Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3127

Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen

Raymond Dormien

Studiengang:	Informatik
Prüfer:	Jun.-Prof.Dr. Dimka Karastoyanova
Betreuer:	Dipl.-Math. Michael Reiter
begonnen am:	23.12.2010
beendet am:	24.06.2011
CR-Klassifikation:	C.2.4, D.2.12, H.2.8, H.3.4, H.3.5, I.6.3, I.6.7, I.6.8, J.2

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

BPEL ……Business Process Execution Language

CORBA ……Common Object Request Broker Architectur

DCOM ……Distributed Component Object Model

DGL ……Differentialgleichung

EPR ……Endpoint Reference

OVF ……Open Virtualization Format

pDGL ……partielle Differentialgleichung

SOA ……Service-orientierte Architektur

W3C ……World Wide Web Consortium

WfMC ……Workflow Management Coalition

WfMS ……Workflow Management System

WS ……Web Service

WSDL ……Web Service Description Language

1 Einleitung

Web Service- und Workflow-Technologien haben sich im Business-Bereich etabliert und bekommen in den letzten Jahren immer mehr Aufmerksamkeit im wissenschaftlichen Bereich. Entsprechend wurde der Begriff Scientific Workflows geprägt. Durch den Einsatz von Workflow-Technologien soll es Wissenschaftlern ermöglicht werden, sich verstärkt auf die Lösung ihrer wissenschaftlichen Probleme zu konzentrieren, da Schritte während des Workflow-Designs und der Ausführung automatisiert werden. Ebenso wird die Zusammenarbeit von Wissenschaftlern gefördert, da eine Wissensweitergabe über gemeinsame Dienste möglich ist und gemeinsam eine Ergebnisanalyse geführt werden kann. Auch durch die Fähigkeit der Workflow-Technologie mit großen Datenmengen umgehen zu können, eignen diese sich für die Ausführung von Simulationen. Solche Simulation-Workflows bilden einen Teilbereich der Scientific Workflows. [1]

Die Arbeit in der Wissenschaft wird immer stärker durch Simulationen und der Analyse der Ergebnisdaten geprägt. Der Einsatz von Simulationen kann vielfältige Gründe haben. Mit Hilfe von Simulationen können zum Beispiel Wissenschaftler, detailliert die Dynamik eines echten Prozesses untersuchen. In vielen Fällen können diese Daten experimentell nicht erfasst werden, da zum Beispiel die betrachteten Zeitskalen zu groß sind (z.B. die Evolution von Galaxien) oder das Experiment aus theoretischen Gründen nicht ausführbar ist (z.B. Kontrafaktische Modelle, bei dem fundamentale Konstanten aus der Natur geändert werden). [2]

Aktuelle Simulationsprogramme sind hochspezialisiert und oft für einen bestimmten Zweck entwickelt. Diese hohe Spezialisierung und Zweckgebundenheit stellt jedoch häufig ein Hindernis für eine Kopplung verschiedener Simulationsprogramme dar. Durch die Kopplung verschiedener Simulationsprogramme werden Multi-Skalen, Multi-Physiken, Multi-Domänen und Multi-Tools Simulationen möglich und lassen eine kombinierte Aussage der Einzelsimulationen zu. Sie erlauben komplexe Fragestellungen wie etwa: Wie verhält sich zum Beispiel auf zellularer Ebene der Auf- und Abbau eines menschlichen Knochens bei einer bestimmten Belastung, und wie verändert sich dadurch die gesamte Knochenstruktur?

Eine Aufgabe des Stuttgarter Exzellenzclusters Simulation Technology (SimTech)[1] ist die interdisziplinäre Zusammenarbeit, um Simulation-Workflows voranzutreiben indem bestehende wie zukünftige Simulationsprogramme durch Workflows integriert werden.

1.1 Aufgabenstellung und Motivation

Im Rahmen dieser Arbeit werden Schnittstellen zum parallelisierten und automatisierten Ablauf von Simulationen aus dem Bereich der Finiten Elemente ausgearbeitet. Die Hauptaufgabe liegt im Entwurf einer geeigneten Architektur, um Pandas[2]-basierte Simulationen in Workflows zu nutzen.

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

Eine Service Bus Erweiterung für Pandas soll hier insgesamt fünf Anforderungen genügen. Hierunter fallen die Bereitstellung von Pandas als Web Service, die Speicherung von Ergebnissen in einer Datenbank, die Interaktion mit anderen Simulationsprogrammen, die Verwendung einer Opensource Datenbank, und die Erstellung eines BPEL-Prozess.

Die Pandas Service-Bus-Erweiterung soll darüber hinaus um die Möglichkeit einen DUNE[3] Matrixlöser zu verwenden, die Möglichkeit die Datenqualität der Matrix während der Simulation zu erfassen und eine Pandas-Matlab Kopplung durchführen zu können erweitert werden. Hierzu werden verschiedene Web Services entwickelt, welche die Pandas-basierten Simulation-Workflows zulassen.

Die Möglichkeit zur weiteren Modifikation wurde mit dieser Arbeit geschaffen. Für ein neues Simulationsszenario, wie zum Beispiel bei der Kopplung von Pandas mit Matlab, muss nur ein neues Table in der Datenbank angelegt werden, wohin Pandas die Daten speichern kann, und die entsprechenden Web Service-Operationen müssen hinzugefügt werden.

[3] http://www.dune-project.org/

2 Grundlagen

In diesem Kapitel wird ein kurzer Einblick in die grundlegenden Konzepte und Technologien gegeben, welche zum Verständnis der Ausarbeitung benötigt werden. Neben den Begriffen Service-orientierte Architekturen, Web Services, Workflows und Business Process Execution Language wird der Bereich e-Science erläutert. Den Abschluss bildet die Einführung in die Simulation der Strukturänderungen im Knochen, welche unter anderem mit dem Framework Pandas durchgeführt wird.

2.1 Service-orientierte Architektur

Dieses Kapitel basiert auf dem Buch [3]. Bei der Service-orientierten Architektur (SOA) handelt es sich nicht um eine bestimmte Technologie, sondern vielmehr beschreibt sie losgelöst von einer konkreten Implementierungsmethode einen speziellen Architekturstil. Der zentrale Bestandteil einer SOA ist das Konzept des Services oder des Dienstes. Ein Dienst kann seine Funktionalität teilweise oder komplett über eine öffentliche wohldefinierte Schnittstelle anbieten. Die so bereitgestellten Dienste können von anderen Anwendungen dynamisch gebunden werden und sind miteinander lose gekoppelt. Die Kommunikation zwischen den Diensten erfolgt nachrichtenbasiert in der Regel über ein Netzwerk.

Im Rahmen der SOA gibt es drei verschiedene Rollen: den Service Provider, den Service Client und das Service Verzeichnis. Der Service Provider bietet einen Dienst an. Das Service Verzeichnis wird von dem Service Provider dazu verwendet seinen Dienst, unter der Angabe einer abstrakten Beschreibung, zu veröffentlichen. Der Service Client kann dann in dem Service Verzeichnis nach einem Service mit einer bestimmten Funktionalität suchen und erhält von Service Verzeichnis die für den Service Client geeigneten Servicebeschreibungen zurück. Der Service Client kann dann, mit der zurückgelieferten Servicebeschreibung, den Dienst des Service Providers nutzen. Den Zusammenhang zwischen diesen Rollen ergibt das sogenannte SOA-Dreieck und ist auf der Abbildung 1 dargestellt.

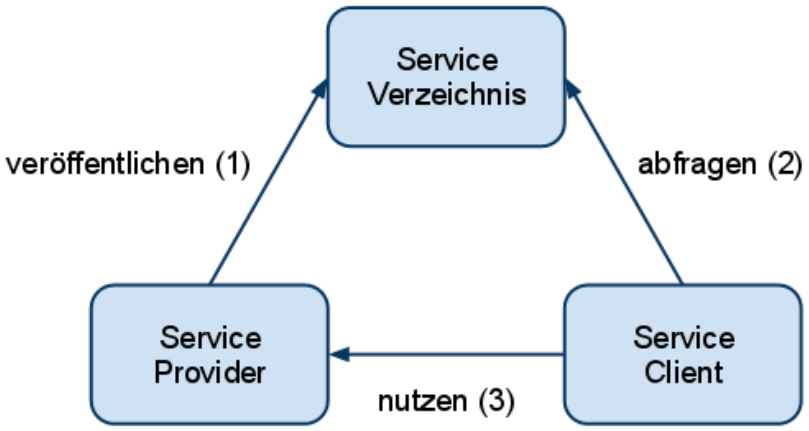

Abbildung 1: SOA-Dreieck

Durch die lose Kopplung ist eine späte Dienstauswahl seitens des Service Clients möglich. Dies sorgt für eine hohe Flexibilität bei der Ausführung und erhöht die Wiederverwendbarkeit von Komponenten.

2.2 Web Services

Web Services stellen eine Möglichkeit der technischen Realisierung einer SOA dar. In [4] wird von dem World Wide Web Consortium (W3C) ein Web Service wie folgt definiert:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

Der Unterschied zwischen dem Web Service Ansatz und der älterer Ansätze, wie zum Beispiel dem Common Object Request Broker Architecture (CORBA) Ansatz oder dem Distributed Component Object Model (DCOM) Ansatz, liegt in dem Aspekt der losen Kopplung und der Plattformunabhängigkeit.

Die Web Service Architektur umfasst viele aufeinander aufbauende Technologien. Dieser modulare Aufbau wird Web Service Stack genannt und wird auf Abbildung 2 dargestellt.

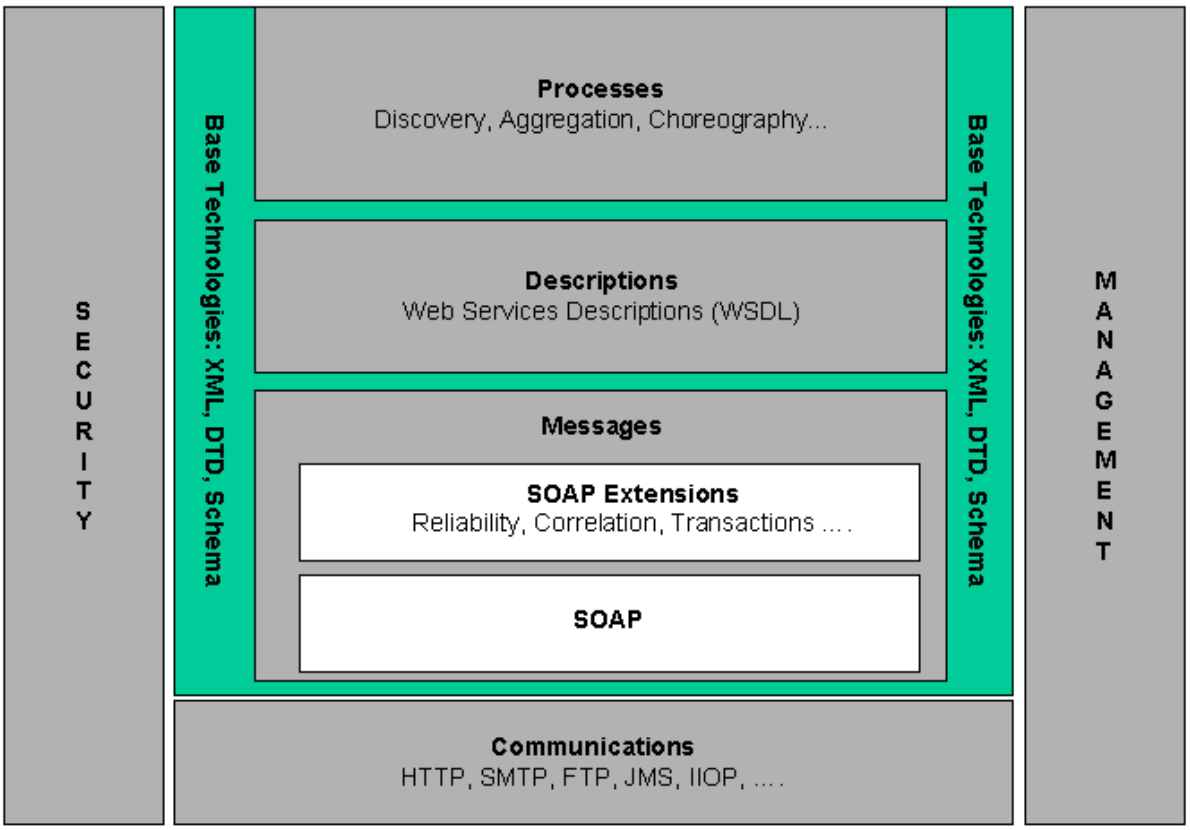

Abbildung 2: Web Service Stack aus [4]

Einen umfassenden Überblick über hierüber kann man über das Buch [3] oder dem Artikel [4] erlangen.

2.2.1 WSDL

Die Web Service Description Language (WSDL) ist die formelle Beschreibungssprache für Web Services. Die Sprache basiert auf XML und ist dabei plattform-, protokoll- und programmiersprachenunabhängig. In einem WSDL-Dokument wird beschrieben was für Funktionen der Web Service hat, wie darauf zugegriffen werden kann und wo sich der Web Service Endpunkt befindet. Es besteht im Allgemeinen aus zwei Arten von Definitionen, einem abstrakten Teil und einem konkreten Teil. Der abstrakte Teil beschreibt was der Dienst an Funktionalitäten bietet. Der konkrete Teil beschreibt wie der Dienst wo aufgerufen werden kann. Abbildung 3 zeigt den strukturellen Aufbau eines WSDL-Dokuments.

Abbildung 3: WSDL 1.1 Struktur aus [3]

Für vertiefende Informationen über den genauen Aufbau von WSDL-Dateien sei auf die Spezifikation des W3C [5] oder auf das Buch [3] verwiesen.

2.2.2 SOAP

Bei SOAP handelt es sich um ein Nachrichten-Framework, welches einerseits das Format und andererseits die Verarbeitung der Nachrichten von Sender zu Empfänger definiert. Die Abbildung 4 zeigt den Aufbau einer solchen SOAP-Nachricht nach [3]. Eine SOAP-Nachricht besteht aus zwei Bereichen: dem Header-Bereich und dem Body-Bereich.

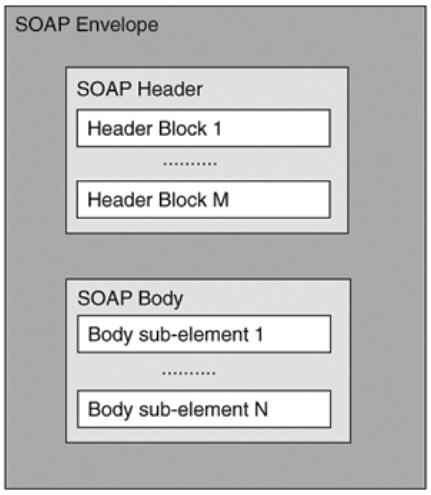

Abbildung 4: Aufbau von SOAP-Nachrichten aus [3]

Der Header-Bereich ist optional und kann mehrere SOAP-Header enthalten. Die Header können von jedem SOAP-Empfänger auf dem Nachrichtenpfad verarbeitet werden. Wie auf Abbildung 5 zu sehen, kann der Nachrichtenpfad vom ursprünglichen SOAP-Sender bis endgültigen SOAP-Empfänger über beliebig viele Zwischenstationen verlaufen. Die Zwischenstationen können dabei die Nachricht weiterverarbeiten oder einfach weiterleiten. Wenn eine Zwischenstation einen Header auswertet, wird dieser in der Regel entfernt. Durch das Setzen eines speziellen Attributs, kann dies jedoch verhindert werden.

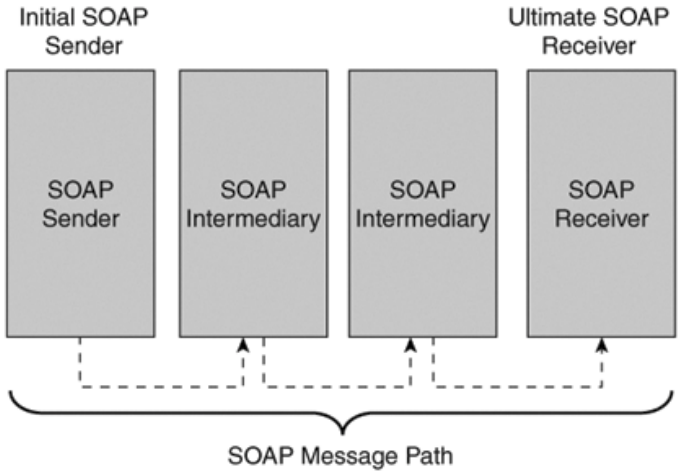

Abbildung 5: SOAP Nachrichtenpfad aus [3]

Der Body-Bereich enthält die Nutzdaten für den Empfänger, wie zum Beispiel vom Dienst zu verarbeitende Eingabedaten. Da SOAP auf XML basiert, werden binäre Eingaben entweder Base64-kodiert im Body-Bereich verwendet oder als SOAP-Attachment an die Nachricht angehängt.

Für weiterführende Informationen zu SOAP sei auf [3] oder [6] verwiesen.

2.2.3 WS-Addressing

WS-Addressing wurde in [7] spezifiziert und dient der Beschreibung von Web Service Endpunkten. Diese Endpunktbeschreibungen können in SOAP-Nachrichten eingebettet werden, um Dienstanbietern und Dienstnutzern den Nachrichtenaustausch zu ermöglichen, auch wenn diese zur Entwicklungszeit noch nicht feststanden. WS-Adressing ist, wie Web Services auch, nicht an ein bestimmtes darunterliegendes Transportprotokoll gebunden.

Dabei spezifiziert WS-Addressing zwei grundlegende Konzepte:

- Endpoint References (EPR) als Zeiger auf einen Web Service Endpunkt
- Möglichkeiten EPRs in SOAP-Nachrichten zu verwenden

WS-Addressing findet im Normalfall bei einem asynchronen Web Service Aufruf Anwendung. Das heißt immer dann, wenn die Anfrage und die Rückantwort nicht im selben Kontext stehen und über separate Einwege-Nachrichten ausgetauscht werden. Hierzu werden mehrere Header definiert, die es ermöglichen EPRs in SOAP-Nachrichten einzubetten. Abbildung 6 zeigt das Erstellen einer Rückantwort-Nachricht eines Dienstes und das Zusammenwirken der EPRs in Anfrage- und Rückantwort-Nachricht.

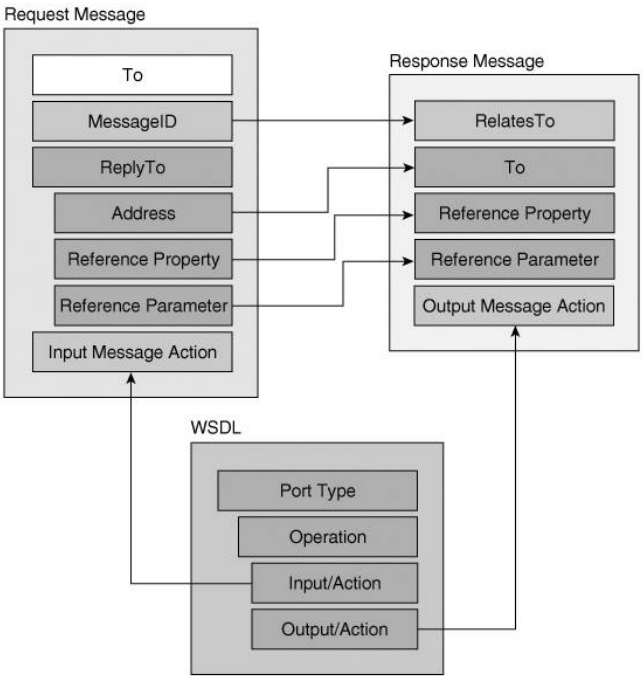

Abbildung 6: WS-Addressing Headers in Request- und Response-Nachrichten aus [3]

Die verschiedenen Message Headers werden in [8] spezifiziert und in [3] ausführlicher beschrieben.

2.2.3.1 End Point Reference

Eine Endpoint Reference (EPR) ist eine Datenstruktur, die alle Informationen enthält um einen Web Service zur Laufzeit zu adressieren. Nach [3] können Endpoint References zusätzliche Informationen enthalten, die sich in zwei Klassen einteilen lassen, die Runtime-Information, welche für die Adressierung benötigt wird und die damit verknüpften Metadaten, die den Endpunkt zusätzlich beschreiben.

Die Runtime-Information bestehen aus drei Feldern. Das Adressfeld ist zwingend erforderlich und gibt den Endpunkt an. Die Referenz-Properties sind optional und enthalten Daten, die bei der Zustellung zum Endpunkt verwendet werden können. Die Referenz-Parameter sind ebenfalls optional und enthalten Daten, die vom Endpunkt verwendet werden können.

Die Metadaten ihrerseits sind optional und hat drei Felder und sind in [9] spezifiziert. Darin können Informationen wie anzuwendende Policies oder die Namen des WSDL services, porttypes oder ports enthalten sein.

Die Abbildung 7 zeigt die Beziehungen zwischen EPR, WSDL und den WS-Addressing Message Headers in einer SOAP-Nachricht.

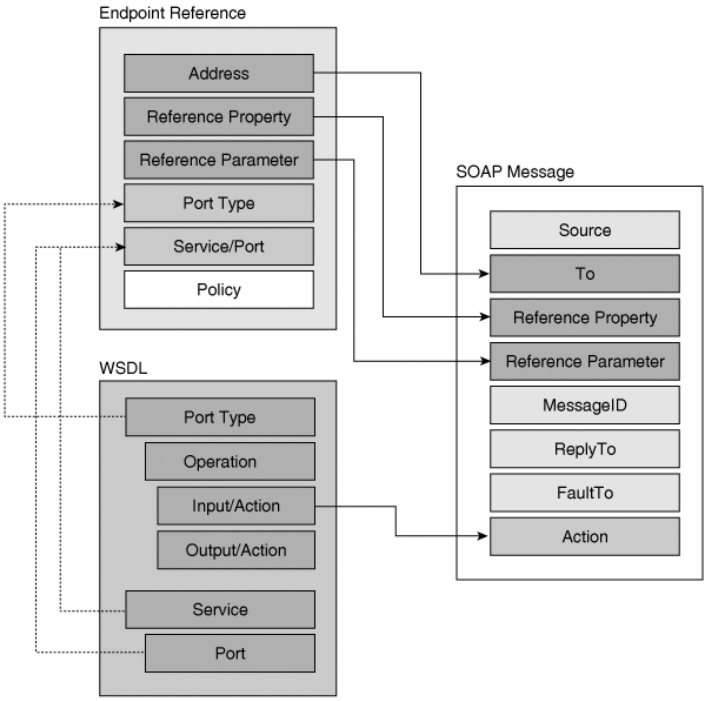

Abbildung 7: WS-Addressing in SOAP-Nachrichten aus [3]

2.3 Workflows

In diesem Unterkapitel wird ein Einblick über das Thema Workflows, indem zunächst auf das Modell und deren Instanziierung sowie auf die Dimensionen eines Workflows eingegangen wird.

Anschließend wird erläutert was ein Workflow Management System (WfMS) ist. Die Sprache BPEL wird abschließend erläutert.

2.3.1 Prozessmodell und Instanz

Nach [10] beschreibt das Prozessmodell die Struktur eines (Geschäfts-) Prozesses in der realen Welt. Das Modell definiert alle möglichen Pfade durch den (Geschäfts-) Prozess, einschließlich der Verzweigungsregeln und den auszuführenden Aktivitäten. Dieses Modell dient als Vorlage für jeden Prozess der ausgeführt wird. Aus dem Prozessmodell wird eine sogenannte Prozessinstanz erzeugt. In einer Versicherungsgesellschaft könnte es zum Beispiel ein Prozessmodell für die Schadensbearbeitung geben und von diesem Modell verschiedene Prozessinstanzen für verschiedene Personen. Dabei hat jede Prozessinstanz seine eigenen Werte, die den genommenen Pfad durch das Modell beschreiben.

Prozesse müssen dabei nicht unbedingt auf einem Computer ausgeführt werden. Ein (Geschäfts-) Prozess kann aus Teilprozessen bestehen, die auf einem Computer ausgeführt werden, und aus Teilprozessen, die von Personen ausgeführt werden müssen. Dies führt zu dem Begriff des Workflowmodells, welches die Teilprozesse eines Prozessmodells bezeichnet, die auf einem Computer ausgeführt werden. Das Workflowmodell kann hierbei ein kleiner Teilprozess eines größeren Prozessmodells sein oder es kann das gesamte Prozessmodell umfassen. Wie bei dem Prozessmodell dient das Workflowmodell als eine Vorlage für die Instanziierung von Workflowinstanzen. Abbildung 8 stellt die Zusammenhänge zwischen Prozessmodell und Workflowmodell und deren Instanziierung dar.

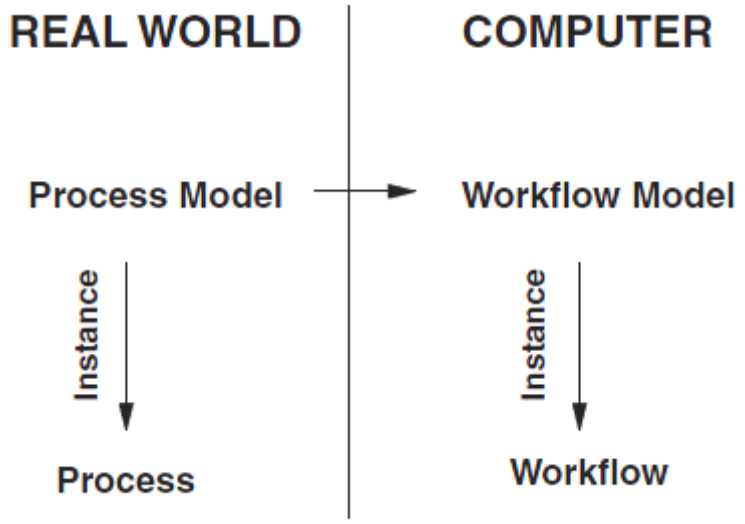

Abbildung 8: Prozesse und Workflows aus [10]

Da sich diese Arbeit ausschließlich mit Prozessen beschäftigt, welche auf Computern ausgeführt werden, werden im Folgenden die Begriffe Prozessmodell und Workflowmodell sowie Prozessinstanz und Workflowinstanz jeweils synonym verwendet.

2.3.2 Workflow Dimensionen

In dem Buch [10] werden Prozesse in drei verschiedene Dimensionen aufgespalten:

Wie:

Diese Dimension steht für die Prozesslogik und beschreibt die Aktivitäten, die ausgeführt werden sollen, und in welcher Reihenfolge die Ausführung stattfinden soll. Die Aktivitäten können hierbei sequentiell als auch parallel ausgeführt werden.

Wer:

Diese Dimension beschreibt die Organisationsstruktur des Unternehmens im Hinblick auf Abteilungen, Rollen und Personen. Diese Information wird dazu verwendet, um festzulegen wer die Aktivität auszuführen hat. Dabei kann für jede Aktivität eine Query angegeben werden, um ermitteln zu können, wer diese Aktivität in der Organisation ausführen kann. Wenn die Aktivität nicht von einem Benutzer ausgeführt werden muss, so führt das Workflow Management System diese Aktivität selbst aus.

Womit:

Diese Dimension beschreibt welche IT-Ressourcen für die auszuführende Aktivität eingesetzt werden soll.

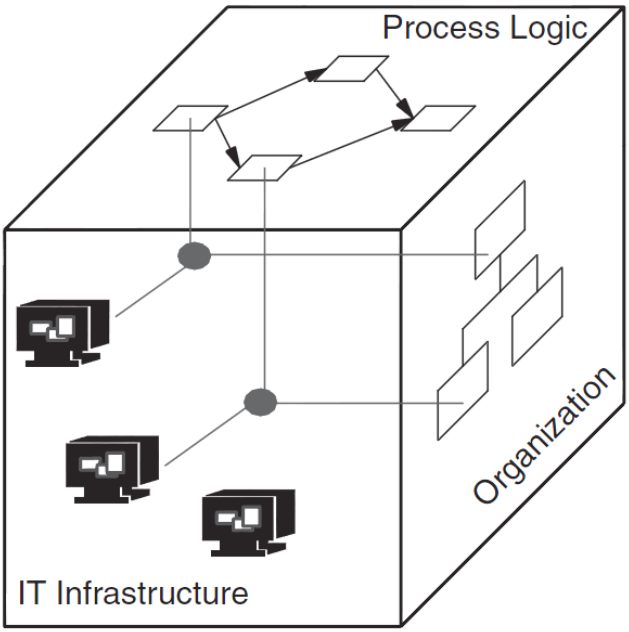

Abbildung 9: Die Workflow Dimensionen aus [10]

Die Abbildung 9 stellt die drei Workflow Dimensionen als Würfel dar. Nach [10] kann die Ausführung eines Prozesses als eine Folge von Punkten in dem dreidimensionalen Workflowraum (wie, wer, womit) aufgefasst werden.

2.3.3 Workflow Management Systeme

Ein Workflow Management System umfasst Anwendungen, welche der aktiven Steuerung von Prozessen dient. Dazu gehört es Workflows zu definieren, zu verwalten und auszuführen. Ein Workflow Management System wird von der Workflow Management Coalition (WfMC) in [11] wie folgend definiert:

> *A system that completely defines, manages and executes "workflows" through the execution of software whose order of execution is driven by a computer representation of the workflow logic.*

Die WfMC gliedert ein WfMS in ihrem Referenzmodell in verschiedene Funktionsbereiche:

Buildtime

Dieser Funktionsbereich beinhaltet die Komponenten, die für die Erstellung und Modellierung von Prozessmodellen und für die Verwaltung von Ressourcen zuständig sind.

Runtime

Dieser Funktionsbereich enthält alle Komponenten, welche zur Ausführung von Prozessinstanzen benötigt werden.

Database

Dieser Funktionsbereich umfasst die komplette Datenhaltung aus den Buildtime und Runtime Bereichen. Die Datenhaltung erfolgt in der Regel über Datenbanken.

In dem Buch [10] wird ein WfMS zusätzlich noch in den folgenden funktionalen Bereich unterteilt:

Metamodel

Hierbei ist dieser Funktionsbereich eine weitere Unterteilung des Buildtime Bereichs. Hier werden die Konstrukte und die damit verbundenen Funktionen, wie zum Beispiel die Struktur eines Prozessmodells, die von dem WfMS unterstützt werden, definiert.

Abbildung 10 zeigt die Hauptkomponenten eines WfMS und deren Beziehungen untereinander.

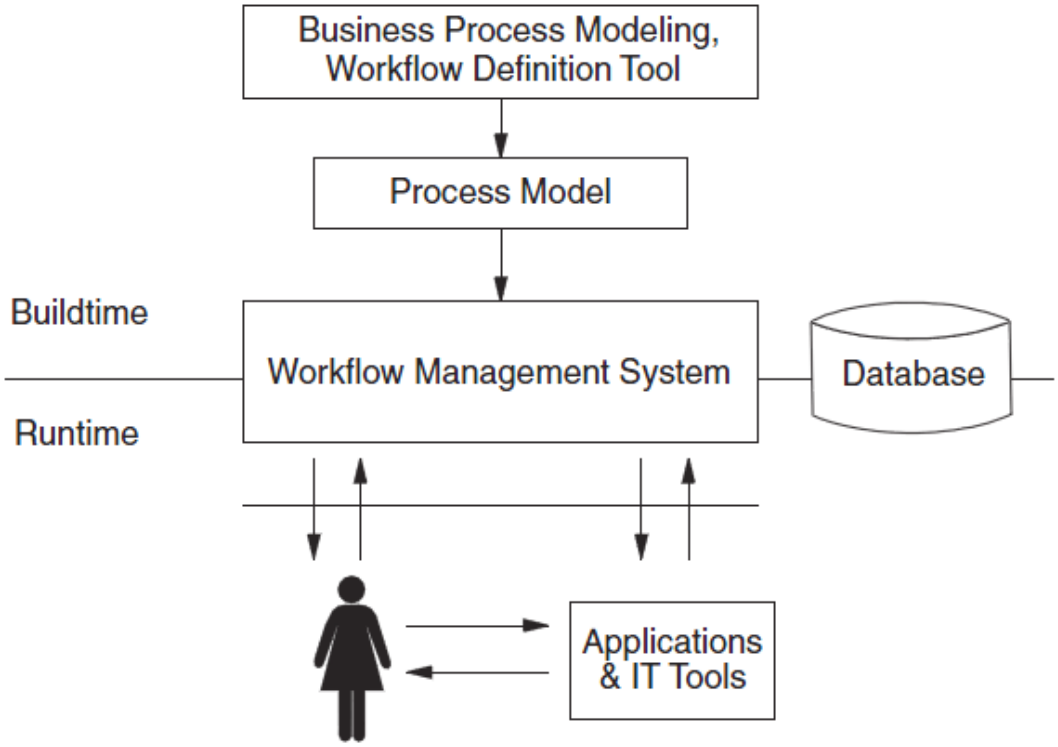

Abbildung 10: Komponenten eines Workflow Management System aus [10]

Für vertiefende Informationen in das Thema der Workflow Management System, sei auf die Lektüre des Buches [10] verwiesen.

2.3.4 BPEL

Die Informationen für dieses Kapitel stammen aus der BPEL-Spezifikation [12] und dem Buch [3]. Die Business Process Execution Language (BPEL) ist Teil der sogenannten WS-*-Spezifikationen. Der eigentliche Name ist daher auch WS-BPEL. Dabei ist BPEL eine XML-basierte Sprache zur Beschreibung von Prozessen und geht als Nachfolger der Sprachen WSFL und XLANG hervor.

Die Aktivitäten des Prozesses sind durch Web Services implementiert und der Prozess selbst, kann als Web Service angeboten werden. Die Daten in BPEL werden in typisierten Variablen gespeichert. Der Typ einer Variablen kann entweder ein einfacher oder komplexer Typ aus einem XML-Schema sein oder durch eine wsdl:message festgelegt sein. Der Zugriff auf die Variablen erfolgt über XPath [13].

Ein BPEL-Prozess ist blockstrukturiert, das heißt er besteht aus verschachtelten Scopes, die lokale Variablen, Aktivitäten, Partner Links und diverse Handler enthalten können. Ein Scope bündelt eine Reihe von Aktivitäten zu einer transaktionalen Einheit.

In BPEL gibt es folgende Handler Typen: Event Handler, Fault Handler und Compensation Handler. In einem Scope können unterschiedliche Handler vorhanden sein und jeder Handler wiederum kann Aktivitäten enthalten, die bei der Aktivierung des entsprechenden Handlers ausgeführt werden.

Die Ausführung des Event Handler läuft parallel zum Prozess und dient zur Bearbeitung von Ereignissen. Ein Fault Handler wird einem Scope zugeordnet und wird innerhalb dieses Scopes aufgerufen, wenn ein Fehlerfall auftritt. Ein Compensation Handler wird einem Scope zugeordnet und wird in der Regel über die compensate-Aktivität aus einem Fault Handler heraus aufgerufen. Durch diesen Handler werden lang-andauernde Transaktionen ermöglicht. Die verschiedenen Handler und deren Anwendung werden in [12] näher erläutert.

Die Aktivitäten werden in BPEL in Basic Activities und in Structured Activities unterschieden. Die Basic Activities, sozusagen atomaren Aktivitäten enthalten keine weiteren Aktivitäten und ihr Aufruf wird als atomare Operation betrachtet. Da BPEL rekursiv definiert ist und selbst als Web Service aufgerufen werden kann, ist der Aufruf eines Prozesses in einem Prozess auch als atomar zu betrachten. Strukturierte Aktivitäten beinhalten selber andere Aktivitäten und lassen somit eine beliebig komplexe Zusammensetzung von Abläufen zu. Die genauen Aktivitäten werden in [12] ausführlich beschrieben.

Durch die Partner Links wird in BPEL-Prozessen ein Partner und dessen Rolle beschrieben. Für alle Aktivitäten, die mit Web Services interagieren, werden Partner Links verwendet. Dabei basiert ein Partner Link auf einen PartnerLinkType aus einer WSDL-Datei. Der PartnerLinkType wiederum definieren einen oder mehrere Rollen und jeder Rolle wird ein portType zugeordnet. Bei der Instanziierung des Prozessmodels wird jeder Rolle ein konkreter Web Service Endpunkt zugeordnet.

2.3.5 Scientific Workflows

Dieser Abschnitt basiert hauptsächlich auf [1]. In den letzten Jahren bekam die Workflow-Technologie immer mehr Aufmerksamkeit im wissenschaftlichen Bereich, wodurch sich der Begriff der Scientific Workflows geprägt hat. In der Wissenschaft bieten Workflows einige Vorteile:

- Sie helfen bei der Wissensweitergabe, indem sie als Dienst für zusammenarbeitende Wissenschaftler zur Verfügung stehen.
- Mit Hilfe von Workflows wird eine Ergebnisanalyse durch die Community ermöglicht.
- Workflows können mit großen Datenmengen umgehen, welche zum Beispiel von Sensoren gesammelt werden.
- Workflows sind in verteilten und hoch heterogenen Umgebungen lauffähig. Dies stellt ein häufiges Szenario bei wissenschaftlichen Berechnungen dar, bei der einen Vielzahl von Plattformen und Programmiersprachen zum Einsatz kommen.
- Durch die Automatisierung der Schritte während des Workflow-Design und der Ausführung, können sich Wissenschaftler auf die Lösung ihrer wissenschaftlichen Probleme konzentrieren.
- Workflows können dazu verwendet werden, um wissenschaftliche Simulationen parallel und automatisiert durchzuführen.

Da sich die Web Service- und Workflow-Technologien im Business-Bereich etabliert haben und hier auch schon Standards und Tools vorhanden sind, ist es sinnvoll diese auch im wissenschaftlichen Bereich zu verwenden. Unterschiede zwischen Business- und Scientific-Workflows werden durch den Vergleich der Lebenszyklen deutlich.

Abbildung 11 zeigt die Lebenszyklen eines Business-Workflows (a) und eines Scientific-Workflows (b), welche anschließend weiter erläutert werden.

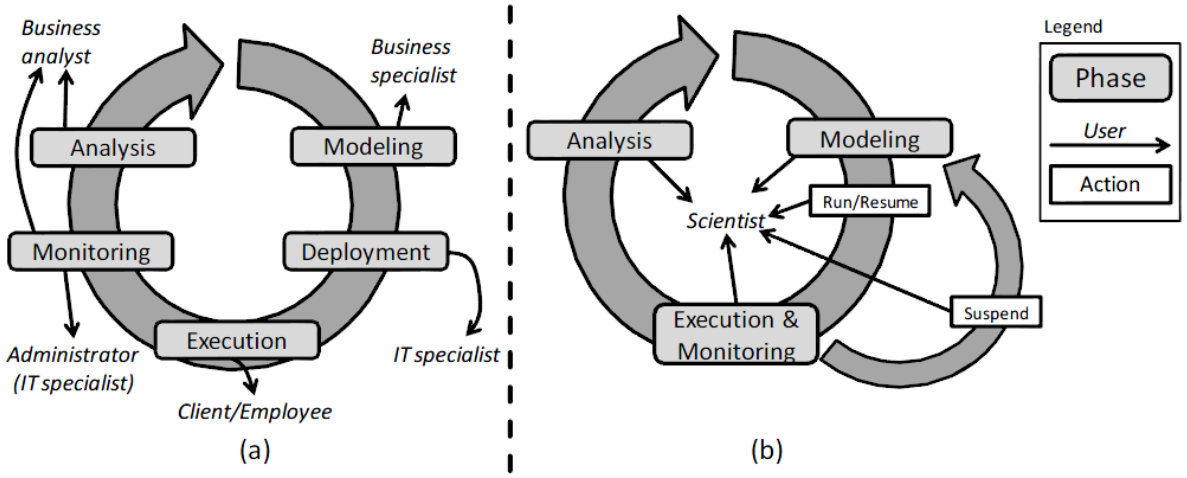

Abbildung 11: Lebenszyklus eines Business-Workflows (a) und eines Scientific-Workflows aus [1]

Ein Workflow wird durch Business-Spezialisten modelliert, der die konkreten Schritte zur Erreichung eines bestimmten Geschäftsziels kennt. Ein IT-Spezialist macht den Workflow ausführbar, indem er ihn auf einer Workflow-Engine bereitstellt. Die Ausführung eines Workflows wird durch einen Angestellten oder Kunden angestoßen. Die Ausführung erfolgt oft erst spät nach dem Deployment und das Workflow-Modell kann mehrfach instanziiert werden. Das Workflow-Monitoring kann mit einzelnen Workflow-Instanzen umgehen und ebenso über mehrere Instanzen Daten zusammenfassen. Das Monitoring kann über den Gesamtzustand eines Systems einschließlich der laufenden und beendeten Workflow-Instanzen Statistiken bereitstellen. Daher kann die Überwachung sowohl für Administratoren als auch für Business-Analysten wertvoll sein. Schließlich kann ein Business-Analyst einen oder mehrere Workflow-Ausführungen analysieren und daraus den Bedarf an Modelländerungen aufzeigen.

Der Lebenszyklus bei Scientific Workflows unterscheidet sich stark zu dem des Business Workflows. Normalerweise übernimmt hier der Wissenschaftler alle Rollen aus dem Business-Workflow-Lebenszyklus. Eine Unterscheidung zwischen Workflow-Modell und Workflow-Instanz findet nicht statt, das heißt der Schwerpunkt ihrer Arbeit liegt auf einer einzelnen Workflow-Instanz. Die Phasen der Modellierung und Ausführung sind nicht in einer strengen Reihenfolge angeordnet, da Wissenschaftler ihre Workflows in einer Trial-and-Error Weise entwickeln. Tatsächlich werden beide Phasen abwechselnd ausgeführt. Durch eine Suspend-Operation wird ein weiterer Zyklus von der Ausführungsphase zur Modellierungsphase gestartet. Die technischen Details bleiben hierbei für den Wissenschaftler transparent. Nach dem Modellieren beginnt sofort die Ausführung des Workflows. Die Phasen der Ausführung und des Monitorings aus dem Lebenszyklus herkömmlicher Workflows sind im Lebenszyklus der Scientific Workflows zusammengelegt. Aus der Sicht eines Wissenschaftlers visualisiert das Monitoring nur die laufende Workflow-Instanz. Nach der Ausführung kann ein Wissenschaftler die berechneten Ergebnisse analysieren, woraufhin der Workflow neu modelliert und, möglicherweise mit geänderten Parametern, erneut ausgeführt werden kann.

Der Lebenszyklus des Scientific Workflows zeigt die Möglichkeit vorhandene Workflow-Technologien zu verwenden, verdeutlicht jedoch auch die Notwendigkeit für Erweiterungen, um für den Einsatz von wissenschaftlichen Experimenten, Berechnungen und Simulationen geeignet zu sein.

2.4 eScience

Der Begriff e-Science steht für Enhanced Science und ist ein Sammelbegriff für groß angelegte wissenschaftliche Projekte, in denen Wissenschaftler weltweit über das Internet zusammenarbeiten und in denen Grid-Computing benutzt wird. Der Begriff wurde in Großbritannien geprägt, welche mehrere e-Science Zentren im ganzen Land haben [14]. Das *National e-Science Centre*[4] aus Großbritannien definiert e-Science folgendermaßen:

> *In the future, e-Science will refer to the large scale science that will increasingly be carried out through distributed global collaborations enabled by the Internet. Typically, a feature of such collaborative scientific enterprises is that they will*

[4] http://www.nesc.ac.uk/index.html

require access to very large data collections, very large scale computing resources and high performance visualisation back to the individual user scientists.

Nach [15] war die Wissenschaft verschiedenen Paradigmen Wandeln unterworfen, von der empirischen Beschreibung der Natur über die Bildung theoretischer Modelle hin zur Simulation komplexer Phänomene. Demnach beschreibt e-Science den aktuellen Paradigmen Wandel, wie die Daten erfasst, verarbeitet und bereitgestellt werden.

Die deutsche E-Science Initiative[5] unterteilt e-Science in folgende vier Bereiche:

- Grid-Computing: eine Form des verteilten Rechnens, um rechenintensive Probleme zu lösen
- Wissensvernetzung: die methodische Einflussnahme auf die vorhandene Wissensbasis
- E-Learning: alle Formen von Lernen, bei denen elektronische Medien zur Distribution von Lerninhalten zum Einsatz kommt
- Open Access: der freie Zugang zu wissenschaftlicher Literatur und anderen Materialien

Somit stellt e-Science eine Art Infrastruktur mit technologischen und sozialen Aspekten für die moderne Wissenschaft dar.

2.4.1 Simulationen

In diesem Unterkapitel wird über das Thema Simulationen ein kleiner Einblick gegeben.

Der Begriff Simulation wird von jedem Fachbereich, mit einem Schwerpunkt auf unterschiedlichen Aspekten, definiert. Da es keine einheitliche Definition gibt, wird in [2] folgende weitgefasste und abstrakte Definition des Begriffs Simulation aufgestellt:

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.

Durch die Simulation wird demnach das Verhalten eines Systems oder Prozesses nachgeahmt. Um Erkenntnisse über das System zu erlangen, werden Experimente an dem Modell der Simulation durchgeführt. Modelle und Simulationen sind eng miteinander verwandt. Dabei stellt das Simulationsmodell eine Abstraktion des nachzuahmenden Systems dar. Daher stellt bei einer Simulation die Erstellung eines validen Modells den ersten Schritt dar. Ein valides Modell selbst ist ein beschränktes Abbild aus der Realität und zeichnet sich nach [16] durch folgende drei Merkmale aus:

Abbildung: Modelle sind stets Modelle von einem natürlichen oder künstlichen Original. Ein Modell ist ein Abbild oder ein Repräsentant des Originals, welches selbst wieder ein Modell sein kann.

[5] http://www.ges2007.de/

Verkürzung: Modelle erfassen üblicherweise nicht alle Attribute des durch sie dargestellten Originals. Es werden von dem Modellerschaffer diejenigen Originalattribute ausgewählt, die der Zielsetzung der Modellbenutzung dienen.

Pragmatismus: Modelle erfüllen eine Ersetzungsfunktion und sind nicht per se zu ihren Originalen zugeordnet. Das Modell muss bezüglich seiner spezifischen Funktion durch die Fragen für wen, wann und wozu relativiert werden. Das heißt ein Modell ist nicht nur ein Modell von etwas, sondern auch für jemanden und es erfüllt einen bestimmten Zweck innerhalb eines Zeitintervalls.

An diesem validen Modell können dann verschiedene Parameter für jeden Simulationslauf geändert werden. Aus den Ergebnissen der Simulation lassen sich dann Rückschlüsse auf das gestellte Problem und deren Lösung ziehen.

Simulationen lassen sich grob mit folgenden Aspekten unterteilen:

Computer: Basiert die Simulation auf einem physischen Modell, bei dem diverse Messungen erfolgen, wie zum Beispiel einem Auto-Modell im Windkanal bei dem Strömungswerte gemessen werden oder basiert die Simulation auf einem Computermodell, bei dem die Simulation vollständig auf einem Computer läuft.

Zeit: Ist die Zeitdimension Teil der Simulation (dynamische Simulation) oder spielt diese keine Rolle (statische Simulation).

Zufall: Werden bei der Simulation zufällige Ereignisse mit eingeschlossen (stochastische Simulation) oder werden dieses ausgeschlossen (deterministische Simulation).

In die Bereiche, die durch die verschiedenen Aspekte, und teilweise deren Kombination miteinander, beschrieben werden können, fallen viele verschiedene Simulationsarten, welche an dieser Stelle nicht weiter erläutert werden.

In [2] werden Simulationen in folgende fünf Funktionsbereiche eingeteilt:

Simulation als Technik

Mit Hilfe von Simulationen können Wissenschaftler, detailliert die Dynamik eines echten Prozesses untersuchen. In vielen Fällen ist es nicht möglich die Daten experimentell zu erfassen, da die betrachtete Zeitskala zu groß ist, zum Beispiel bei der Evolution von Galaxien, oder zu klein, wie zum Beispiel bei nuklearen Reaktionen.

Simulation als heuristisches Werkzeug

Simulationen spielen eine wichtige Rolle in der Entwicklung von Hypothesen, Modellen und sogar neuen Theorien. Durch die Ausführung verschiedener Simulationsläufe mit geänderten Eingabeparametern, können neue und einfache Regeln abgeleitet werden, welche ansonsten nicht über die Modellannahmen hätten gemacht werden können. Einige dieser Hypothesen wiederum können als Grundannahmen für ein neues und einfacheres Modell dienen.

Simulation als Ersatz für Experimente

Simulationen können Wissenschaftlern helfen Situationen zu untersuchen, die experimentell nicht untersucht werden können. Die Ausführung eines Experiments kann sich aus pragmatischen, theoretischen und ethischen Gründen als unmöglich erweisen. Ein Beispiel für pragmatisch unmögliches Experiment wäre die Untersuchung der Entstehung von Galaxien. Ein Beispiel für ein theoretisch unmögliches Experiment wären Situationen, bei denen fundamentale Konstanten, wie zum Beispiel die Ladung eines Elektrons, geändert werden. Ein Beispiel für ein ethisch unmögliches Experiment wäre eine Untersuchung der Langzeitauswirkungen einer Einkommenssteuererhöhung um den Faktor 1,5. In all jenen Fällen ist das Beste, was ein Wissenschaftler tun kann, eine angemessene Simulation durchführen.

Simulation als Werkzeug für experimentell Arbeitende

Eine Simulation kann bei realen Experimenten bei der Planung, Entwicklung und Ausführung hilfreich sein. Simulationen können reale Experimente inspirieren, wenn aus den Simulationsergebnissen sich neue Regeln oder Hypothesen ergeben haben. Ebenso können Simulationen bei der Vorauswahl von möglichen System und Aufbauten für das Experiment helfen. Hierbei werden verschiedene Experimentaufbauten simuliert und das am meisten erfolgversprechendste ausgewählt. Bei der Analyse von Experiment-Ergebnissen, können Simulationen helfen, indem triviale oder gut verstandene Zusammenhänge aus den Daten ausgeblendet werden können.

Simulation als pädagogisches Werkzeug

Simulationen haben sich in der Ausbildung als äußert nützlich erwiesen. Durch das „Spielen“ an einem Simulationsmodell und der Visualisierung der Ergebnisse, wird das Verständnis des darunterliegenden Prozesses gefördert und eine Intuition für ähnliche Experimente entwickelt. Diese Art des Lernens ist in vielen Fällen billiger und schneller, als ein reales Experiment auszuführen.

2.4.2 FEM

Bei der Finite Elemente Methode (FEM) handelt es sich um ein numerisches Verfahren zur Lösung von partiellen Differentialgleichungen (pDGL). Diese Methode findet in der Technik und in der Physik zur Lösung von Problemen Anwendung. Die FEM stellt einen Oberbegriff für unterschiedliche Ansätze für unterschiedliche Problemstellungen dar. Die zu lösenden Problemstellungen können hierbei zum Beispiel eine Crash-Test-Simulation (Körperverformung) oder eine Diffusions-Simulation verschiedener Flüssigkeiten (keine Körperverformung) sein. Für die verschiedenen Problemstellungen müssen unterschiedliche Ansätze, wie zum Beispiel der Ritz-Ansatz bei Körperverformungen oder der Galerkin-Ansatz für zeitabhängige Probleme, verwendet werden.

Der Galerkin-Ansatz lässt sich auf einen großen Problemkreis anwenden und wird im Folgenden erläutert. Bei der FEM wird zum Lösen der Differentialgleichung (DGL) das Grundgebiet in endlich viele Teilgebiete unterteilt. Dieses Grundgebiet stellt somit die zu simulierende Welt dar und wird

durch ein Netzwerk aus endlich vielen kleineren Gebieten angenähert. Diese kleineren Gebiete werden Elemente genannt und haben meist eine einfach zu beschreibende geometrische Form. Aus der endlichen Anzahl der Elemente leitet sich der Name Finite Elemente ab. Diese Elemente werden dann diskretisiert. Dazu werden an bestimmten Stellen in den Elementen, den sogenannten Knotenpunkten, Gleichungen aufgestellt. Diese Gleichungen beschreiben näherungsweise die gesuchte Lösungsfunktion. Der Zustand eines Elements wird somit durch eine Matrix von Gleichungen beschrieben. Aus den Matrizen der Elemente lässt sich eine globale Matrixgleichung aufstellen, die den Zustand des gesamten Systems beschreibt. Die Matrixgleichung wird numerisch gelöst. Dass der Galerkin-Ansatz unter strengen Bedingungen konvergiert, wurde mathematisch bewiesen [17]. Eine Bedingung für die Konvergenz des Galerkin-Ansatzes ist, dass die gesuchte Lösungsfunktion eine bestimmte Form besitzen muss.

Gebietsunterteilung in Elemente

Die Auswahl der Geometrie der Elemente, die das Grundgebiet annähern sollen, ist Problemabhängig. Jedoch besitzen sie meist eine einfache Geometrie, die auf Drei- oder Vierecken basieren. Die Elementgeometrie kann jedoch auch durch eine Vektorfunktion beschrieben werden. Durch die gewählte Geometrie ergibt sich eine gewisse Anzahl von Knoten und Kanten. Die einzelnen Elemente werden über diese Knoten miteinander verbunden.

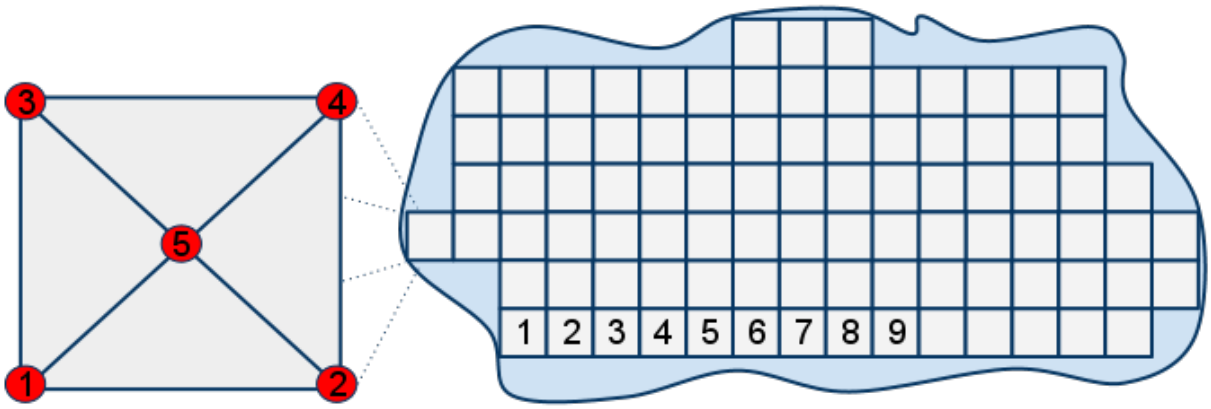

Abbildung 12: finite Elemente nähern Grundgebiet an

Abbildung 12 zeigt ein quadratisches Element und wie diese miteinander verbundenen Elemente das Grundgebiet annähern. Die miteinander verbundenen Elemente bilden zusammen das sogenannte Gitter. Die Elemente werden innerhalb dieses Gitters eindeutig nummeriert, wodurch die Reihenfolge der Elementberechnung festgelegt wird. Da die Reihenfolge einen großen Einfluss auf das Ergebnis der Simulation hat, gibt es unterschiedliche Nummerierungsverfahren.

Lösen der DGL

Die zu lösende DGL hat die Form:

$$F\big(x, y(x), Dy(x), D^2y(x), D^3y(x)\big) = 0$$

Die gesuchte Funktion $y(x)$ wird durch folgendes Polynom an den Knotenpunkten approximiert:

$$y \cong N_0 + \sum_{k=0}^{n} h_k N_k$$

Der Faktor N_k aus dem Polynom wird Formfaktor genannt und wird dem Problem entsprechend gewählt. Die Formfunktion selbst ist oft ein Polynom und an sie werden in der Regel strenge Bedingungen geknüpft. Die Terme der Formfunktion werden h_k genannt. Bei dem Galerkin-Ansatz werden die h_k Terme so berechnet, dass das Polynom die gesuchte Funktion möglichst gut approximiert. Dabei soll der Fehler R, das sogenannte Residuum, an den Knotenpunkten minimiert werden. Der Betrag des Residuums soll dabei am Knotenpunkt i gegen Null gehen.

$$R = \left| \left(N_0 + \sum_{k=0}^{n} h_k N_k \right) - y \right|$$

Oft wird eine zusätzliche Gewichtsfunktion W_i benutzt, um eine numerische Stabilität zu erreichen, das heißt W_i sorgt dafür, dass das Residuum am Knotenpunkt i nur vom Knotenpunkt i abhängt. Das Residuum wird an jedem Knotenpunkt solange minimiert, bis R kleiner als ein vorgegebenes Epsilon ist. Durch weitere Umformungen kommt eine Gleichung der folgenden Form zustande:

$$Ah + Bh' = 0$$

In dieser Gleichung ist A die Gesamtmatrix und B eine Korrektur-Matrix, die die numerische Eigenschaft verbessert. h entspricht dabei der Formfunktion N aus obiger Gleichung und h' entspricht der ersten Ableitung der Funktion h.

Die Bücher [18], [19] und [17] geben einen umfassenden Einblick in das Thema der FEM und deren genaue Lösungsverfahren.

2.4.2.1 Theorie der porösen Medien

Dieser Abschnitt basiert im Wesentlichen auf [20]. Andere Quellen werden separat angegeben.

Unter einem porösen Medium versteht man ein Festkörper-Skelett, dessen Poren mit einem Fluid oder einem Gas gefüllt sind. Bei vielen Problemen aus der Mechanik werden die Werkstoffe häufig als Ein-Phasen-Kontinuum behandelt, auch wenn diese aus mehreren Phasen bestehen. Da die Nichtlinearitäten der verschiedenen Phasen oft vernachlässigbar sind, liefert diese Vereinfachung in der Regel zufriedenstellende Ergebnisse. Bei einem porösen Medium sind die Kopplungseffekte zwischen den Phasen zu groß, wodurch diese Vereinfachung nicht mehr zulässig ist. [21]

Hierbei ergeben sich zwei Lösungswege. Bei der ersten Methode wird in dem porösen Medium jeder Teilkörper durch ein Einphasenmaterial und seiner Bewegung beschrieben. Zusätzlich müssen an den Innenseiten des Gesamtkörpers Kopplungsmechanismen, in Form von Übergangs- und Kontaktbedingungen, formuliert werden. Abbildung 13 zeigt diesen Ansatz schematisch. Da die Geometrie der inneren Porenstrukturen sowie die äußere Geometrie der Einzelkörper unbekannt sind, lässt sich diese Methode in den meisten Fällen nicht anwenden.

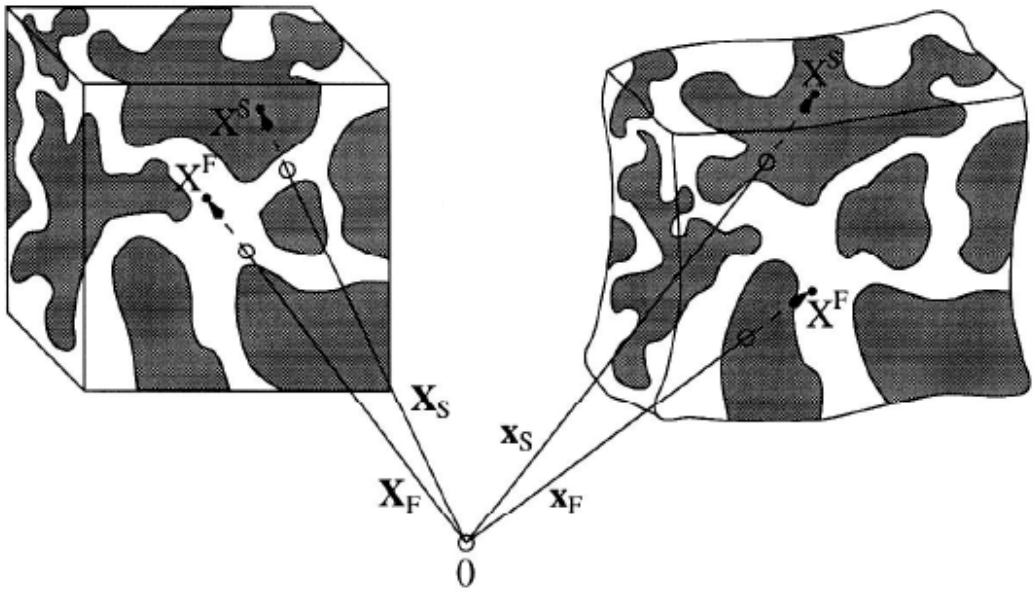

Abbildung 13: Gekoppeltes Festkörper-Fluid-Problem mit individuellen Bewegungsfunktionen der Teilkörper aus [20]

Bei der zweiten Methode wird ein statistischer Durchschnitt der Teilkörper eines betrachteten Bereichs gebildet. Hierbei muss beachtet werden, dass der betrachtete Bereich für eine statistische Aussage groß genug ist und, dass die lokalen Substrukturen für eine verschmierte Modellbildung fein genug sind. Dadurch erhält man ein Kontinuum mit statistisch verteilten und unvermischbaren Konstituierenden, welche sich gleichzeitig im gemeinsamen Kontrollraum befinden. Abbildung 14 zeigt exemplarisch eine solche Durchschnittsbildung. Dadurch lassen sich mithilfe der Mischungstheorie interne Kopplungsmechanismen zwischen den Phasen beschreiben. Um dieser Beschreibung von allgemeinen Mehrphasenmaterialien ein volumetrisches Maß hinzuzufügen, wird von dem Konzept der Volumenanteile Gebrauch gemacht.

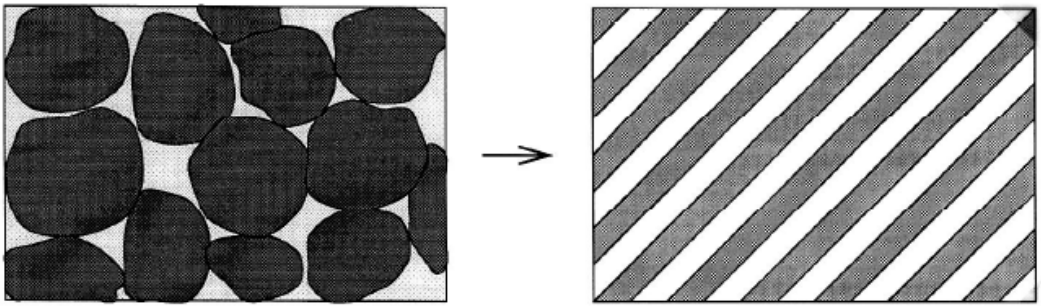

Abbildung 14: Durchschnittsbildung eines fluidgesättigten granularen Festkörpers aus [20]

In [20] wird die Theorie der porösen Medien wie folgt definiert:

> *Damit ist die Theorie der porösen Medien definiert als eine Kontinuumstheorie für mehrphasige Materialien, die sich aus den Elementen „Mischungstheorie“ und „Konzept der Volumenanteile“ zusammensetzt.*

Das Konsolidationsproblem ist ein klassischer Anwendungsfall für die Theorie der porösen Medien. Hierbei wird auf einem flüssigkeitsgesättigten Boden eine zusätzliche äußere Belastung aufgebracht. Diese Belastung kann zum Beispiel ein Bauwerk darstellen, welches über die Zeit zu Verformungen des Untergrunds führt. Die Verformung ist von einem Prozess austretendem Wassers begleitet. Dieser Konsolidierungsprozess ist beendet, wenn ein neuer Gleichgewichtszustand zwischen der inneren Spannung des Bodens und der äußeren Belastung eingetreten ist. Abbildung 15 zeigt eine schematische Darstellung des Konsolidationsproblems.

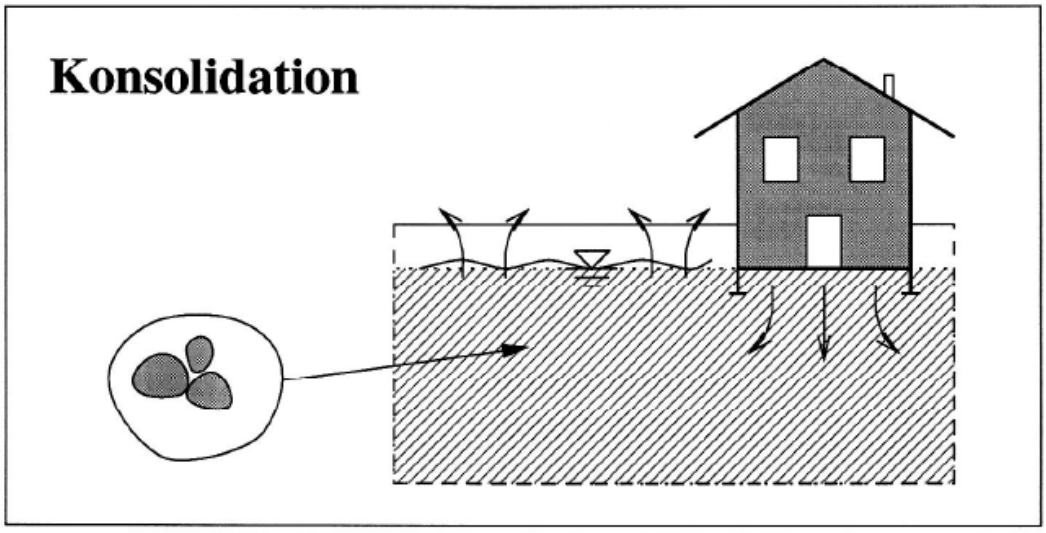

Abbildung 15: klassisches Konsolidationsproblem aus [20]

Für die mathematischen Konzepte hinter der TPM sei auf [20] oder [22]verwiesen, da dies den Rahmen dieser Arbeit übersteigt.

2.5 Strukturänderungen im Knochen

Im Rahmen des Exzellenzclusters SimTech werden Strukturänderungen bei wechselnden Belastungen am menschlichen Knochen simuliert. Ein Knochen wird von einer ihm eng anliegenden Bindegewebshaut umgeben und besteht aus einer sogenannten Knochenmatrix. Diese Knochenmatrix besteht aus 70% anorganischen Stoffen, 20% organischen Materialien und aus 10% Wasser. Somit lässt sich die Knochenmatrix mit der Theorie der porösen Medien gut beschreiben. Ein Knochen ist dem ständigen Umbau unterworfen, indem Osteoblasten den Knochen aufbauen und Osteoklasten den Knochen abbauen. Je nach Belastungsart und Belastungsdauer stellen sich unterschiedliche Gleichgewichte dieser Auf- und Abbauprozesse ein. Abbildung 16 zeigt beispielsweise wechselnde Belastungen beim Rudern am Oberschenkelhalsknochen.

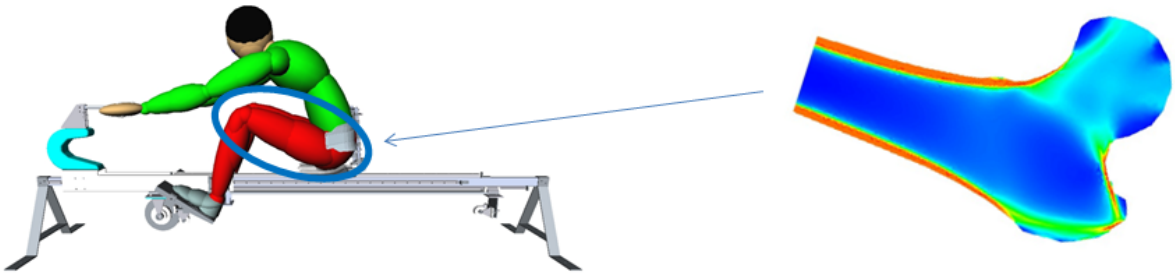

Abbildung 16: wechselnde Belastungen am Oberschenkelhalsknochen

Durch die Aufbau- und Abbauprozesse, kann sich der Knochen innerhalb weniger Wochen an eine geänderte Belastung anpassen. Die Fähigkeit diese Strukturänderungen am menschlichen Knochen simulieren zu können, ist aus medizinischer Sicht recht interessant. Denn hierdurch lassen sich beispielsweise nach einer Verletzung Reha-Maßnahmen festlegen oder die Trainingsmethoden bei Langzeitaufenthalten von Astronauten auf einer Raumstation, deren Knochen in der Schwerelosigkeit kaum bis keine Belastungen erfahren, optimieren.

So eine Simulation ist recht komplex und stellt eine Multi-* Simulation dar. Eine Multi-* Simulation deckt mindestens zwei folgender Bereiche ab:

Multi-Domänen: Die hier angewendeten Methoden stammen aus verschiedenen Fachbereichen. Zum Beispiel finden Modelle aus der Mechanik und der Biologie Anwendung.

Multi-Skalen: Für die Simulation werden unterschiedliche Zeit- und /oder Raumskalen verwendet.

Multi-Physiken: In der Simulation kommen unterschiedliche physikalische Gesetze zum Einsatz.

Multi-Tools: Für die Simulation kommen unterschiedliche Tools zum Einsatz.

Ein besonderer Schwerpunkt der Strukturänderungssimulation im Knochen stellt der Multi-Skalen Bereich dar. Neben der unterschiedlichen Zeitskala, bedingt durch Belastungsintervalle und

Ruhephasen, ist die Raumskala von bedeutender Rolle. Um das Skelett bzw. den Knochen zu simulieren, muss eine ganze Kaskade von Simulationen unterschiedlicher Modelle mit unterschiedlichen Raumskalen durchgeführt werden, welche sich gegenseitig bedingen. Abbildung 17 veranschaulicht diese Simulationskaskade, welche von der Zell-Ebene über die Gewebs- und Organ-Ebene zur Skelett-Ebene verläuft.

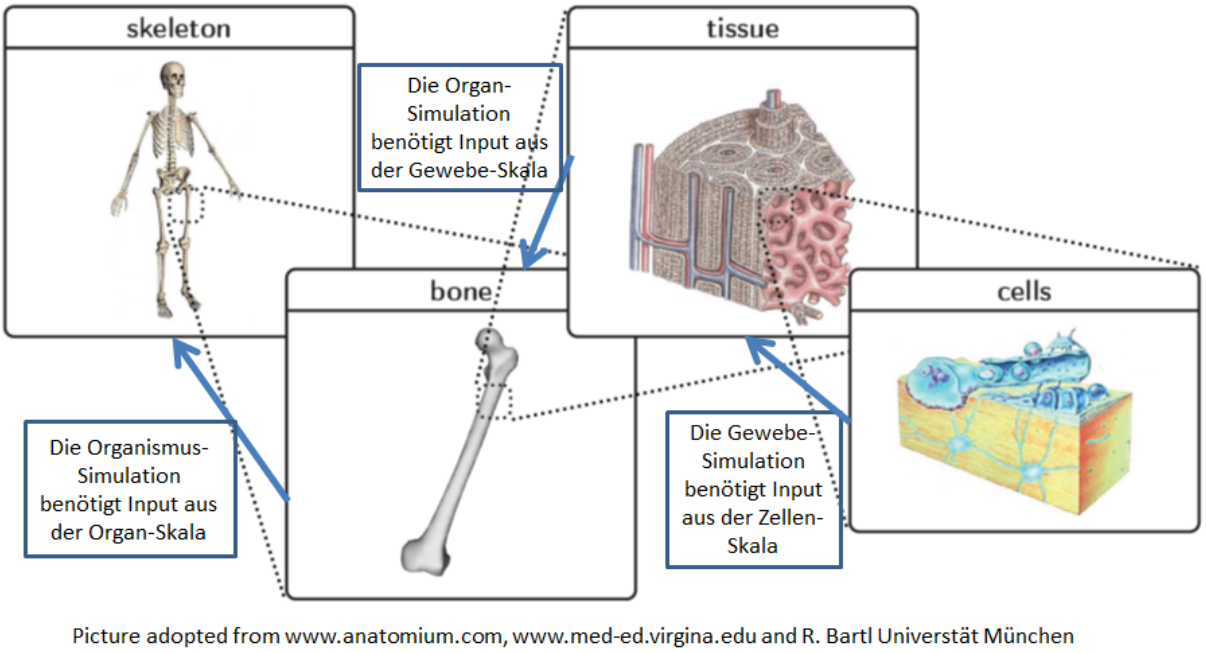

Abbildung 17: Multi-Skalen Simulation aus [23]

Die Strukturänderungen im Knochen werden über die Simulationsprogramme Pandas und Matlab ausgeführt. Pandas simuliert hierbei das biomechanische Knochenmodell auf einer größeren Raumskala und Matlab simuliert das biologische Modell auf einer kleineren Raumskala. Neben der unterschiedlichen Zeitskala der Belastungs- und Ruhephase, laufen die Modelle selbst in unterschiedlichen Zeitskalen ab. Das biomechanische Modell besitzt eine große Zeitskala, welche beispielsweise in Tagen abläuft, und das biologische Modell hat eine kleinere Zeitskala, die zum Beispiel in Minuten oder Stunden abläuft. Zunächst wird in Pandas ein Knochenmodell erstellt, welches auf Abbildung 18 zu sehen ist.

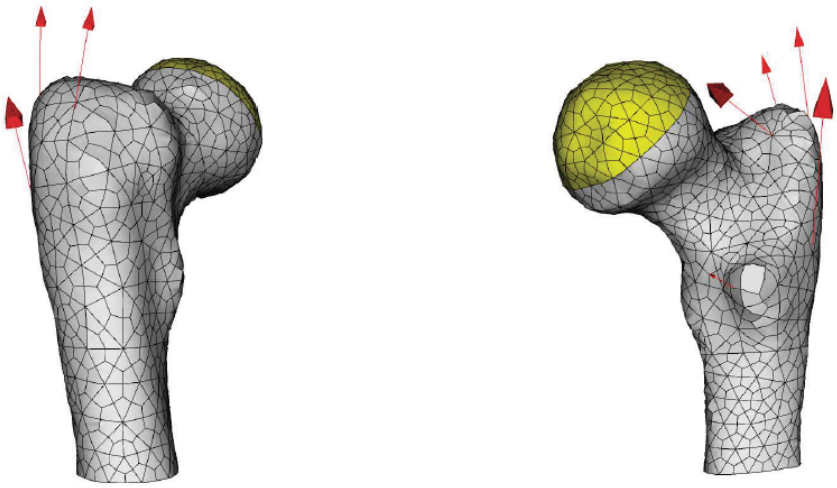

Abbildung 18: Finite Elemente Netz und Randbedingungen aus [23]

Nachdem das Knochenmodell erzeugt wurde, können die Auswirkungen der mechanischen Belastungen berechnet werden. Teilergebnisse von der Berechnung aus der größeren Raumskala, welche die Belastungsbedingungen an verschiedenen Punkten im Knochenmodell beschreiben, müssen dann der Simulation auf der kleineren Raumskala bereitgestellt werden. Mit diesen Daten kann dann das Verhalten auf Zell-Ebene simuliert werden. Die Ergebnisse der Simulation aus der kleineren Raumskala, welche beschreiben, ob an den verschiedenen Punkten im Knochenmodell Knochenstrukturen auf- oder abgebaut wurden, müssen dann wieder der Simulation auf der größeren Raumskala zur Verfügung gestellt werden. Dieser Zyklus beschreibt einen Berechnungsschritt der Gesamtsimulation und wird so oft wiederholt bis ein angestrebter Zeitpunkt erreicht wurde. Abbildung 19 zeigt eine Ergebnisreihe zeitlicher und räumlicher Veränderungen im Knochen.

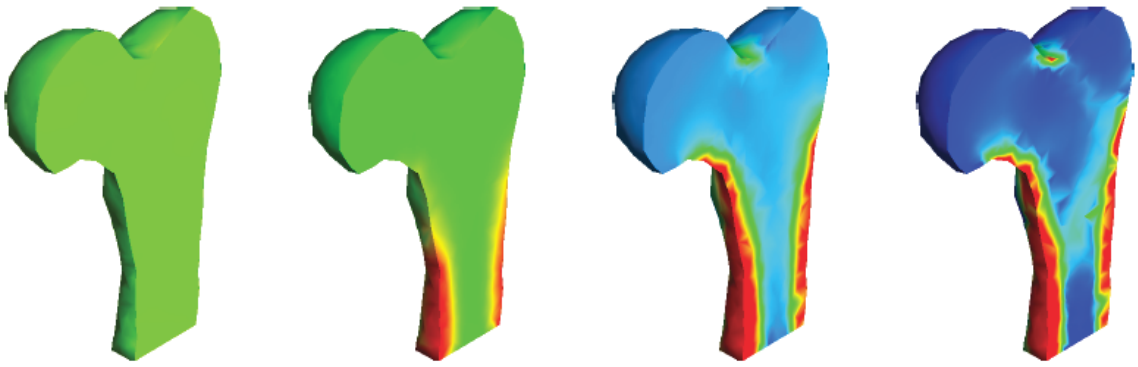

Abbildung 19: Ergebnisreihe zeitlicher und räumlicher Veränderungen im Knochen aus [23]

3 Spezifikation

In diesem Kapitel wird die Pandas Service-Bus-Erweiterung näher spezifiziert. Zunächst wird auf die Anforderungen und die verwendeten Konzepte eingegangen, danach folgen die Anwendungsfälle der Pandas Service-Bus-Erweiterung.

3.1 Anforderungen

Bereitstellung von Pandas als Web Service **(𝔸1)**

Pandas soll als Web Service bereitgestellt werden, worüber dessen Steuerung möglich ist. Hierbei soll Java als Programmiersprache verwendet werden, und die Web Services kommunizieren mit Hilfe von SOAP Nachrichten. Der Web Service soll des Weiteren unter eine geeigneten Opensource Lizenz gestellt werden.

Speicherung von Ergebnissen in einer Datenbank **(𝔸2)**

Pandas soll die Möglichkeit bieten (Teil-) Ergebnisse, wie zum Beispiel ein Finite Elemente Gitter oder die Lösungsmatrix, in einer Datenbank abzuspeichern. Hierzu soll ein Datenbank-Schema erarbeitet werden, um eine Simulation abzubilden.

Interaktion mit anderen Simulationsprogrammen **(𝔸3)**

Die Daten aus Pandas sollen von anderen Simulationsanwendungen verwendet werden können. Hierbei muss unter Umständen ein Transformator oder eine Schnittstelle für das Ein- und Auslesen der Daten bereitgestellt werden. Als Beispiel wäre hier die Verwendung eines externen DUNE-Löser oder die Kopplung von Pandas mit Matlab zu nennen.

Verwendung einer Opensource-Datenbank **(𝔸4)**

Im Rahmen dieser Arbeit soll eine Opensource Datenbank verwendet werden.

BPEL-Prozess für die Simulation des Knochenwachstums **(𝔸5)**

Im Rahmen dieser Arbeit soll ein BPEL-Prozess erstellt werden, der den Pandas Web Service verwendet und eine Simulation des Knochenwachstums durchführt.

3.2 Konzepte

Für die Realisierung der Service-Bus-Erweiterung von Pandas, finden verschiedene Softwarelösungen eine Verwendung. Ein Überblick darüber ist auf Abbildung 20 zu sehen und wird anschließen weiter erläutert.

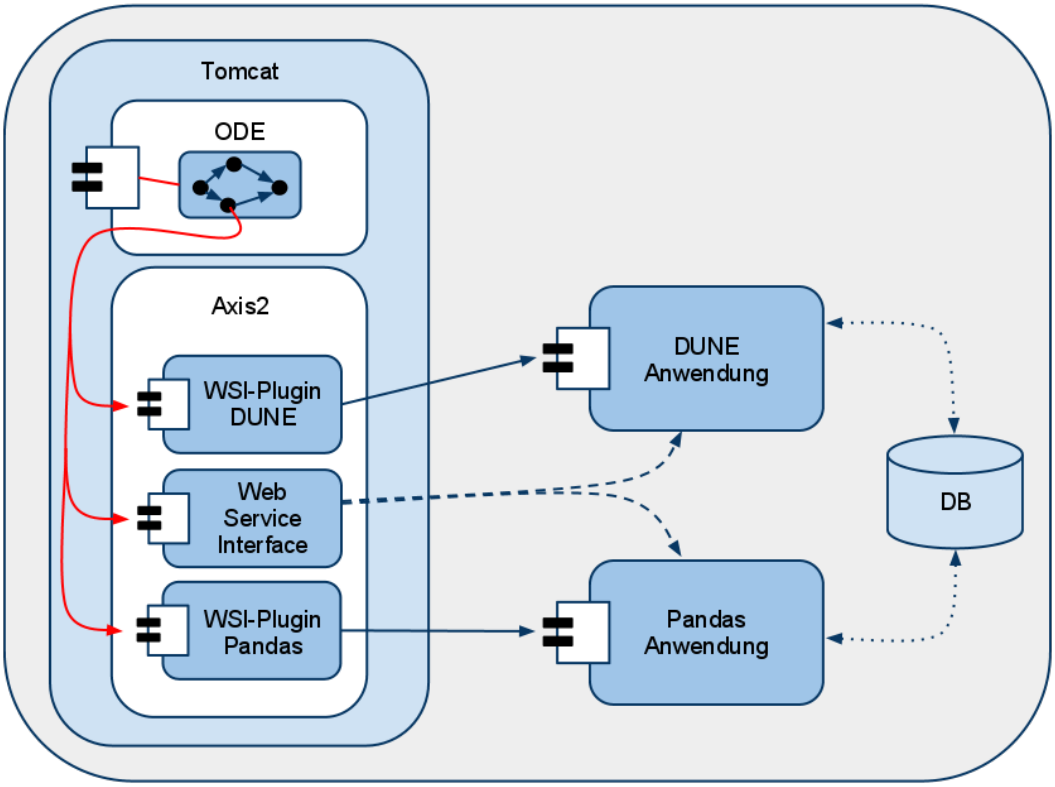

Abbildung 20: Überblick Service-Bus-Erweiterung

Hierbei bekommt Pandas eine direkte Datenbankanbindung über die OpenDBX-Bibliothek[6] zur Speicherung und zum Austausch von Daten mit anderen Simulationsanwendungen. Hierbei wird eine Opensource Datenbank verwendet. Die Fähigkeit Web Service Anfragen verarbeiten zu können bekommt Pandas über das gSOAP Toolkit. Da Pandas für jede Simulation neu kompiliert und gestartet werden muss, kann es hierüber nicht als Dienst verwendet werden. Für die Funktionalität als Dienst wird ein Pandas-Adapter erstellt, der auf dem Web Service Interface aus der Diplomarbeit [24] aufbaut. Diese Adapter laufen auf einem Apache Tomcat Server in einem Web Container. Als Workflow Engine kommt Apache ODE zum Einsatz, welche auch auf dem Tomcat Server läuft. Damit kann der Pandas-Adapter von einem Client oder aus einem Workflow heraus aufgerufen werden, der wiederum Pandas kompiliert und startet.

[6] http://www.linuxnetworks.de/doc/index.php/OpenDBX

3.2.1 gSOAP

Um Pandas mit einer Web Service Schnittstelle zu versehen, wird das gSOAP Web Service-Toolkit[7] verwendet. Damit wird ein Binding von SOAP und XML-Typen auf C-Funktionen ermöglicht. Hierbei werden die Web Service spezifischen Teile in Stubs und Skeletons gekapselt, welche dann von Pandas verwendet werden können. gSOAP stellt eine Web Service-Server Bibliothek zur Verfügung, welche direkt in Pandas mit eingebunden und kompiliert werden kann.

Für die Erzeugung der Stubs und Skeletons wird sowohl ein Top-Down wie ein Bottom-Up Ansatz zur Verfügung gestellt. Bei beiden Ansätzen ist eine spezielle Schnittstellen-Definition nötig – die sogenannte gSOAP-Header File. Diese Header File entspricht einer normalen C/C++ Header File, die jedoch mit zusätzlichen Informationen in den Kommentaren angereichert ist. Darin stehen Informationen wie z.B. der Name des Web Services oder der Typ des Services. Bei dem Top-Down Ansatz wird diese spezielle Schnittstellen-Definition aus einer gegebenen WSDL-Datei erzeugt und daraus dann die Stubs und Skeletons. Bei dem Bottom-Up Ansatz hingegen muss die gSOAP-Header File schon vorhanden sein, woraus dann die WSDL-Datei sowie die Stubs und Skeletons erzeugt werden.

Damit ist Pandas zwar mit einer Web Service Schnittstelle versehen, lässt sich aber so nicht als Dienst, der ständig zu Verfügung steht, verwenden. Denn Pandas muss für jede Simulation neu mit dem zu simulierenden Problem kompiliert werden. Um diese Problematik zu umgehen, ist ein stellvertretender Adapter nötig, der Pandas kompilieren und starten kann.

3.2.2 Web Service Interface

Der stellvertretende Pandas Adapter baut auf dem erweiterungsfähigen Web Service Interface aus der Diplomarbeit [24] auf. Das Web Service Interface ist auf der Abbildung 21 zu sehen und wird im Folgenden näher beschrieben.

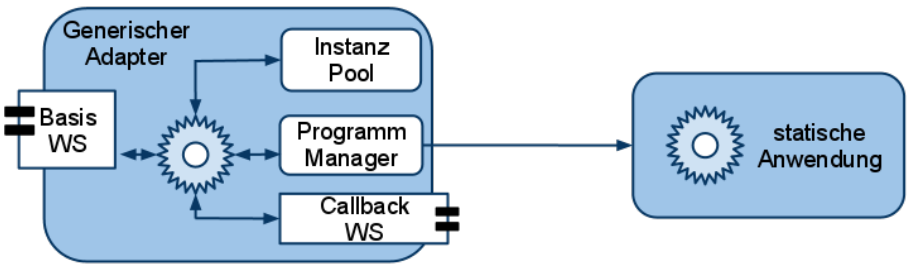

Abbildung 21: Web Service Interface nach [24]

[7] http://www.cs.fsu.edu/~engelen/soap.html

Das Web Service Interface, nachfolgend nur noch WSI genannt, stellt einen generischen Adapter zur Verwaltung von Simulationsinstanzen bereit.

Der Basis Web Service stellt die primäre Schnittstelle des Adapters dar und bietet Operationen für den generischen Umgang mit Simulationsinstanzen an. Dazu gehören z.B. Operationen zum Erzeugen bzw. Löschen von Instanzen oder das Ausführen von Anwendungen.

Der Instanz Pool verwaltet die einzelnen Simulationsinstanzen mit dessen Eigenschaften wie das Instanzverzeichnis, dem Zustand und dem Lebenszyklus. Der Lebenszyklus einer Simulationsinstanz wird auf der Abbildung 22 gezeigt.

Über den Programm-Manager können externe Programme ausgeführt werden oder Archive entpackt werden. Des Weiteren ist es möglich die Standard-Ausgabe der hierüber gestarteten Programme in einem Log zu speichern.

Der Callback-Web Service dient als Rückmeldungsschnittstelle für modifizierte Simulationsanwendungen. Hierüber wird dem generischen Adapter von der Simulationsanwendung der verwendete TCP-Port und die Bereitschaft der Bearbeitung von Web Service Anfragen mitgeteilt.

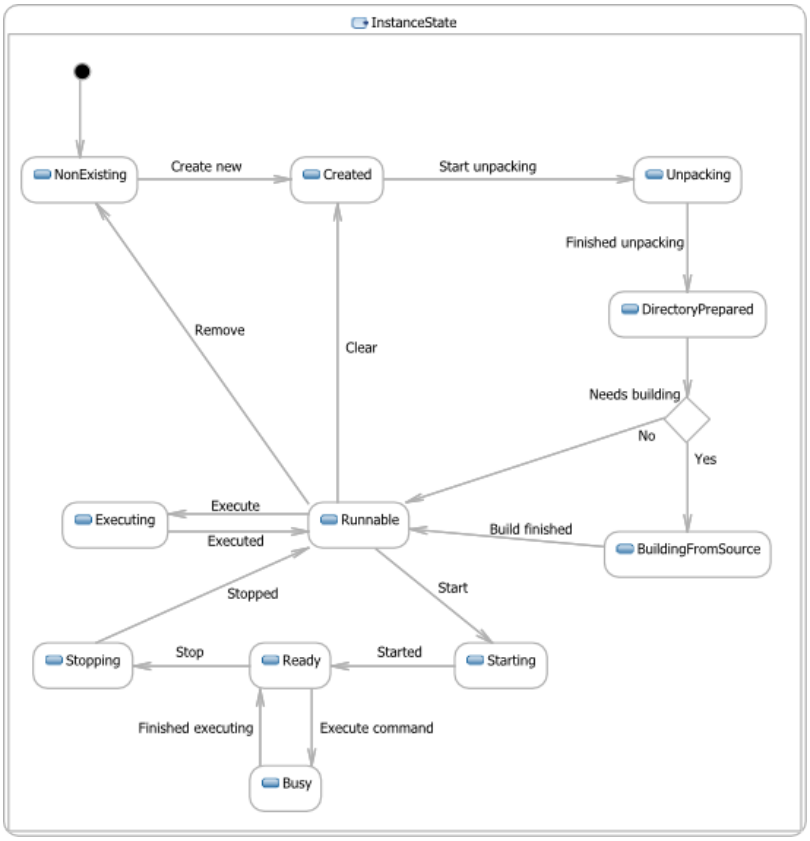

Abbildung 22: Zustände einer Simulationsinstanz aus [24]

3.2.3 OpenDBX

Da in Simulationen schnell große Datenmengen anfallen können, soll eine effiziente Datenspeicherung möglich sein. Damit diese Datenmengen nicht über SOAP-Nachrichten transportiert oder in die Workflow-Engine geladen werden müssen, wird die Datenbankanbindung direkt von Pandas aus gewährleistet. Hierfür wird die OpenDBX Bibliothek verwendet, da diese als C-Bibliothek einfach in das Programm mit eingebunden werden kann und eine Vielzahl an Datenbanken unterstützt.

3.2.4 Pandas Service-Bus-Adapter

Dieses Szenario stellt die Pandas Service-Bus-Erweiterung dar und baut auf dem WSI-Adapter auf. Durch diese Service-Bus-Erweiterung wird einerseits die Grundfunktionalität Pandas bereitgestellt und andererseits eine Datenbankanbindung ermöglicht. Der Akteur in den nachfolgenden Anwendungsfällen ist der Client des Pandas Service-Bus-Adapter. Abbildung 23 zeigt eine Übersicht der Anwendungsfälle der Pandas-Service-Bus-Erweiterung.

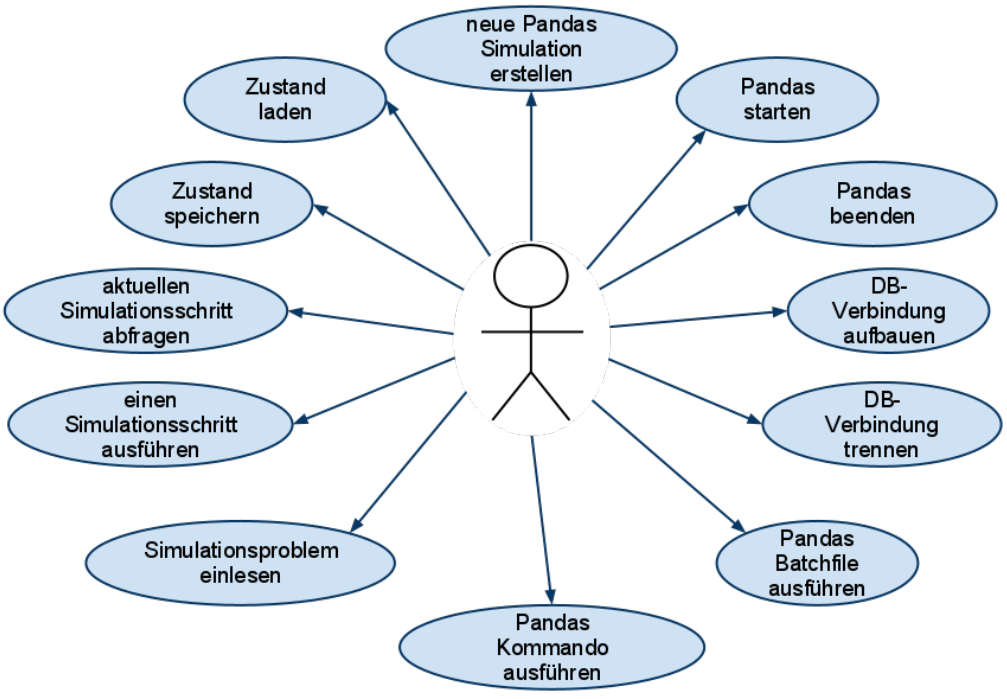

Abbildung 23 : Übersicht über die Pandas-Service-Bus-Erweiterung Anwendungsfälle

3.2.4.1 neue Pandas Simulation erstellen (A1)

Beschreibung: Damit man eine Pandas Simulation starten und steuern kann, muss zunächst eine neue Simulationsinstanz erzeugt werden. Hierzu wird zunächst über den generischen WSI-Adapter eine neue Instanz erzeugt. Mit der neuen Instanz wird eine neue Simulations-ID und ein neues Verzeichnis erstellt. In diese neue Simulationsinstanz wird anschließend der Pandas Sourcecode und ein Simulationsproblem entpackt und kompiliert. Der Instanzstatus wird auf *Runnable* gesetzt.

Vorbedingung: Im Archiv-Verzeichnis des generischen WSI-Adapters muss das Unterverzeichnis `src` existieren und darin das Pandas-Sourcecode-Archiv `pandas.tar.gz`. Zudem muss in dem Archiv-Verzeichnis das Unterverzeichnis `problems` existieren und darin das Simulationsproblem-Archiv.

Nachbedingung: Die Simulationsinstanz wurde erzeugt, die Pandas Archive wurden darin entpackt und anschließend kompiliert. Die Simulationsinstanz ist in einem ausführbaren Zustand.

Fehler:

- Archive-Files existieren nicht.
- Schreibfehler, volles Filesystem.

Ablauf:

1. Es wird eine neue Simulationsinstanz erzeugt.
2. Der Pandas Sourcecode wird entpackt.
3. Das Pandas Simulationsproblem wird entpackt.
4. Der Instanz Status wird auf *DirectoryPrepared* gesetzt.
5. Der Instanz Status wird auf *Runnable* gesetzt.
6. Der Pandas Sourcecode wird mit dem Simulationsproblem kompiliert.

3.2.4.2 Pandas starten (A1)

Beschreibung: Es wird von einer bestehenden Pandas-Simulationsinstanz die Pandas-Anwendung gestartet.

Vorbedingung: Die Pandas-Simulationsinstanz wurde erfolgreich erzeugt und Pandas wurde darin erfolgreich kompiliert.

Nachbedingung: Pandas ist gestartet.

Fehler:

- Pandas ist nicht erfolgreich kompiliert.
- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Der angegebene Pfad zu den Pandas-Binaries existiert nicht.

Ablauf:

1. Pandas wird gestartet.

2. Der integrierte Pandas Web Service-Server wird gestartet.
3. Pandas wartet auf Befehle.

3.2.4.3 Pandas beenden (A1)

Beschreibung: Es wird eine gestartete Pandas-Simulationsinstanz beendet.

Vorbedingung: Pandas wurde gestartet.

Nachbedingung: Pandas ist beendet.

Fehler:

- Pandas ist nicht gestartet.
- Die angegebene Pandas-Simulationsinstanz existiert nicht.

Ablauf:

1. Eine eventuell bestehende Datenbank-Verbindung wird getrennt.
2. Der integrierte Pandas Web Service-Server wird heruntergefahren.
3. Pandas wird beendet.

3.2.4.4 DB-Verbindung aufbauen (A1, A2)

Beschreibung: Bei einer gestarteten Pandas-Simulationsinstanz wird eine Verbindung zu einer Datenbank hergestellt.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und wartet auf Anfragen, und es besteht keine Datenbank-Verbindung.

Nachbedingung: Pandas ist mit der angegebenen Datenbank verbunden.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Die angegebene Datenbank existiert nicht.
- Die Datenbank-Verbindungsdaten sind nicht korrekt.

Ablauf:

1. Es wird geprüft, ob schon eine Verbindung besteht.
2. Verbindung wird hergestellt.

3.2.4.5 DB-Verbindung trennen (A1, A2)

Beschreibung: Bei einer gestarteten Pandas-Simulationsinstanz, die eine Datenbank-Verbindung hat, wird diese von Pandas getrennt.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden.

Nachbedingung: Pandas hat die Datenbank-Verbindung getrennt.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.

Ablauf:

1. Es wird geprüft ob eine Verbindung besteht.
2. Wenn eine Verbindung besteht, wird diese getrennt.

3.2.4.6 Pandas Batchfile ausführen (A1)

Beschreibung: Bevor eine Simulation ausgeführt werden kann, muss Pandas zuvor entsprechend initialisiert werden. Dies geschieht in der Regel über eine Pandas Batchfile, welche eine Reihe von Pandas Kommandos beinhaltet. Hierüber wird Pandas veranlasst eine solche Batchfile auszuführen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Zudem sollte die Batchfile in dem Verzeichnis liegen, in dem die Pandas-Binaries liegen.

Nachbedingung: Es wurde die Pandas Batchfile ausgeführt.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Die angegebene Batchfile existiert nicht.

Ablauf:

1. Es wird das Kommando zum Ausführen einer Batchfile an Pandas geschickt.
2. Pandas führt die Batchfile aus.

3.2.4.7 Simulationsproblem einlesen (A1)

Beschreibung: Bevor eine Simulation ausgeführt werden kann, muss Pandas zuvor initialisiert werden. Dabei ist das Einlesen des Simulationsproblems ein Teil der Initialisierung. Hierbei werden z.B. Dateien eingelesen, welche die Geometrie oder die Materialeigenschaften des Problems beschreiben. Es werden dabei vier unterschiedliche Möglichkeiten des Einlesens unterstützt.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden.

Nachbedingung: Das Simulationsproblem wurde eingelesen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Die angegebenen Problem-Files existieren nicht.

Ablauf:

Problem wird über Mesh-Generator eingelesen:

1. Shape-Datei wird eingelesen.
2. Geometrie-Datei wird eingelesen.
3. Netzgenerator wird eingestellt.
4. Datei der internen Variablen wird eingelesen.
5. Materialparameter-Datei wird eingelesen.

Problem wird über Grid-Generator eingelesen:

1. Shape-Datei wird eingelesen.
2. Geometrie-Datei wird eingelesen.
3. Gittergenerator wird gestartet.
4. Datei der internen Variablen wird eingelesen.
5. Materialparameter-Datei wird eingelesen.

Problem wird über Nodes und Elements File eingelesen:

1. Shape-Datei wird eingelesen.
2. Geometrie-Datei wird eingelesen.
3. Knoten-Datei wird eingelesen.
4. Elems-File wird eingelesen.
    a. Randbedingungs-Datei wird eingelesen, falls angegeben.
5. Datei der internen Variablen wird eingelesen.
6. Materialparameter-Datei wird eingelesen.

Problem wird über eine Problembeschreibungsdatei eingelesen:

1. Problembeschreibungsdatei wird eingelesen.

3.2.4.8 Pandas Kommando ausführen (A1)

Beschreibung: Hierüber kann jedes Pandas Kommando zur Steuerung ausgeführt werden

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden.

Nachbedingung: Pandas hat das Kommando ausgeführt.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Das angegebene Kommando existiert nicht.

Ablauf:

1. Es wird das Kommando an Pandas abgeschickt.
2. Pandas führt dieses Kommando aus.

3.2.4.9 Einen Simulationszeitschritt ausführen (A1)

Beschreibung: Pandas wird veranlasst einen Simulationszeitschritt auszuführen. Hierfür wird das Kommando **`step 1`** ausgeführt.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein.

Nachbedingung: Pandas hat einen Simulationszeitschritt ausgeführt

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas wurde nicht initialisiert.
- Pandas ist nicht mit der Datenbank verbunden.

Ablauf:

1. Es wird das Kommando **`step 1`** an Pandas abgeschickt.
2. Pandas führt einen Simulationszeitschritt aus.

3.2.4.10 Aktuellen Simulationszeitschritt abfragen *(A1)*

Beschreibung: Zur Ablaufsteuerung kann es nötig sein den aktuellen Simulationszeitschritt abzufragen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein.

Nachbedingung: Der aktuelle Simulationszeitschritt wurde zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas wurde nicht initialisiert.
- Pandas ist nicht mit der Datenbank verbunden.

Ablauf:

1. Der aktuelle Simulationszeitschritt wurde zurückgeliefert.

3.2.4.11 Zustand speichern *(A1, A2)*

Beschreibung: Der Zustand von Pandas soll in der Datenbank abgespeichert werden. Um den Zustand zu erzeugen, wird das Pandas Kommando **`save`** verwendet.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein.

Nachbedingung: Pandas hat seinen Zustand in der Datenbank gespeichert

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas wurde nicht initialisiert.
- Pandas ist nicht mit der Datenbank verbunden.

Ablauf:

1. Der Pandas Zustand wird über das Kommando **`save`** erzeugt.
2. Die entstandene Zustandsdatei wird in der Datenbank gespeichert.
3. Die Zustandsdatei wird gelöscht.

3.2.4.12 Zustand laden *(A1, A2)*

Beschreibung: Ein zuvor in der Datenbank gespeicherter Pandas Zustand soll von der aktuellen Pandas Instanz geladen werden. Um den Zustand zu laden, wird das Pandas Kommando **`load`** verwendet.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt. Pandas ist gestartet und mit einer Datenbank verbunden.

Nachbedingung: Pandas hat den Zustand geladen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Der angegebene Zustand existiert nicht.

Ablauf:

1. Die Zustandsdatei wird aus der Datenbank geholt.
2. Der Pandas Zustand wird über das Kommando **`load`** aus der Zustandsdatei geladen.
3. Die Zustandsdatei wird gelöscht.

3.2.5 Evaluierung der Datenbank

Im Rahmen der Diplomarbeit [25] wurden verschiedene Kriterien und Testprozesse zur Datenbankevaluierung aufgestellt. Darin wurden Hauptspeicher und Extern-Speicher-basierte Datenbanken auf ihre relationalen bzw. XML Fähigkeiten untersucht. Für diese Arbeit soll die relationale Datenbank PostgreSQL mit den Ergebnissen der relationalen Extern-Speicher-basierten Datenbanken aus der Arbeit [25] verglichen werden. Nachfolgend werden die 11 Kriterien für die Datenbank-Evaluierung kurz erläutert.

3.2.5.1 Kriterien

Kriterium 1: Lizenz und Kosten

Das erste Kriterium betrachtet die Lizenz der Datenbank. Handelt es sich um eine Opensource oder kommerzielle Datenbank? Wenn es sich um eine Opensource Lizenz handelt, gibt es copyleft[8] Bedingungen? Wie gestaltet sich der Preis bei einer kommerziellen Lizenz und gibt es für den wissenschaftlichen Gebrauch spezielle Angebote?

[8] Eine copyleft-Bedingung schreibt fest, dass Bearbeitungen des Werks nur dann erlaubt sind, wenn alle Änderungen ausschließlich unter den identischen oder im Wesentlichen gleichen Lizenzbedingungen weitergegeben werden

Kriterium 2: Plattform

Bei dem zweiten Kriterium werden die unterstützten Plattformen der Datenbank betrachtet. Für welche Betriebssysteme wird die Datenbank angeboten, und gibt es 32 bit und 64 bit Versionen? Gibt es besondere Anforderungen an die Hardware? Ebenso wird gefragt, ob JDBC-Treiber angeboten werden.

Kriterium 3: Installation und Integration

Das dritte Kriterium beleuchtet die Installation und Integration der Datenbank. Ist eine Installationsanleitung vorhanden, und wie aufwändig gestaltet sich die Installation? Wie lange dauert die Installation, und wie groß ist die Installation?

Kriterium 4: Administration und Benutzerfreundlichkeit

Bei dem vierten Kriterium werden die Administration und die Benutzerfreundlichkeit bewertet. Ist eine Anleitung bzw. eine Dokumentation vorhanden, und wenn ja, wie ausführlich ist diese? Sind Administrationswerkzeuge oder ein Web Front-end vorhanden?

Kriterium 5: SQL-Features

Das fünfte Kriterium beleuchtet die vorhandenen SQL-Features einer relationalen Datenbank. Es wird untersucht, ob alle SQL-Sprachelemente unterstützt werden und ob die Standard SQL Datentypen unterstützt werden. Ebenso wird geprüft, ob eine bestimmte Version des SQL-Standards eingehalten wird.

Kriterium 6: XQuery

Bei dem sechsten Kriterium findet nur bei Datenbanken die XQuery-Anfragen beherrschen Anwendung. Hierbei wird unter anderem untersucht, ob XML-Dokumente importiert und manipuliert werden können oder ob FLOWR-Ausdrücke verarbeitet werden können.

Kriterium 7: Transaktionen und ACID

Das siebte Kriterium untersucht die Datenbank auf ihre ACID-Fähigkeiten und Locking-Techniken.

Kriterium 8: Cache Consistency

Bei dem achten Kriterium wird die Synchronisierung zwischen lokalen Cache und Data Store untersucht.

Kriterium 9: Performance relationale DB

Das neunte Kriterium testet die Performance einer relationale Datenbank. Hierbei werden verschiedene Testqueries an die Datenbank gestellt und dabei die Ausführungszeit festgehalten.

Kriterium 10: Performance XML DB

Bei dem zehnten Kriterium wird die Performance einer XML-Datenbank getestet. Hierbei wird der XMark Test verwendet.

Kriterium 11: Verwendung als ODE Modell

Bei dem elften Kriterium wird die Datenbank als ODE Model DB verwendet. Dabei wird anschließend ein Workflow ausgeführt, der einen genetischen Algorithmus verwendet, um die Charakteristika von Simulationsworkflows zu simulieren.

3.2.5.2 Evaluierung

Im Rahmen dieser Arbeit soll eine Opensource Datenbank zum Einsatz kommen. Daher beschränkt sich das Auswahlkriterium auf das Performancekriterium für relationale Datenbanken.

In dem Performancetest für relationale Datenbanken gibt es eine Liste von Testqueries, die die Datenbank ausführen muss. Die Testqueries lassen sich in mehrere Test-Kategorien aufteilen:

[TPCH]: Die erste Test-Kategorie basiert auf den TPC-H Benchmark der Version 2.8.0. Hierbei werden typische Queries aus dem Businessumfeld und ad-hoc Queries abgedeckt. Zusätzlich zu den Queries werden auch die Datenbanktabellen und deren Inhalt bereitgestellt.

[Concurrent]: Bei dieser Test-Kategorie werden mehrere simultane Zugriffe auf eine Datenbanktablle simuliert. Hierbei gibt es wieder verschiedene Zugriffsarten: „read only“, „read and write“ und „insert only“.

[CO2]: Diese Test-Kategorie basiert auf Daten eines FEM-Grid, welches von dem Workshop on Numerical Models for Carbon Dioxide Storage in Geological Formations[9] bereitgestellt wurde.

Schema Dropping: Hier werden alle Tabellen und ihre Daten gelöscht.

3.2.5.3 Ergebnisse der Datenbank-Evaluierung (A4)

In der Tabelle 1 ist das wesentliche Ergebnis des Performance-Tests zu sehen. Die komplette Tabelle mit allen Ergebnissen und allen Datenbanken ist auf der Abgabe-DVD zu finden.

In der Arbeit von [25] schneidet die DB2 Datenbank bei den Tests für relationale Datenbanken am besten ab. Da in dieser Arbeit eine Opensource Datenbank zum Einsatz kommen soll, wird die DB2 nur als Referenzdatenbank aufgeführt. Das Ergebnis der drei gegenübergestellten Datenbanken ist farblich kodiert:

- Grün : Bestes Testergebnis
- Rot : Schlechtestes Testergebnis
- Schwarz: Test konnte nicht ausgeführt werden

Aus diesem Performance-Test-Ergebnis ist klar ersichtlich, dass die PostgreSQL Datenbank gegenüber der Apache Derby Datenbank in den meisten Testfällen besser abschneidet. Die Apache Derby

[9] http://www.iws.uni-stuttgart.de/co2-workshop/

Datenbank konnte sogar einen Testfall nicht ausführen. Aus diesem Grunde wird im Rahmen dieser Arbeit die PostgreSQL Datenbank verwendet.

Test Group	DB2	PostgreSQL	Derby
[TPCH] Import	00:02:08,313	00:06:08,035	00:06:50,254
[TPCH] Keys, Indizes	00:03:18,948	00:01:50,191	00:05:49,289
[TPCH] Select Count CUSTOMER	00:00:00,033	00:00:00,090	00:00:00,123
[TPCH] Select Count LINEITEM	00:00:00,153	00:00:00,363	00:00:00,587
[TPCH] Select Count Distinct ORDERS	00:00:00,341	00:00:02,565	00:00:08,741
[TPCH] Select * CUSTOMER	00:00:01,164	00:00:01,904	00:00:02,116
[TPCH] Select Column CUSTOMER	00:00:00,187	00:00:00,572	00:00:00,412
[TPCH] Select * Where ORDERS	00:00:09,712	00:00:10,185	00:00:18,954
[TPCH] Select Column Where ORDERS	00:00:08,219	00:00:02,596	00:00:05,310
[TPCH] Select Column Where ORDERS with Index	00:00:08,504	00:00:02,645	00:00:05,348
[TPCH] Select Join without Index	00:00:01,197	00:00:00,087	00:00:00,361
[TPCH] Select Join with Index	00:00:00,628	00:00:00,959	00:00:01,072
[TPCH] Select Aggregation	00:00:20,362	00:00:32,727	00:00:45,403
[TPCH] Select Aggregation Join Order	00:00:08,974	00:00:01,805	00:00:02,403
[TPCH] Query 2	00:00:01,261	00:00:01,515	00:00:05,952
[TPCH] Query 4 mod	00:00:09,044	00:00:00,803	00:00:08,113
[TPCH] Query 6	00:00:01,310	00:00:00,301	00:00:01,521
[TPCH] Query 16	00:00:01,169	00:00:07,808	00:00:07,084
[TPCH] Query 17	00:18:32,992	00:06:27,392	00:00:00,206
[TPCH] Query 22	00:00:13,464	01:12:19,852	
[CONCURRENT] Read Customer	00:00:05,773	00:00:07,816	00:00:07,126
[CONCURRENT] Modify Patient (Read Committed)	00:00:58,599	00:01:05,088	00:01:52,037
[CONCURRENT] Modify Patient (Repeatable Read)	00:01:06,533	00:01:02,407	00:01:16,595
[CONCURRENT] Insert Emergency Call (Read Committed)	00:10:34,808	00:03:28,880	00:03:36,529
[CONCURRENT] Insert Emergency Call (Repeatable Read)	00:09:51,093	00:03:24,632	00:03:37,466
[CO2] Import	00:00:12,921	00:00:09,071	00:00:16,945
[CO2] Keys	00:00:11,111	00:00:01,664	00:00:20,655
[CO2] Performance Adopt Properties	00:00:00,679	00:00:02,816	00:00:06,853
[CO2] Performance Select All	00:00:00,281	00:00:01,488	00:00:00,803
[CO2] Performance Join 1	00:00:00,469	00:00:01,210	00:00:36,602
[CO2] Performance Join 2	00:00:01,639	00:00:05,006	00:00:29,075
[CO2] Performance Join 3	00:00:09,630	00:00:06,338	00:00:45,048
Schema Dropping	00:00:03,829	00:00:01,122	00:00:03,459
RSS Memory Peak	786.444	432.520	706.320

Tabelle 1: Ergebnis Datenbank-Evaluierung

3.3 Erweiterungen des Pandas-Adapter

In diesem Abschnitt werden die Anwendungsfälle der verschiedenen Simulationsszenarien im Pandas-Umfeld beschrieben. Der Akteur in den nachfolgenden Anwendungsfällen der verschiedenen Szenarien ist der Client des entsprechenden WSI-Adapters.

3.3.1 DUNE-Matrixlöser

In diesem Szenario soll es ermöglicht werden, die Lösungsmatrix von Pandas extern zu speichern und von einem externen Löser zu lösen. Die Matrix wird dabei in der Datenbank abgelegt, die wiederum von einem Service mit einem DUNE-Löser geladen und gelöst wird. Die gelöste Matrix soll wieder in der Datenbank abgespeichert werden und von Pandas wieder geladen werden können. Abbildung 24 zeigt eine Übersicht über die Anwendungsfälle des externen DUNE-Lösers.

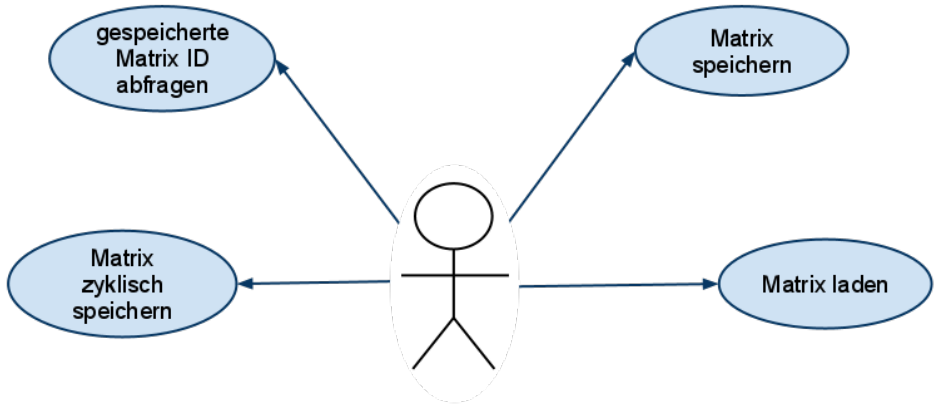

Abbildung 24: Übersicht über die PANDAS-DUNE Matrixlöser Anwendungsfälle

3.3.1.1 Gespeicherte Matrix-ID abfragen (A12, A13)

Beschreibung: Es soll die Matrix-ID einer zuvor abgespeicherten Lösungsmatrix von einer Simulation ausgegeben werden.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren sollte eine Matrix abgespeichert worden sein.

Nachbedingung: Die Matrix-ID wurde zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch keine Matrix abgespeichert.

Ablauf:

- Es wird die Matrix-ID zurückgeliefert.

3.3.1.2 Matrix speichern (A12, A13)

Beschreibung: Es wird die Lösungsmatrix von Pandas in der Datenbank abgelegt. Dabei kann die Speicherung werteweise oder der komplette Speicherbereich binär erfolgen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Die Matrix wurde in der Datenbank gespeichert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.

Ablauf:

1. Es wird geprüft, ob schon Metadaten über die Matrix abgespeichert wurden.
2. Wenn noch keine Metadaten abgespeichert wurden, so werden diese nun abgespeichert und ein neuer Simulationszeitschritt in der Datenbank angelegt.
3. Es wird die Matrixqualität erfasst.
4. Die Matrix wird gespeichert.
5. Es wird der Simulationszeitschritt, zu dem die Matrix gehört, zurückgegeben.

3.3.1.3 Matrix laden (A12, A13)

Beschreibung: Es wird die Lösungsmatrix aus der Datenbank in Pandas geladen. Dabei muss angegeben werden, in welcher Form die Matrix abgespeichert wurde – werteweise oder binär.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Matrix wurde in Pandas geladen

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.

- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.
- Die angegebene Matrix existiert nicht in der Datenbank.

Ablauf:

1. Es wird geprüft, ob eine Verbindung zur Datenbank besteht.
2. Bei bestehender Datenbank-Verbindung wird in Pandas die Matrix aus der Datenbank geladen.

3.3.1.4 Matrix zyklisch speichern (A2, A3)

Beschreibung: Es soll möglich sein, Pandas anzuweisen automatisch seine Matrix zyklisch mit fester Schrittweite in der Datenbank abzulegen. Dabei muss mit angegeben werden, wie die Speicherung zu erfolgen hat – werteweise oder binär.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt und Pandas ist gestartet.

Nachbedingung: Einstellungen wurden übernommen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.

Ablauf:

1. Die Speicherungsart wird übernommen.
2. Die Schrittweite zum Speichern wird übernommen.

3.3.2 Data-Quality

In diesem Szenario soll es Pandas ermöglicht werden, während der Simulation die Matrixqualität zu erfassen und in der Datenbank abzulegen. Bei der Matrixqualität sollen Metriken von den Hauptdiagonalen, den Vektoren und der Werteverteilung auf der Matrix erfasst werden. Ebenso soll es möglich sein, die verwendete Parameterfile zur Bestimmung der Materialeigenschaften abzufragen. Die erfassten Metriken können dann von externer Seite weiterverarbeitet werden, wie zum Beispiel dem Data-Quality-Framework aus der Diplomarbeit [26]. Diese Ergebnisse können dann zur Steuerung der Simulation verwendet werden. Abbildung 25 zeigt eine Übersicht über die Anwendungsfälle des Data-Quality Szenarios.

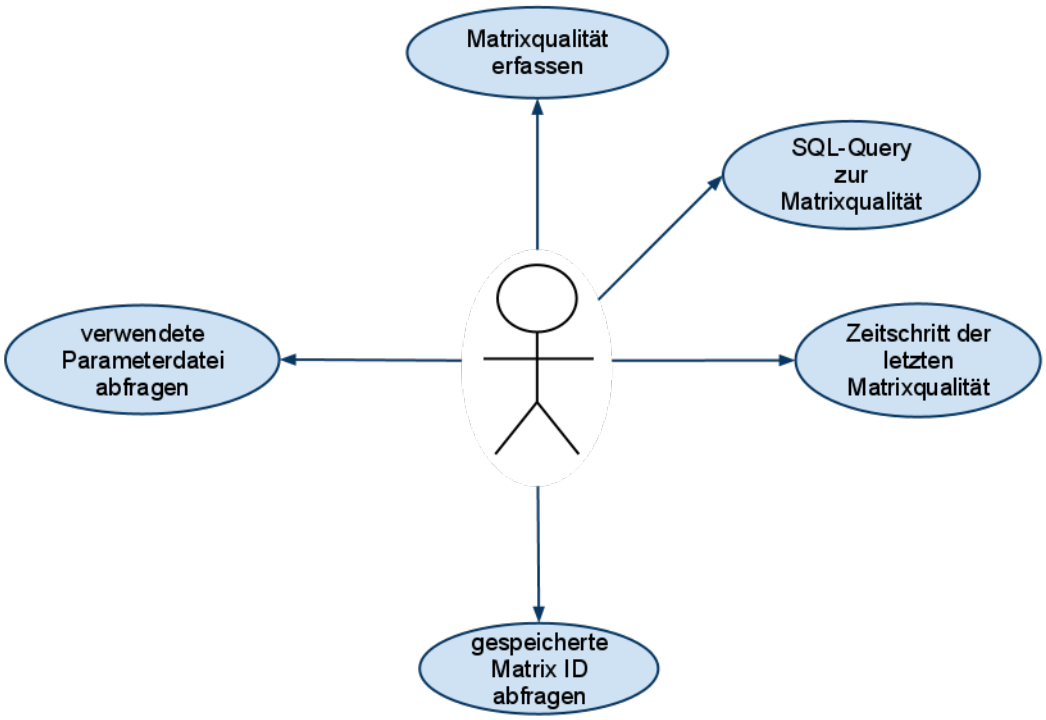

Abbildung 25: Übersicht über die PANDAS Data-Quality Anwendungsfälle

3.3.2.1 Matrixqualität erfassen (A12, A13)

Beschreibung: Es soll von der Matrix A sowie den Vektoren x und b, die bei der Gleichung $A * x = b$ auftreten, diverse Metriken erfasst werden. Dabei wird von der Matrix A die Anzahl der Null-Werte und der Nicht-Null-Werte gespeichert. Von der Hauptdiagonalen der Matrix A sowie den Vektoren x und b werden jeweils der minimale Wert, der maximale Wert und der Quotient aus dem minimalen und maximalen Wert in der Datenbank gespeichert.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Die Metrikwerte wurden in der Datenbank gespeichert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.

Ablauf:

1. Es wird geprüft, ob die Matrix vorhanden ist.

2. Wenn die Matrix vorhanden ist, werden alle Einträge durchlaufen, und es werden die Metriken dabei erfasst.
3. Es wird geprüft, ob eine Datenbank-Verbindung besteht.
4. Wenn eine Verbindung besteht, werden die erfassten Metriken gespeichert.
5. Es wird der Simulationszeitschritt, zu dem die Metriken gehören, gespeichert.
6. Es wird der Simulationszeitschritt, zu dem die Metriken gehören, zurückgegeben.

3.3.2.2 SQL-Query zur Matrixqualität (A13)

Beschreibung: Es soll ein SQL-Query zur zuletzt gespeicherten Matrixqualitätseintrag zurückgeliefert werden.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein, ein Simulationszeitschritt ausgeführt und eine Matrixqualität erfasst worden sein.

Nachbedingung: Die SQL-Query wurde zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde keine Matrixqualität erfasst.

Ablauf:

1. Es wird aus dem Simulationszeitschritt der zuletzt gespeicherten Matrixqualität und der Matrix-ID das SQL-Query zusammengesetzt.
2. Das SQL-Query wird zurückgegeben.

3.3.2.3 Zeitschritt der letzten Matrixqualität (A13)

Beschreibung: Es soll der Simulationszeitschritt der zuletzt gespeicherten Matrixqualität zurückgeliefert werden.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein, ein Simulationszeitschritt ausgeführt und eine Matrixqualität erfasst worden sein.

Nachbedingung: Der Simulationszeitschritt der zuletzt gespeicherten Matrixqualität wurde zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.

- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde keine Matrixqualität erfasst.

Ablauf:

1. Der Simulationszeitschritt der zuletzt gespeicherten Matrixqualität wird zurückgeliefert.

3.3.2.4 Gespeicherte Matrix-ID abfragen (A13)

Beschreibung: Es soll die Matrix-ID einer zuvor abgespeicherten Lösungsmatrix von einer Simulation ausgegeben werden.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren sollte eine Matrix abgespeichert worden sein.

Nachbedingung: Die Matrix-ID wurde zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch keine Matrix abgespeichert.

Ablauf:

1. Es wird die Matrix-ID zurückgeliefert.

3.3.2.5 Verwendete Parameterdatei abfragen (A13)

Beschreibung: Von einer Pandas Simulation soll der Pfad der verwendeten Parameter-File ausgegeben werden, damit die darin definierten Materialeigenschaften analysiert werden können.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt und es existiert eine Parameter-File.

Nachbedingung: Es wurde der Pfad zu dem Parameter-File zurückgeliefert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Es existiert keine Parameter-File.

Ablauf:

1. Es wird das Instanz-Verzeichnis nach der Parameter-File durchsucht.

2. Der Pfad zur Parameter-File wird zurückgeliefert.

3.3.3 Mehrere Pandas-Instanzen

Bei diesem Szenario sollen zwei verschiedene Pandas Instanzen eine gemeinsame Simulation führen. Jede Instanz hat dabei seine eigene Physik und eine eigene Zeitschrittgröße. Die beiden Instanzen tauschen dabei die Freiheitsgrade der Mesh-Elemente aus, wobei der Austausch über die angebundene Datenbank erfolgt. Bei der Simulation fängt eine Instanz an zu rechnen und speichert nach x Schritten seine Daten ab. Diese Daten werden von der zweiten Instanz geladen und mit deren Physik und Zeitschrittgröße weitergerechnet. Nach y Schritten wiederum werden die Ergebnisse der zweiten Instanz gespeichert und können wieder von der ersten Instanz geladen werden. Dieser Zyklus wird so lange wiederholt bis das Ende der Simulation erreicht wurde. Abbildung 26 zeigt eine Übersicht über die Anwendungsfälle einer Simulation mit mehreren Pandas-Instanzen.

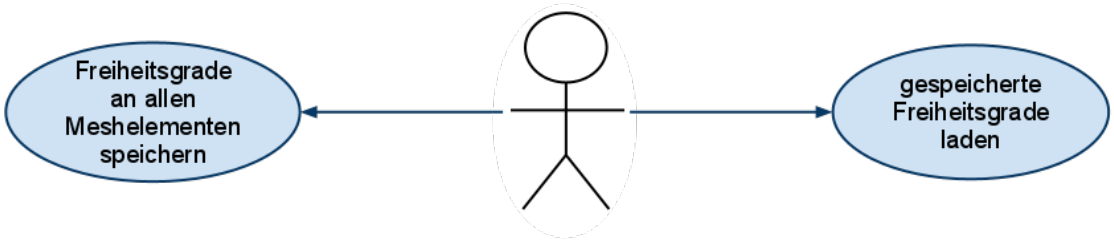

Abbildung 26: Übersicht über die Anwendungsfälle einer Simulation mit mehreren Pandas-Instanzen

3.3.3.1 Freiheitsgrade an allen Mesh-Elementen speichern (A12, A13)

Beschreibung: Es werden alle Freiheitsgrade der Mesh-Elemente in der Datenbank gespeichert. Für die Speicherung wird die interne Pandas-Funktion **`MeshSave`** verwendet. In dem zweidimensionalen Sonderfall wird zusätzlich die interne Pandas-Funktion **`BisectSave`** verwendet, um einen Koordinatenvergleich des aktuellen und des zu ladenden Meshes zu ermöglichen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Die Freiheitsgrade wurden in der Datenbank gespeichert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.

- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.

Ablauf:

1. Es wird eine Mesh-File über die Pandas-Funktion **`MeshSave`** erzeugt.
2. Das erzeugte Mesh-File wird in der Datenbank abgespeichert.
3. Das erzeugte Mesh-File wird gelöscht.
4. Falls es sich um eine zweidimensionale Simulation handelt, wird ein Object-File über die Funktion **`BisectSave`** erstellt.
5. Das erzeugte Object-File wird in der Datenbank abgespeichert.
6. Das erzeugte Object-File wird gelöscht.

3.3.3.2 Gespeicherte Freiheitsgrade laden (A12, A13)

Beschreibung: Es werden alle Freiheitsgrade aus der DATENBANK in Pandas geladen. Für das Laden wird die interne Pandas-Funktion **`MeshLoad`** verwendet. In dem zweidimensionalen Sonderfall wird zusätzlich die interne Pandas-Funktion **`BisectLoad`** verwendet, damit die Koordinaten des aktuellen und des zu ladenden Meshes verglichen werden können.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Alle Freiheitsgrade sind geladen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.
- Die Datenbank enthält die zu ladende Daten nicht.
- Die Koordinaten des aktuellen und des gespeicherten Meshes passen nicht zusammen. (Nur im zweidimensionalen Fall!)

Ablauf:

1. Falls es sich um eine zweidimensionale Simulation handelt, wird das Object-File aus der Datenbank geholt.
2. Das Object-File wird temporär geladen.
3. Im zweidimensionalen Fall werden die Koordinaten des aktuellen und des temporären Meshes verglichen.
4. Falls die beiden Meshes nicht zusammenpassen, wird das temporäre Mesh verworfen und ein Fehler ausgegeben.
5. Falls die beiden Meshes zusammenpassen, wird das temporäre Mesh verworfen und das zu ladende Mesh-File wird aus der Datenbank geholt.

6. Das erzeugte Mesh-File wird über die Funktion **`MeshLoad`** in Pandas geladen.

3.3.4 Pandas-Matlab Kopplung

In diesem Szenario soll eine Kopplung von Pandas mit Matlab stattfinden, das heißt Pandas und Matlab führen gemeinsam eine Multi-Tool, Multi-Skalen und Multi-Domänen Simulation aus. Hierbei werden die Strukturveränderungen eines menschlichen Oberschenkelknochens simuliert. Pandas berechnet die mechanischen Belastungen, welche auf den Knochen einwirken, und Matlab berechnet die systembiologische Komponente der Kalziumbildung des Knochens auf zellularer Ebene. Das Modell für die Matlab-Simulation liegt in Form einer oder mehrerer Dateien mit Matlab-Anweisungen vor. Diese Dateien haben die Endung **`.m`** und werden dementsprechend als M-Files bezeichnet. Hierbei tauschen Pandas und Matlab bestimmte Variablen an den Gausspunkten der Mesh-Elemente aus. Der Austausch selbst erfolgt über die Datenbank und der daraus erstellten CSV-Dateien. Bei der Simulation beginnt Pandas mit der Berechnung des biomechanischen Knochenmodells. Die Berechnung endet, bezüglich des biomechanischen Knochenmodells, nach einer bestimmten Anzahl von Zeitschritten. Da das biomechanische und das biologische Modell voneinander abhängen, speichert Pandas sein Berechnungsergebnis in der Datenbank ab. Bevor die Berechnung des biologischen Knochenmodells auf zellularer Ebene gestartet werden kann, muss aus dem Berechnungsergebnis eine CSV-Datei erstellt werden, welche als Eingabe für Matlab dient. Nun kann das biologische Knochenmodell mit Matlab berechnet werden. Diese Berechnung endet, bezüglich des biologischen Knochenmodells, nach einer bestimmten Anzahl von Zeitschritten.Das Ergebnis der Matlab Berechnung ist selbst wieder eine CSV-Datei, welche in die Datenbank und anschließend in Pandas geladen werden kann. Dieser Zyklus vollzieht sich bis die Simulation ihr Ende erreicht hat. Abbildung 27 zeigt eine Übersicht über die Anwendungsfälle der Pandas-Matlab Kopplung.

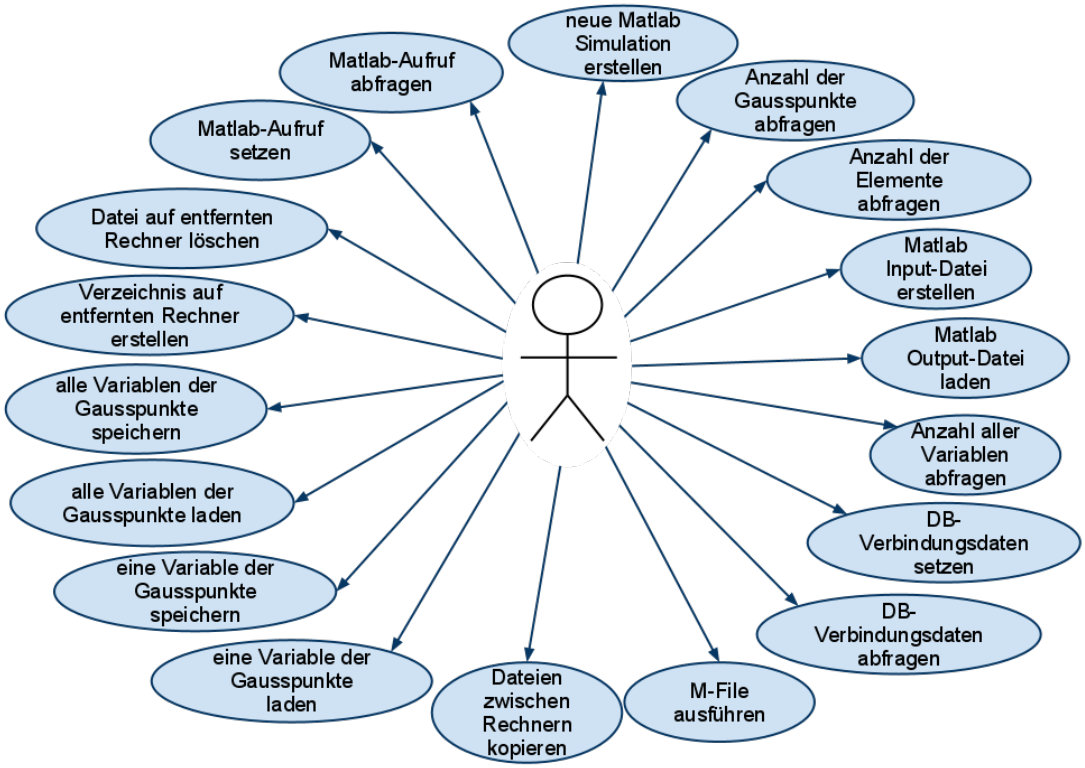

Abbildung 27: Übersicht über die Pandas-Matlab Anwendungsfälle

3.3.4.1 Neue Matlab Simulation erstellen (A13)

Beschreibung: Damit man eine Matlab Simulation starten und steuern kann, muss zunächst eine neue Simulationsinstanz erzeugt werden. Hierzu wird zunächst über den generischen WSI-Adapter eine neue Instanz erzeugt, womit eine neue ID und ein neues dazugehöriges Verzeichnis erstellt werden. In diese neue Simulationsinstanz wird anschließend das Matlab Archiv entpackt, welches die M-Files und notwendigen Dateien für die Simulation enthält. Der Instanzstatus wird auf *Runnable* gesetzt.

Vorbedingung: Im Archiv-Verzeichnis des generischen WSI-Adapters muss das Unterverzeichnis `src` existieren und darin das Matlab Archiv, welches man angeben kann.

Nachbedingung: Die Simulationsinstanz wurde erzeugt und das Matlab Archive wurden darin entpackt. Die Simulationsinstanz ist in einem ausführbaren Zustand.

Fehler:

- Archiv-Files existieren nicht.
- Schreibfehler, volles Filesystem.

Ablauf:

1. Es wird eine neue Simulationsinstanz erzeugt.
2. Das Matlab Archiv wird entpackt.

3. Der Instanz Status wird auf *Runnable* gesetzt.

3.3.4.2 Alle Variablen der Gausspunkte speichern (A12, A13)

Beschreibung: Speichert alle Variablen an allen Gausspunkten an allen Elementen von Pandas in der Datenbank ab.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Alle Variablen an allen Gausspunkten von allen Elementen sind in der Datenbank abgespeichert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.

Ablauf:

1. Es werden alle Mesh-Elemente und deren Gausspunkte traversiert.
2. An einem Gausspunkt werden alle Variablen in der Datenbank abgespeichert.

3.3.4.3 Alle Variablen der Gausspunkte laden (A12, A13)

Beschreibung: Es werden alle Werte aller Variablen an allen Gausspunkten und an allen Elementen aus der Datenbank in Pandas geladen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein. Die zu ladenden Variablen sollten in der Datenbank vorhanden sein.

Nachbedingung: Pandas hat alle Variablen aus der Datenbank geladen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.
- Die Datenbank enthält keine Werte.
- Datenbank enthält die zu ladenden Daten nicht.

Ablauf:

1. Es werden alle Mesh-Elemente und deren Gausspunkte traversiert.
2. An einem Gausspunkt werden alle Variablen aus der Datenbank in Pandas geladen.

3.3.4.4 Eine Variable der Gausspunkte speichern (A2, A3)

Beschreibung: Speichert eine vorgegebene Variable von allen Gausspunkten an allen Elementen von Pandas in der Datenbank ab.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein, ein Simulationszeitschritt ausgeführt worden sein.

Nachbedingung: Die vorgegebene Variable wurde an allen Gausspunkten von allen Elementen in der Datenbank abgespeichert.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.
- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.

Ablauf:

1. Es werden alle Mesh-Elemente und deren Gausspunkte traversiert.
2. An einem Gausspunkt wird die vorgegebene Variable in der Datenbank abgespeichert.

3.3.4.5 Eine Variable der Gausspunkte laden (A2, A3)

Beschreibung: Es werden die Werte der angegebenen Variable an allen Gausspunkten und an allen Elementen aus der Datenbank in Pandas geladen.

Vorbedingung: Es wurde eine Pandas-Simulationsinstanz erzeugt, Pandas ist gestartet und mit einer Datenbank verbunden. Des Weiteren muss Pandas und das Simulationsproblem initialisiert sein und ein Simulationszeitschritt ausgeführt worden sein. Die zu ladende Variable sollte in der Datenbank vorhanden sein.

Nachbedingung: Es wurde die angegebene Variable aller Gausspunkte aus der Datenbank in Pandas geladen.

Fehler:

- Die angegebene Pandas-Simulationsinstanz existiert nicht.
- Pandas ist nicht in dieser Simulationsinstanz gestartet.

- Pandas ist nicht mit der Datenbank verbunden.
- Es wurde noch kein Simulationszeitschritt ausgeführt.
- Die Datenbank enthält die zu ladende Variable nicht.
- Datenbank enthält die zu ladenden Daten nicht.

Ablauf:

1. Es werden alle Mesh-Elemente und deren Gausspunkte traversiert.
2. An einem Gausspunkt wird die angegebene Variable aus der Datenbank in Pandas geladen.

3.3.4.6 Anzahl der Gausspunkte abfragen (A13)

Beschreibung: Die Datenbank wird abgefragt, wie viele Gausspunkte ein Element hat. Dies wird benötigt, um eine CSV-Datei zu erstellen bzw. eine CSV-Datei einzulesen.

Vorbedingung: Die Variablen an den Gausspunkten wurden zuvor in der Datenbank gespeichert.

Nachbedingung: Die Anzahl der Gausspunkte an einem Element wurde ausgegeben.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Die Datenbank-Verbindungsdaten sind nicht korrekt.
- Die Variablen an den Gausspunkten wurden nicht in der Datenbank gespeichert.

Ablauf:

1. Die Anfrage wird an die Datenbank geschickt.
2. Die Anzahl der Gausspunkte an einem Element wird zurückgegeben.

3.3.4.7 Anzahl der Elemente abfragen (A13)

Beschreibung: Die Datenbank wird abgefragt, wie viele Elemente das Mesh hat. Dies wird benötigt, um eine CSV-Datei zu erstellen bzw. eine CSV-Datei einzulesen.

Vorbedingung: Die Variablen an den Gausspunkten wurden zuvor in der Datenbank gespeichert.

Nachbedingung: Die Anzahl der Elemente wurde ausgegeben.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Die Datenbank-Verbindungsdaten sind nicht korrekt.
- Die Variablen an den Gausspunkten wurden nicht in der Datenbank gespeichert.

Ablauf:

1. Die Anfrage wird an die Datenbank geschickt.
2. Die Anzahl der Elemente wird zurückgegeben.

3.3.4.8 Anzahl aller Variablen abfragen (A13)

Beschreibung: Die Datenbank wird abgefragt, wie viele Variablenwerte insgesamt von dem Mesh gespeichert wurden.

Vorbedingung: Die Variablen an den Gausspunkten wurden zuvor in der Datenbank gespeichert.

Nachbedingung: Die Anzahl aller Variablenwerte wurde zurückgegeben.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Die Datenbank-Verbindungsdaten sind nicht korrekt.
- Die Variablen an den Gausspunkten wurden nicht in der Datenbank gespeichert.

Ablauf:

1. Die Anfrage wird an die Datenbank geschickt.
2. Die Anzahl der Elemente wird zurückgegeben.

3.3.4.9 Matlab Input-Dateien erstellen (A13)

Beschreibung: Es soll eine CSV-Eingabedatei für Matlab erstellt werden. In dieser Datei stehen alle Variablen aller Gausspunkte. Dabei stellt in der CSV-Datei eine Zeile einen Gausspunkt und die Spalten die dazugehörigen Variablen dar. Ein Element hingegen besitzt mehrere Gausspunkte. Das heißt ein Element mit seinen Gausspunkten und Variablen erstreckt sich über mehrere Zeilen der CSV-Datei. Die Struktur der Datei enthält somit implizit Informationen über verschiedene benötigte Indizes wie zum Beispiel die Elementnummer oder die Gausspunktnummer. Der Umfang der Datei wird über die Anzahl der Elemente gesteuert.

Vorbedingung: Die Variablen an den Gausspunkten wurden zuvor in der Datenbank gespeichert.

Nachbedingung: Die CSV-Eingabedatei wurde in das Matlab-Instanzverzeichnis geschrieben.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Die Variablen an den Gausspunkten wurden nicht in der Datenbank gespeichert.
- Die Datenbank-Verbindungsdaten sind nicht korrekt.
- Das Filesystem ist voll die Datei konnte nicht erstellt werden.

Ablauf:

1. Eine Datenbank-Abfrage nach allen Variablen an allen Gausspunkten wird abgesetzt.
2. Aus der Datenbank-Antwort wird die CSV-Datei erstellt.

3.3.4.10 Matlab Output-Datei laden (A13)

Beschreibung: Es soll eine Matlab-CSV-Ausgabedatei eingelesen werden, und die Variablen an den Gausspunkten in die Datenbank geladen werden. Dabei stellt in der CSV-Datei eine Zeile einen Gausspunkt und die Spalten die dazugehörigen Variablen dar. Ein Element hingegen besitzt mehrere Gausspunkte. Das heißt ein Element mit seinen Gausspunkten und Variablen erstreckt sich über mehrere Zeilen der CSV-Datei. Die Struktur der Datei enthält somit implizit Informationen über verschiedene benötigte Indizes wie zum Beispiel die Elementnummer oder die Gausspunktnummer.

Vorbedingung: Die CSV-Ausgabedatei existiert im Matlab-Instanzverzeichnis und das Format ist korrekt. Von dieser Matlab-Simulationsinstanz wurde zuvor eine CSV-Eingabedatei erstellt, damit die Datenbank für das Laden des Ergebnisses von Matlab vorbereitet wurde. Zudem sollten die Verbindungsdaten der Datenbank korrekt sein.

Nachbedingung: Alle Werte aus der CSV-Ausgabedatei sind in die Datenbank geladen.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Das Format der CSV-Datei ist nicht korrekt.
- Die CSV-Datei existiert nicht im Matlab-Instanzverzeichnis.
- Es wurde zuvor keine Matlab CSV-Eingabedatei erstellt.

Ablauf:

1. Es wird geprüft, ob die CSV-Ausgabedatei im Matlab-Instanzverzeichnis existiert.
2. Die CSV-Ausgabedatei wird eingelesen und werteweise in die Datenbank gespeichert.

3.3.4.11 DB-Verbindungsdaten setzen (A13)

Beschreibung: Es soll möglich sein, die Datenbank anzugeben, worüber der Austausch der Variablen an den Gausspunkten stattfindet. Die Datenbank-Verbindungsdaten werden in einem Property-File im Matlab-Instanzverzeichnis gespeichert.

Vorbedingung: Es existiert die Matlab-Simulationsinstanz.

Nachbedingung: Das Property-File mit den Datenbank-Verbindungsdaten wurde erzeugt oder aktualisiert.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.

Ablauf:

1. Es wird geprüft, ob das Property-File existiert.
2. Falls das Property-File existiert, wird es mit den neuen Werten aktualisiert.
3. Falls das Property-File nicht existiert, wird es neu erzeugt.

3.3.4.12 DB-Verbindungsdaten abfragen (A3)

Beschreibung: Es soll möglich sein die Datenbank-Verbindungsdaten abzufragen.

Vorbedingung: Die Matlab-Simulationsinstanz und darin die Property-File mit den Verbindungsdaten existiert.

Nachbedingung: Die Datenbank-Verbindungsdaten wurden zurückgegeben.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Das Property-File existiert nicht.

Ablauf:

1. Es wird geprüft, ob das Property-File existiert.
2. Das Property-File wird gelesen und die Verbindungsdaten zurückgegeben.

3.3.4.13 M-File ausführen (A3)

Beschreibung: Es soll möglich sein, auf einem entfernten Rechner eine M-File von Matlab oder Octave ausführen zu lassen und somit eine Berechnung anzustoßen. Dies soll über einen SSH-Aufruf realisiert werden. Dabei muss auf dem entfernten Rechner ein Verzeichnis mit dem auszuführenden M-File vorhanden sein.

Vorbedingung: Die Matlab-Simulationsinstanz und darin das Skript für den SSH-Aufruf müssen existieren. Auf dem entfernten Rechner muss ein Verzeichnis vorbereitet sein, welches das auszuführende M-File enthält. Des Weiteren muss zwischen dem entfernten Rechner und dem Rechner auf dem der WSI-Adapter läuft ein SSH-Schlüsselaustausch stattgefunden haben, damit keine weitere Authentifizierung im späteren Ablauf stattfinden muss.

Nachbedingung: Das M-File wurde auf dem entfernten Rechner ausgeführt.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.

- Es wurde kein SSH-Schlüsselaustausch zwischen den beteiligten Rechnern vorgenommen.
- Das M-File ist auf dem entfernten Rechner nicht vorhanden.
- Das Shell-Skript zum Absetzen des SSH-Befehls ist nicht vorhanden.

Ablauf:

1. Es wird eine SSH-Verbindung zum entfernten Rechner aufgebaut.
2. Auf dem entfernten Rechner wird in das Simulationsinstanz Verzeichnis gewechselt.
3. Matlab oder Octave wird mit dem M-File ausgeführt.

3.3.4.14 Dateien zwischen Rechnern kopieren (A3)

Beschreibung: Es muss möglich sein zwischen zwei entfernten Rechnern Dateien zu kopieren. Dies wird benötigt, um Dateien von und zum Matlab-Rechner zu kopieren.

Vorbedingung: Die Matlab-Simulationsinstanz und darin das Skript für den SCP-Aufruf müssen existieren. Auf dem entfernten Rechner muss ein Verzeichnis mit Schreibrechten vorhanden sein. Des Weiteren muss zwischen dem entfernten Rechner und dem Rechner auf dem der WSI-Adapter läuft ein SSH-Schlüsselaustausch stattgefunden haben, damit keine weitere Authentifizierung im späteren Ablauf stattfinden muss. Die zu kopierende Datei muss vorhanden sein.

Nachbedingung: Die Datei wurde kopiert.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Es wurde kein SSH-Schlüsselaustausch zwischen den beteiligten Rechnern vorgenommen.
- Das Shell-Skript zum Absetzen des SCP-Befehls ist nicht vorhanden.
- Auf der Zielseite sind keine Schreibrechte vorhanden.
- Die zu kopierende Datei ist nicht vorhanden.

Ablauf:

1. Es wird ein SCP-Aufruf abgesetzt.
2. Die Datei wird über SCP kopiert.

3.3.4.15 Verzeichnis auf entfernten Rechner erstellen (A3)

Beschreibung: Erstellt ein Verzeichnis auf einen anderen Rechner über SSH. Dies wird benötigt um auf dem Matlab-Rechner ein Simulationsverzeichnis zu erstellen.

Vorbedingung: Die Matlab-Simulationsinstanz und darin das Skript für den SSH-Aufruf müssen existieren. Auf dem entfernten Rechner müssen Schreibrechte vorhanden sein. Des Weiteren muss zwischen dem entfernten Rechner und dem Rechner auf dem der WSI-Adapter läuft ein SSH-

Schlüsselaustausch stattgefunden haben, damit keine weitere Authentifizierung im späteren Ablauf stattfinden muss.

Nachbedingung: Auf dem Zielrechner wurde das Verzeichnis erstellt.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Es wurde kein SSH-Schlüsselaustausch zwischen den beteiligten Rechnern vorgenommen.
- Das Shell-Skript zum Absetzen des SCP-Befehls ist nicht vorhanden.
- Auf der Zielseite sind keine Schreibrechte vorhanden.

Ablauf:

1. Es wird über SSH der Befehl zum Erstellen des Verzeichnisses abgesetzt.
2. Das Verzeichnis wird auf dem entfernten Rechner erstellt.

3.3.4.16 Datei auf entfernten Rechner löschen (A13)

Beschreibung: Löscht eine Datei auf einem entfernten Rechner über einen SSH Befehl. Dies wird benötigt, um das Instanzverzeichnis auf einem anderen Rechner aufzuräumen.

Vorbedingung: Die Matlab-Simulationsinstanz und darin das Skript für den SSH-Aufruf müssen existieren. Auf dem entfernten Rechner müssen Schreibrechte vorhanden sein. Des Weiteren muss zwischen dem entfernten Rechner und dem Rechner auf dem der WSI-Adapter läuft ein SSH-Schlüsselaustausch stattgefunden haben, damit keine weitere Authentifizierung im späteren Ablauf stattfinden muss. Die zu löschende Datei muss vorhanden sein.

Nachbedingung: Die Datei wird auf der Zielseite gelöscht.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Es wurde kein SSH-Schlüsselaustausch zwischen den beteiligten Rechnern vorgenommen.
- Das Shell-Skript zum Absetzen des SCP-Befehls ist nicht vorhanden.
- Auf der Zielseite sind keine Schreibrechte vorhanden.
- Die Datei auf der Zielseite existiert nicht.

Ablauf:

1. Es wird über SSH der Befehl zum Löschen der Datei abgesetzt.
2. Die Datei wird auf dem entfernten Rechner gelöscht.

3.3.4.17 Matlab-Aufruf setzen (A3)

Beschreibung: Es muss möglich sein das Kommando zum Ausführen eines M-File in einer Matlab-Simulationsinstanz festzulegen, da auf jedem entfernten Rechner sich der Installationspfad unterscheiden kann. Ebenso ist es damit möglich, statt einem Matlab-Aufruf ein Octave-Aufruf festzulegen. Dieser Aufruf wird in einem Property-File in dem Matlab-Instanzverzeichnis gespeichert.

Vorbedingung: Die Matlab-Simulationsinstanz muss existieren.

Nachbedingung: Das Property-File wurde erstellt oder aktualisiert.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.

Ablauf:

1. Es wird geprüft, ob das Property-File existiert.
2. Falls das Property-File existiert, wird es mit den neuen Werten aktualisiert.
3. Falls das Property-File nicht existiert, wird es neu erzeugt.

3.3.4.18 Matlab-Aufruf abfragen (A3)

Beschreibung: Es soll möglich sein den gesetzten Aufruf zum Ausführen eines M-Files abzufragen.

Vorbedingung: Die Matlab-Simulationsinstanz muss existieren und es muss zuvor ein Aufruf gesetzt worden sein.

Nachbedingung: Der Aufruf wurde zurückgeliefert.

Fehler:

- Die angegebene Matlab-Simulationsinstanz existiert nicht.
- Der Aufruf wurde nicht gesetzt bzw. das Property-File existiert nicht.

Ablauf:

1. Es wird geprüft, ob das Property-File existiert.
2. Das Property-File wird gelesen und der Aufruf zurückgegeben.

4 Entwurf

In diesem Kapitel werden neben den Architekturen der verschiedenen WSI-Adapter auch deren Web Service Operationen und Parameter vorgestellt. Zum Abschluss wird das Datenbankschema zur Speicherung von Daten aus der Simulation gezeigt.

4.1 Architektur

Architektur des WSI-Pandas-Adapters

Die Architektur des WSI-Pandas-Adapters ist auf der Abbildung 28 zu sehen und wird im Folgenden beschrieben. Der WSI-Pandas-Adapter ist der Stellvertreter für die Pandas-Anwendung, da diese nicht als Dienst laufen kann. Der WSI-Pandas-Adapter bietet im Grunde die gleichen Operationen an wie die Pandas-Anwendung selber. Damit der WSI-Pandas-Adapter mit der Pandas-Anwendung interagieren kann, muss die Pandas-Anwendung zunächst erstellt und gestartet werden. Hierzu greift der WSI-Pandas-Adapter auf den Instanz-Pool und dem Programm-Manager des generischen WSI-Adapters zu, um eine Simulationsinstanz zu erzeugen und darin den Sourcecode der Pandas-Anwendung zu entpacken. Instanz-Pool und Programm-Manager werden dann dazu verwendet den Sourcecode von Pandas zu kompilieren und zu starten. Die modifizierte Pandas Anwendung verwendet nach dem Start den Callback Web Service des generischen WSI-Adapters, um sich zurückzumelden. Durch diese Rückmeldung ist die Simulationsinstanz in einem lauffähigen Zustand. Nun kann der WSI-Pandas-Adapter die Web Service Anfragen durch den Pandas Service Stub an die Pandas Anwendung weiterleiten.

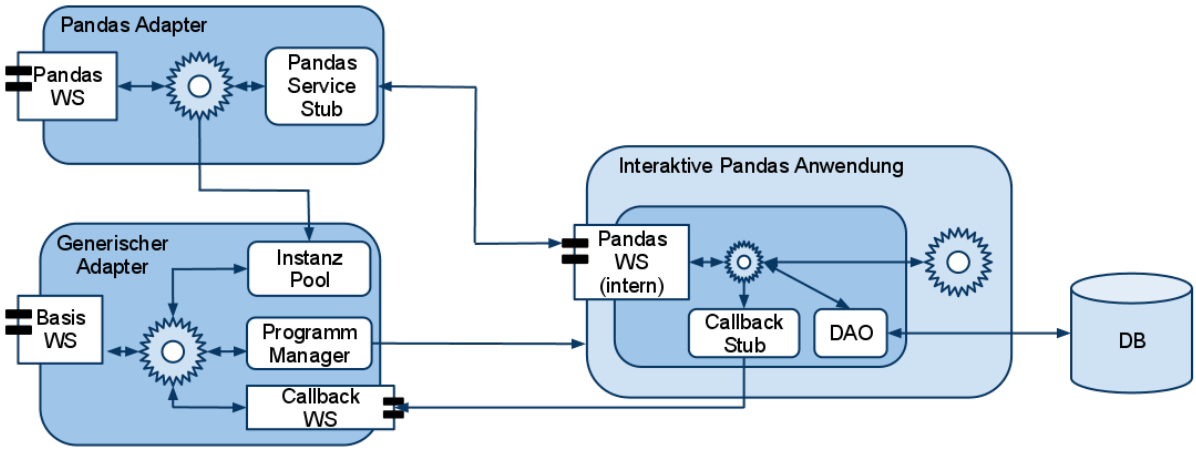

Abbildung 28: Architektur Pandas Adapter

Architektur des WSI-Matlab-Adapters

Auf der Abbildung 29 ist die Architektur des WSI-Matlab-Adapters zu sehen und wird im Folgenden näher beschrieben. Da Matlab nicht ohne weiteres als eigener Web Service zur

Verfügung steht, werden für den WSI-Matlab-Adapter ausschließlich Funktionalitäten des generischen WSI-Adapters verwendet. Wie bei dem WSI-Pandas-Adapter wird der Instanz-Pool und der Programm-Manager zur Erstellung und Verwaltung der Simulationsinstanz sowie zum Entpacken des Matlab-Archives verwendet. In dem Archiv befindet sich ein Shell Skript, welches es dem generischen WSI-Adapter erlaubt via SSH auf entfernten Rechnern Befehle abzusetzen und via SCP Dateien zwischen einem entfernten Rechner und dem erzeugten Simulationsinstanzverzeichnis zu kopieren.

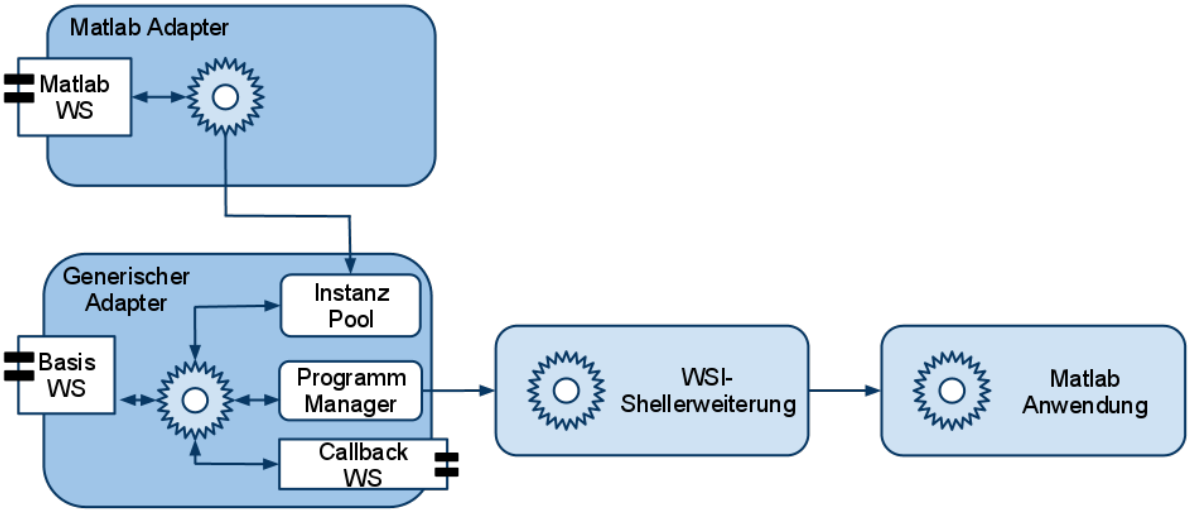

Abbildung 29: Architektur Matlab Adapter

Architektur des WSI-PMConnector-Adapters

Die Architektur des WSI-PMConnector-Adapters ist auf der Abbildung 30 dargestellt und wird nun näher beschrieben. PM steht für Pandas-Matlab. Der WSI-PMConnector-Adapter erstellt für die Matlab-Simulation die CSV-Eingabedatei aus der Datenbank und speichert von der Matlab-Simulation die CSV-Ausgabedatei in der Datenbank ab. Damit der WSI-PMConnector-Adapter Zugriff auf das Matlab-Simulationsinstanzverzeichnis bekommt, wird auf den Instanz-Pool des generischen WSI-Adapters zugegriffen.

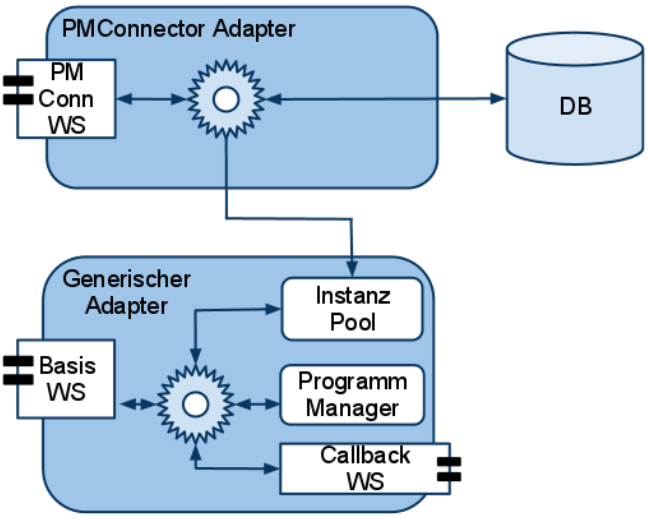

Abbildung 30: Architektur PMConnector Adapter

4.2 Web Service-Operationen

In diesem Abschnitt werden die gesamten Schnittstellen der WSI-Adapter aufgelistet und die Parameter der Operationen beschrieben.

4.2.1 WSI_Pandas

prepareSimulation (𝔸1)

Erstellt eine neue und ausführbare Pandas Simulationsinstanz.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: neue Pandas Simulation erstellen

Input/Output	Parametername	Datentyp	Beschreibung
Input	ProblemName	String	Name des auszupackenden Simulationsproblem
Output	SimID	Long	Die Simulations-ID der neuen Instanz
Output	ReturnMessage	String	Return Message

Tabelle 2: Parameter der Operation prepareSimulation

startPandas (𝔸1)

Startet Pandas der angegebenen Simulationsinstanz.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas starten

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	ProgramPath	String	Der Pfad zum Pandas-Binary relativ zur Instanz
Input	WorkingDirectory	String	Der Pfad des zu verwendenden Arbeitsverzeichnisses
Input	Arguments	String	Die zu verwendenden Kommandozeilen-Parameter
Input	ExecutionLog	String	Der Pfad zur Datei, welche die Ausgabe von Pandas speichert.
Output	ReturnMessage	String	Return Message

Tabelle 3: Parameter der Operation startPandas

stopApplication ($\mathbb{A}1$)

Beendet Pandas.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas beenden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz

Tabelle 4: Parameter der Operation stopApplication

connect-db ($\mathbb{A}1$, $\mathbb{A}2$)

Verbindet Pandas mit einer Datenbank.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: DB-Verbindung aufbauen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	backend	String	gibt den Datenbank-Typ an (Bsp. pgsql)
Input	host	String	Host auf dem die Datenbank läuft
Input	port	String	Port auf dem die Datenbank lauscht
Input	db	String	Name der Datenbank
Input	user	String	Username zum Anmelden
Input	pw	String	Passwort zum Anmelden
Input	encryption	Boolean	soll die Verbindung Verschlüsselt werden
Input	compression	Boolean	soll die Verbindung komprimiert werden

Tabelle 5: Parameter der Operation connect-db

disconnect-db ($\mathbb{A}1$, $\mathbb{A}2$)

Trennt die Datenbank-Verbindung von Pandas.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: DB-Verbindung trennen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	host	String	Host auf dem die Datenbank läuft
Input	port	String	Port auf dem die Datenbank lauscht
Input	db	String	Name der Datenbank

Tabelle 6: Parameter der Operation disconnect-db

set-option ($\mathbb{A}2$, $\mathbb{A}3$)

Hierüber sollte man das Verhalten der Matrixspeicherung steuern können. Darüber hinaus lässt sich hierüber initial die Datenbank-Tables anlegen.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix zyklisch speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	save-matrix	Boolean	soll die Matrix zyklisch gespeichert werden?
Input	load-matrix	Boolean	soll die Matrix geladen werden?
Input	save-binary	Boolean	soll die Matrix binär gespeichert werden?
Input	create-tables	Boolean	sollen die Datenbank-Tables erstellt werden?
Input	step-width	Integer	Schrittweite zum zyklischen speichern

Tabelle 7: Parameter der Operation set-option

run-cmd ($\mathbb{A}1$)

Führt ein Batch-File aus, um Pandas zu initialisieren.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas Batchfile ausführen

Input/Output	Parametername	Datentyp	Beschreibung
Input	sid	Long	Simulations-ID der Instanz
Input	cmd-filename	String	Name der auszuführenden CMD-File

Tabelle 8: Parameter der Operation run-cmd

readProblem ($\mathbb{A}1$)

Lässt Pandas ein Simulationsproblem über 4 unterschiedliche Alternativen einlesen.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Simulationsproblem einlesen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	ShapeFile	String	Name der Shape-Datei (*.shape)
Input	GeomFile	String	Name der Geometrie-Datei (*.geom)
Input	createMesh	Boolean	soll das Netz generiert oder eingelesen werden
Input	BaseName	String	Parameter für den TRIANGLE-Netzgenerator – bestimmt Ausgabedatei
Input	minAngle	Float	Parameter für den TRIANGLE-Netzgenerator – Vorgabe eines minimalen Winkels
Input	maxArea	Long	Parameter für den TRIANGLE-Netzgenerator – Vorgabe einer max. Elementfläche
Input	o2	Boolean	Parameter für den TRIANGLE-Netzgenerator – Zusatzoption für Dreieckelemente mit quad. Ansatz
Input	creategrid	Boolean	soll ein einfaches Vierecks-Gitter generiert werden
Input	ShapeType	String	Parameter für Grid-Generator – Typ der Ansatzfunktion
Input	PhysType	String	Parameter für Grid-Generator – Name der Physik
Input	nel_x	Integer	Parameter für Grid-Generator – Anzahl der Elemente in Richtung der x-Koordinate
Input	nel_y	Integer	Parameter für Grid-Generator - Anzahl der Elemente in Richtung der y-Koordinate
Input	useNodeElems	Boolean	soll das Mesh über die Knoten- und Element-Datei eingelesen werden
Input	NodesFile	String	Name der Knoten-Datei (*.nodes)
Input	ElemsFile	String	Name der Element-Datei (*.elems)
Input	BoundaryFile	String	Name der Randbedingungs-Datei (*.bound)
Input	DescriptionFile	String	Name der Problembeschreibungsdatei (*.descr)
Input	IvarsFile	String	Name der Datei der internen Variablen (*.ivars)
Input	ParamFile	String	Name der Materialparameter-Datei (*.param)

Tabelle 9: Parameter der Operation readProblem

executeCommandSync (𝔸1)

Führt ein Pandas-Kommando aus.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas Kommando ausführen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	Command	Sting	Der auszuführende Befehl in Pandas

Tabelle 10: Parameter der Operation executeCommandSync

do-step ($\mathbb{A}1$)

Führt einen Simulationszeitschritt über das Pandas-Kommando **`step 1`** aus.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Einen Simulationszeitschritt ausführen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	stepnr	Integer	der aktuelle Simulationszeitschritt

Tabelle 11: Parameter der Operation do-step

getStepnr ($\mathbb{A}1$)

Hierüber wird der aktuelle Simulationszeitschritt von Pandas abgefragt.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Aktuellen Simulationszeitschritt abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Stepnr	Integer	der aktuelle Simulationszeitschritt

Tabelle 12: Parameter der Operation getStepnr

saveState ($\mathbb{A}1$, $\mathbb{A}2$)

Speichert den aktuellen Zustand von Pandas in der Datenbank ab.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Zustand speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Stepnr	Integer	der aktuelle Simulationszeitschritt

Tabelle 13: Parameter der Operation saveState

loadState ($\mathbb{A}1$, $\mathbb{A}2$)

Lädt einen abgespeicherten Zustand von Pandas aus der Datenbank.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Zustand laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz

Input	load_SimID	Long	Simulations-ID der zu ladenden Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt des Zustandes

Tabelle 14: Parameter der Operation loadState

getDataQualityQuery (𝔸3)

Liefert ein Query zur zuletzt abgespeicherten Matrix-Qualität zurück.

Szenario: Data-Quality

Anwendungsfall: SQL-Query zur Matrixqualität

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	sql-stmt	String	Query zur zuletzt abgespeicherten Matrix-Qualität

Tabelle 15: Parameter der Operation getDataQualityQuery

getLastSavedStepnr (𝔸3)

Liefert den Simulationszeitschritt zurück, bei dem zuletzt die Matrix-Qualität abgespeichert wurde.

Szenario: Data-Quality

Anwendungsfall: Zeitschritt der letzten Matrixqualität

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Stepnr	Integer	Der Simulationszeitschritt des Zustandes

Tabelle 16: Parameter der Operation getLastSavedStepnr

getMid (𝔸3)

Liefert die Matrix-ID zurück.

Szenario: Data-Quality, DUNE-Matrixlöser

Anwendungsfall: Gespeicherte Matrix-ID abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Mid	Integer	Matrix-ID

Tabelle 17: Parameter der Operation getMid

saveDataQuality (𝔸2, 𝔸3)

Erfasst die aktuelle Matrixqualität und speichert diese ab.

Szenario: Data-Quality

Anwendungsfall: Matrixqualität erfassen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz

Tabelle 18: Parameter der Operation saveDataQuality

getUsedParamFile (𝔸3)

Gibt den Pfad zur verwendeten Parameterfile zurück.

Szenario: Data-Quality

Anwendungsfall: Verwendete Parameterdatei abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	ProblemName	String	Der Name des Simulationsproblem
Output	Path	String	Der Pfad zur verwendeten ParamFile

Tabelle 19: Parameter der Operation getUsedParamFile

save-dof (𝔸2, 𝔸3)

Speichert die Freiheitsgrade des Meshes in der Datenbank ab.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfall: Freiheitsgrade an allen Mesh-Elementen speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	stepnr	Integer	Der aktuelle Simulationszeitschritt

Tabelle 20: Parameter der Operation save-dof

load-dof (𝔸2, 𝔸3)

Lädt die Freiheitsgrade des Meshes aus der Datenbank.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfall: Gespeicherte Freiheitsgrade laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	load_SimID	Long	Simulations-ID der zu ladenden Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt der zu ladenden Instanz

Tabelle 21: Parameter der Operation load-dof

saveMatrix ($\mathbb{A}2$, $\mathbb{A}3$)

Speichert die Matrix von Pandas in der Datenbank binär oder werteweise ab.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	saveBinary	Boolean	soll die Matrix binär abgespeichert werden?
Output	Stepnr	Integer	Der Simulationszeitschritt bei dem die Matrix abgespeichert wurde

Tabelle 22: Parameter der Operation saveMatrix

loadMatrix ($\mathbb{A}2$, $\mathbb{A}3$)

Lädt die Matrix von Pandas aus der Datenbank.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	load_SimID	Long	die Simulations-ID der zu ladenden Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt bei dem die Matrix gespeichert wurde
Input	loadBinary	Boolean	soll die Matrix binär geladen werden

Tabelle 23: Parameter der Operation loadMatrix

saveAllGauss ($\mathbb{A}2$, $\mathbb{A}3$)

Speichert alle Variablen von allen Gausspunkten aller Elemente in der Datenbank ab.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Alle Variablen der Gausspunkte speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Stepnr	Integer	Der Simulationszeitschritt bei dem die Variablen an den Gausspunkten abespeichert wurde

Tabelle 24: Parameter der Operation saveAllGauss

loadAllGauss (𝔸2, 𝔸3)

Lädt aus der Datenbank alle Variablen von allen Gausspunkten aller Elemente.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Alle Variablen der Gausspunkte laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	load_SimID	Long	die Simulations-ID der zu ladenden Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt bei dem die Gausspunkte gespeichert wurde

Tabelle 25: Parameter der Operation loadAllGauss

saveGaussName (𝔸2, 𝔸3)

Speichert die angegebene Variable von allen Gausspunkten aller Elemente in der Datenbank ab.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Eine Variable der Gausspunkte speichern

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	Name	String	Der Name der zu speichernden Variable
Output	Stepnr	Integer	Der Simulationszeitschritt bei dem die Variablen an den Gausspunkten abespeichert wurde

Tabelle 26: Parameter der Operation saveGaussName

loadGaussName (𝔸2, 𝔸3)

Lädt aus der Datenbank die angegebene Variable von allen Gausspunkten aller Elemente.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Eine Variable der Gausspunkte laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	load_SimID	Long	die Simulations-ID der zu ladenden Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt bei dem die Gausspunkte gespeichert wurde
Input	Name	String	Der Name der zu ladenden Variblen an den Gausspunkten

Tabelle 27: Parameter der Operation loadGaussName

4.2.2 WSI_Matlab

prepareSimulation (𝔸3)

Erstellt eine neue Matlab-Simulationsinstanz.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Neue Matlab Simulation erstellen

Input/Output	Parametername	Datentyp	Beschreibung
Input	Name	String	Name des auszupackenden Archives
Output	SimID	Long	Die Simulations-ID der neuen Instanz
Output	ReturnMessage	String	Return Message

Tabelle 28: Parameter der Operation prepareSimulation

Start (𝔸3)

Führt auf einem entfernten Rechner über SSH ein M-File aus. Hierbei muss zuvor der Matlab Path gesetzt sein, damit der richtige Matlab Aufruf stattfindet.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: M-File ausführen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	User	String	User des Matlab Rechners
Input	Host	String	Host des Matlab Rechners
Input	Path	String	Pfad auf dem Matlab Rechner, wo sich die M-File befindet
Input	Program	String	der Name des M-Files
Output	ReturnMessage	String	Return Message

Tabelle 29: Parameter der Operation Start

Copy (𝔸3)

Kopiert eine Datei über SCP zwischen zwei Rechnern.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Dateien zwischen Rechnern kopieren

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	SrcUser	String	User des Rechners auf dem die Quelldatei liegt
Input	SrcHost	String	Host auf dem die Quelldatei liegt
Input	SrcFile	String	der absolute Pfad der Quelldatei
Input	DstUser	String	User des Rechners wo die Datei hin kopiert

			werden soll
Input	DstHost	String	Host des Rechners wo die Datei hin kopiert werden soll
Input	DstFile	String	der absolute Pfad der Zieldatei
Output	ReturnMessage	String	Return Message

Tabelle 30: Parameter der Operation Copy

Mkdir (𝔸3)

Erstellt ein Verzeichnis über SSH auf dem Zielrechner.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Verzeichnis auf entfernten Rechner erstellen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	User	String	User des Rechners auf dem das Verzeichnis angelegt werden soll
Input	Host	String	Host auf dem das Verzeichnis angelegt werden soll
Input	Dir	String	Der absolute Pfad des zu erstellenden Verzeichnisses
Output	ReturnMessage	String	Return Message

Tabelle 31: Parameter der Operation Mkdir

deleteFile (𝔸3)

Löscht eine Datei über SSH auf dem Zielrechner.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Datei auf entfernten Rechner löschen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	User	String	User des Rechners auf dem die Datei gelöscht werden soll
Input	Host	String	Host auf dem die Datei gelöscht werden soll
Input	File	String	Der absolute Pfad der zu löschenden Datei
Output	ReturnMessage	String	Return Message

Tabelle 32: Parameter der Operation deleteFile

setMatlabPath (𝔸3)

Setzt den zu verwendenden Aufruf für Matlab oder Octave. Hierbei ist zu beachten dass der absolute Pfad zum Programm samt Aufrufparametern anzugeben ist.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Matlab-Aufruf setzen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	MatlabPath	String	Der Absolute Pfad zu Matlab oder Octave
Output	ReturnMessage	String	Return Message

Tabelle 33: Parameter der Operation setMatlabPath

getMatlabPath (𝔸3)

Gibt den aktuell gesetzten Aufruf für Matlab oder Octave für die angegebene Instanz.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Matlab-Aufruf abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	MatlabPath	String	Der gesetzte Aufruf für Matlab oder Octave

Tabelle 34: Parameter der Operation getMatlabPath

4.2.3 WSI_PMConnector

prepareSimulation (𝔸3)

Erstellt eine neue Matlab-Simulationsinstanz.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Neue Matlab Simulation erstellen

Input/Output	Parametername	Datentyp	Beschreibung
Input	Name	String	Name des auszupackenden Archives
Output	SimID	Long	Die Simulations-ID der neuen Instanz
Output	ReturnMessage	String	Return Message

Tabelle 35: Parameter der Operation prepareSimulation

getCountGauss (𝔸3)

Liefert die Anzahl der Gausspunkte pro Element.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Anzahl der Gausspunkte abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	FromSimID	Long	Simulations-ID der abgespeicherten Instanz

Input	Stepnr	Integer	Der Simulationszeitschritt der abgespeicherten Instanz
Output	Count	Integer	Anzahl der Gausspunkte pro Element

Tabelle 36: Parameter der Operation getCountGauss

getCountElement (𝔸3)

Liefert die Anzahl von abgespeicherten Elementen zurück.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Anzahl der Elemente abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	FromSimID	Long	Simulations-ID der abgespeicherten Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt der abgespeicherten Instanz
Output	Count	Integer	Anzahl der abgespeicherten Elemente

Tabelle 37: Parameter der Operation getCountElement

createList (𝔸3)

Erstellt eine CSV-Datei als Eingabe für Matlab oder Octave. Hierbei stehen pro Zeile alle Variablen an einem Gausspunkt. Jedes Element besitzt mehrere Gausspunkte. Die Anzahl der Gausspunkte kann nur über die Anzahl der Elemente geregelt werden.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Matlab Input-Dateien erstellen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	FromSimID	Long	Simulations-ID der abgespeicherten Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt der abgespeicherten Instanz
Input	StartElement	Integer	das Element, bei dem die CSV-Datei beginnt
Input	EndElement	Integer	das Element, bei dem die CSV-Datei endet
Input	Time	Integer	die Endzeit, bei der die Matlab Berechnung abbrechen soll
Output	ReturnMessage	String	Return Message

Tabelle 38: Parameter der Operation createList

insertList (𝔸3)

Liest eine CSV-Datei ein und speichert diese in der Datenbank ab.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Matlab Output-Datei laden

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt aus dem die CSV-Datei stammt
Input	StartElement	Integer	bei welchem Element beginnt die CSV-Datei
Input	EndElement	Integer	bei welchem Element endet die CSV-Datei
Input	CountGauss	Integer	wieviele Gausspunkte sind pro Element vorhanden
Output	ReturnMessage	String	Return Message

Tabelle 39: Parameter der Operation insertList

getCountAllValues (𝔸3)

Liefert die Anzahl aller abgespeicherten Variablen an allen Gausspunkten aller Elemente.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Anzahl aller Variablen abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	FromSimID	Long	Simulations-ID der abgespeicherten Instanz
Input	Stepnr	Integer	Der Simulationszeitschritt der abgespeicherten Instanz
Output	Count	Integer	Anzahl der abgespeicherten Variablen

Tabelle 40: Parameter der Operation getCountAllValues

setDBConn (𝔸3)

Setzt die Verbindungsdaten zur Datenbank des WSI_PMConnectors.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: DB-Verbindungsdaten setzen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Input	Host	String	Der Host auf dem die Datenbank läuft
Input	DB	String	Der Datenbankname
Input	User	String	Der Datenbank-User
Input	PW	String	Das Passwort des Datenbank-Users
Output	ReturnMessage	String	Return Message

Tabelle 41: Parameter der Operation setDBConn

getDBConn (𝔸3)

Liefert die Verbindungsdaten des WSI_PMConnectors.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: DB-Verbindungsdaten abfragen

Input/Output	Parametername	Datentyp	Beschreibung
Input	SimID	Long	Simulations-ID der Instanz
Output	Host	String	Der Host auf dem die Datenbank läuft
Output	DB	String	Der Datenbankname

Tabelle 42: Parameter der Operation getDBConn

4.3 Datenbankschema (𝔸2, 𝔸3)

In diesem Abschnitt wird das Datenbankschema mit seinen Tabellen für die Pandas Service-Bus-Erweiterung beschrieben, welches eine Simulation und die verschiedenen Simulationsszenarien abbilden soll.

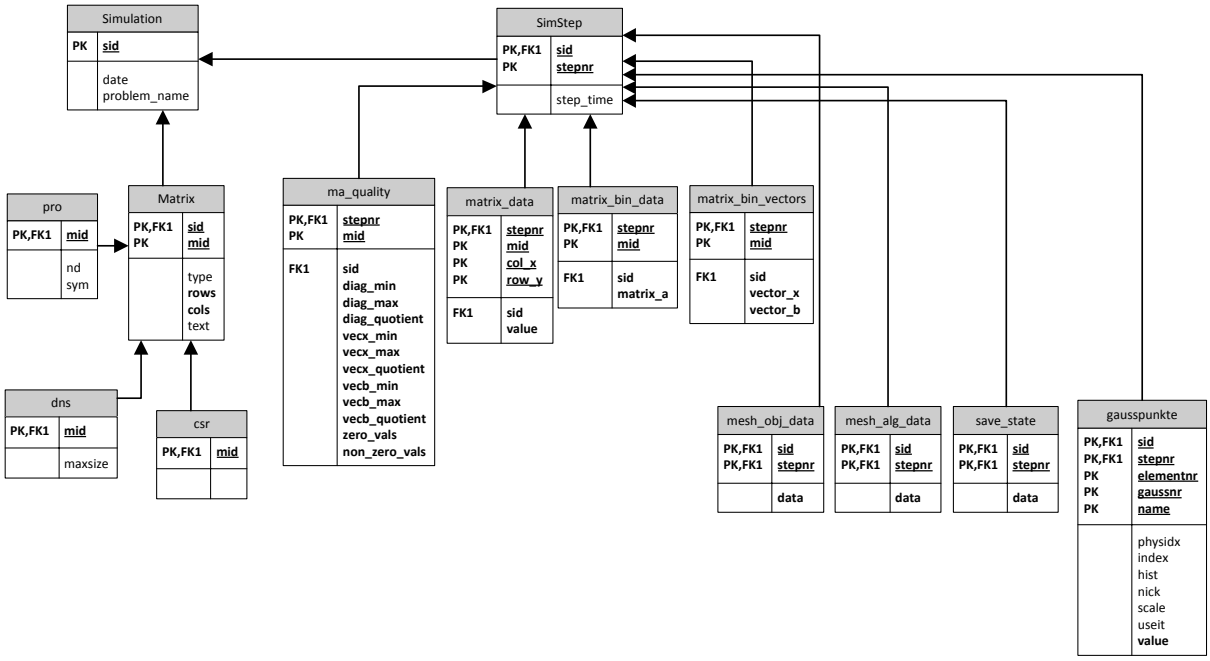

Abbildung 31: Datenbankschema

simulation

Diese Tabelle enthält alle Simlationsinstanzen.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfälle:

- neue Pandas Simulation erstellen

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
date	TIMESTAMP	Zeitpunkt des Simulationsstarts
problem_name	TEXT	Der Name der Simulation

Tabelle 43: Attribute von simulation

simsteps

Diese Tabelle enthält alle Simulationszeitschritte einer referenzierten Simulation.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfälle:

- Einen Simulationszeitschritt ausführen
- Aktuellen Simulationszeitschritt abfragen

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
step_time	TIMESTAMP	Zeitpunkt des Simulationszeitschritts

Tabelle 44: Attribute von simsteps

save_state

Diese Tabelle enthält einen Zustand eines Simulationszeitschrittes von einer Pandas-Instanz.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfälle:

- Zustand speichern
- Zustand laden

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
data	BYTEA	Zustandsdatei Pandas

Tabelle 45: Attribute von save_state

matrix

Diese Tabelle enthält allgemeine Metadaten einer Matrix einer Pandas-Instanz.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern

- Gespeicherte Matrix-ID abfragen

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
mid	INTEGER	Die Matrix-ID
type	TEXT	Der Matrixtyp: Profile, Dense, …
rows	INTEGER	Die Anzahl der Zeilen der Matrix
cols	INTEGER	Die Anzahl der Spalten der Matrix
text	TEXT	textuelle Beschreibung der Matrix – wird nicht genutzt

Tabelle 46: Attribute von matrix

ma_pro

Diese Tabelle ist die Spezialisierung der `matrix`−Tabelle und enthält zusätzliche Attribute bezüglich einer Profile-Matrix.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern

Attributname	Datentyp	Beschreibung
mid	INTEGER	Die Matrix-ID
nd	BIGINT	Die Anzahl der Elemente je Matrixdreieck
sym	BOOLEAN	Flag für Symmetrie

Tabelle 47: Attribute von ma_pro

ma_dns

Diese Tabelle ist die Spezialisierung der `matrix`−Tabelle, und enthält zusätzliche Attribute bezüglich einer Dense-Matrix.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern

Attributname	Datentyp	Beschreibung
mid	INTEGER	Die Matrix-ID
maxsize	BIGINT	Die Anzahl der Elemente der Matrix

Tabelle 48: Attribute von ma_dns

ma_csr

Diese Tabelle ist die Spezialisierung der `matrix`−Tabelle, und enthält zusätzliche Attribute bezüglich einer Compressed-Sparse-Row-Matrix. Da diese Matrix nicht in der Pandas-Testversion vorhanden ist, gibt es in dieser Tabelle keine weiteren Attribute.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern

Attributname	Datentyp	Beschreibung
mid	INTEGER	Die Matrix-ID

Tabelle 49: Attribute von ma_csr

matrix_data

In dieser Tabelle werden die Werte einer Matrix zu einer Pandas-Instanz eines Simulationszeitschrittes gespeichert.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern
- Matrix laden
- Matrix zyklisch speichern

Attributname	Datentyp	Beschreibung
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
mid	INTEGER	Die Matrix-ID
col_x	INTEGER	Die Spalte des Matrixelements
row_y	INTEGER	Die Zeile des Matrixelements
sid	BIGINT	Die Simulations-ID
value	DOUBLE PRECISION	Der Wert des Matrixelements

Tabelle 50: Attribute von matrix_data

matrix_bin_data

In dieser Tabelle wird der komplette Speicherbereich der Matrix zu einer Pandas-Instanz eines Simulationszeitschrittes gespeichert.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern
- Matrix laden

- Matrix zyklisch speichern

Attributname	Datentