Provisioning Pattern nicht zu einer Reduktion der Anzahl an Workflowschritten. Den eigentlichen Nutzen

dieses Patterns zeigt Teilabbildung b): Während die allgemeinen Datenmanagementpatterns 8 datenspezifische Parameter benötigen, benötigt das simulationsorientierte Data Provioning Pattern 5 simulationsspezifische Parameter(das Simulationsmodell und die mathematischen Variablen) und einen software-spezifischen Parameter(die Zielinstanz). Das Pattern abstrahiert also erfolgreich alle datenspezifischen Details der Datenbereitstellung und reduziert zudem auch die Gesamtanzahl zu spezifizierenden Parameterwerte für Patterns. Dies bringt einen großen Vorteil bei der Verwendung dieses Patterns, da simulations- spezifische Parameter naturgemäß von Wissenschaftlern viel einfacher zu Verstehen sind als datenspezifische Parameter.

Das Simulation Model Realization Pattern wiederum reduziert drastisch die Anzahl der Workflowschritte (Teilabbildung a)). Das Pattern abstrahiert alle Details des eigentlichen Simulationsworkflows und benötigt gleichzeitig nur 3 simulationsspezifische Parameterwerte. Genauer handelt es sich bei diesen Parameterwerten um ein Simulationsmodell, einen zu simulierenden Knochen, sowie ein Bewegungsmuster.

7.2 Generische Einsetzbarkeit

Um eine hohe Wiederverwendbarkeit zu ermöglichen, sollten die Patterns möglichst generisch einsetzbar sein. Im folgenden soll bewertet werden, inwiefern dies auf die in dieser Arbeit umgesetzten Patterns zutrifft.

7.2.1 Simulationsorientiertes Data Provisioning Pattern

Das simulationsorientierte Data Provisioning Pattern liest die Hierarchie der mathematischen Variablen aus einer Ontologie des jeweiligen Simulationsmodells aus. Es kann dabei mit beliebigen Hierarchien in Ontologien umgehen und ist daher generisch für unterschiedliche Simulationsmodelle einsetzbar.

7.2.2 Simulation Model Realization Pattern

Bei der Transformation des Simulation Model Realization Patterns wird das Pattern auf ein Workflowfragment des jeweiligen Simulationsworkflows abgebildet. Aus diesem Grund kann das Simulation Model Realization Pattern nur für solche Simulationsmodelle eingesetzt werden für die vorher bereits ein Workflowfragment implementiert wurde. Derzeit ist es nur für die biomechansiche und systembiologische Simulation einsetzbar. Die eigentliche Abbildung der Parameterwerte auf das systemorientierte Data Provisioning Pattern ist jedoch generisch.

8 Zusammenfassung

Der Einsatz von Workflows nimmt im wissenschaftlichen Bereich stetig zu. Patterns sind eine Möglichkeit Wissenschaftlern das Erstellen von Scientific Workflows zu erleichtern. In dieser Arbeit wurden anhand der Simulation von Strukturänderungen in Knochen entsprechende Datenmanagement Patterns implementiert. Um dies zu ermöglichen wurde der bestehende Workflow der biomechanischen Simulation von Strukturänderungen in Knochen entsprechend angepasst. Zudem wurde ein neuer Workflow für die systembiologische Simulation implementiert.

Für die Datenbereitstellung dieser beiden Workflows wurde das simulationsorientierte Data Provisioning Pattern entworfen. Dieses Pattern abstrahiert datenspezifische Einzelheiten der Datenbereitstellung und ist generisch für beliebige Simulationsmodelle und Eingabedaten einsetzbar.

Außerdem wurde das Simulation Model Realization Pattern implementiert. Dieses Pattern der höchsten Abstraktionsstufe abstrahiert alle Einzelheiten des eigentlichen Workflowaufbaus und benötigt nur simulationsspezifische Parameter. Implementiert wurde dieses Pattern speziell für den systembiologischen und den biomechanischen Workflow.

In Kapitel 5 wurde der Entwurf dieser Patterns und der beiden Workflows näher beschrieben und in Kapitel 6 wurde die eigentliche Implementierung dargelegt.

Kapitel 7 hat die neu implementierten Patterns auf ihre Abstraktionsunterstützung und die generische Einsetzbarkeit hin bewertet. Dabei hat sich gezeigt, dass das simulationsorientierte Data Provisioning Pattern erfolgreich alle Details der eigentlichen Datenbereitstellung abstrahiert, und gleichzeitig nur simulationsspezifische Parameter benötigt. Dieses Pattern ist zudem durch seine Verwendung von Ontologien generisch für beliebige Simulationsmodelle einsetzbar. Das Simulation Model Realization Model wiederum verringert auf drastische Weise die Anzahl an Workflowschritten und abstrahiert dadurch alle Details des eigentlichen Simulationsworkflows. Dieses Pattern ist sowohl für den biomechanische, als auch für den systembiologischen Workflow einsetzbar.

Zusammen abstrahieren diese beiden Patterns erfolgreich alle Details der Datenbereitstellung und benötigen dabei nur simulationsspezifische Parameter. Ein Wissenschaftler, der diese Patterns verwendet, muss also lediglich solche Parameter angeben, mit denen er sich bereits auskennt.

Literaturverzeichnis

[BPSM+] T. Bray, J. Paoli, C. Sperberg-Mcqueen, E. Maler, F. Yergeau. Extensible Markup Language (XML) 1.0 W3C Recommendation. 2008 http://www. w3. org/TR/2008. *REC-xml-20081126*. (Zitiert auf Seite 7)

[Dor11] R. Dormien. *Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen*. Diplomarbeit, Universität Stuttgart, 2011. (Zitiert auf den Seiten 5 und 25)

[GSK+11] K. Görlach, M. Sonntag, D. Karastoyanova, F. Leymann, M. Reiter. Conventional workflow technology for scientific simulation. In *Guide to e-Science*, S. 323–352. Springer, 2011. (Zitiert auf den Seiten 5 und 12)

[Jor07] J. Jordan, D.; Evdemon. Web Services Business Process Execution Language Version 2.0, 2007. URL http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf. (Zitiert auf den Seiten 12 und 37)

[KE11] A. Kemper, A. Eickler. *Datenbanksysteme: Eine Einführung*. Oldenbourg Verlag, 2011. (Zitiert auf Seite 13)

[KSR+11] R. Krause, D. Schittler, M. Reiter, S. Waldherr, F. Allgöwer, D. Karastoyanova, F. Leymann, B. Markert, W. Ehlers. Bone remodelling: A combined biomechanical and systems-biological challenge. *PAMM*, 11(1):99–100, 2011. (Zitiert auf den Seiten 5, 17 und 25)

[LR00] F. Leymann, D. Roller. Production Workflow: Concepts and Techniques. 2000. (Zitiert auf Seite 12)

[Mel10] I. Melzer. *Service-orientierte Architekturen mit Web Services: Konzepte-Standards-Praxis*. Springer DE, 2010. (Zitiert auf den Seiten 4, 9 und 10)

[oct] Webseite Octave. URL http://www.gnu.org/software/octave/. (Zitiert auf den Seiten 5 und 17)

[pan] Webseite des Pandas Rahmenwerks. URL http://www.mechbau.uni-stuttgart.de/pandas/index.html. (Zitiert auf den Seiten 5 und 17)

[Pie12] H. A. Pietranek. *Datenmanagementpatterns in multi-skalaren Simulationsworkflows*. Diplomarbeit, Universität Stuttgart, 2012. (Zitiert auf den Seiten 5, 21, 23, 24, 30, 32 und 34)

[Rei11] P. Reimann. SimTech Milestone Report: Data Provisioning for Scientific Workflows. Technischer Bericht, Universität Stuttgart, 2011. (Zitiert auf den Seiten 4, 23 und 24)

[RRS+11] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, F. Leymann. SIMPL-A Framework for Accessing External Data in Simulation Workflows. In *BTW*, S. 534–553. 2011. (Zitiert auf Seite 21)

[RS13] P. Reimann, H. Schwarz. Datenmanagementpatterns in Simulationsworkflows. In *BTW*, S. 279–293. 2013. (Zitiert auf den Seiten 4, 6, 22, 23, 24 und 25)

[RS14] P. Reimann, H. Schwarz. Simulation Workflow Design Tailor-Made for Scientists. *International Conference on Scientific and Statistical Database Management*, 2014. (Zitiert auf den Seiten 4, 21 und 22)

[RSM13] P. Reimann, H. Schwarz, B. Mitschang. A Data Pattern Approach to Flexible Interoperability in Dynamic Simulation Environments. In *Unveröffentlichter interner Bericht des Instituts für Parallele und Verteilte Systeme der Universität Stuttgart*. 2013. (Zitiert auf den Seiten 4 und 19)

[RSM14] P. Reimann, H. Schwarz, B. 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*. 2014. (Zitiert auf den Seiten 4, 5, 19 und 20)

[RWWS14] P. Reimann, T. Waizenegger, M. Wieland, H. Schwarz. Datenmanagement in der Cloud für den Bereich Simulationen und Wissenschaftliches Rechnen. In *Tagungsband des 2. Workshops Data Management in the Cloud, 44. Jahrestagung der Gesellschaft für Informatik (GI) (ToAppear)*. Stuttgart, Deutschland, 2014. (Zitiert auf den Seiten 4 und 18)

[Sch03] H. Schöning. *XML und Datenbanken*, Band 1. Hanser, 2003. (Zitiert auf Seite 7)

[Seb10] T. J. Sebestyen. *XML: Einstieg für Anspruchsvolle*. Pearson Deutschland GmbH, 2010. (Zitiert auf Seite 7)

[Stu09] H. Stuckenschmidt. *Ontologien: Konzepte, Technologien und Anwendungen*. Springer-Verlag, 2009. (Zitiert auf Seite 14)

[TDGS07] I. J. Taylor, E. Deelman, D. Gannon, M. Shields. *Workflows for e-Science*. Springer-Verlag London Limited, 2007. (Zitiert auf Seite 5)

[Zou12] Y. Zou. *Simulation des Verhaltens von Zellkomponenten in biologischen Netzwerken mit Hilfe von Workflow Technologie*. Diplomarbeit, Universität Stuttgart, 2012. (Zitiert auf den Seiten 26, 31 und 37)

Alle URLs wurden zuletzt am 17. 06. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift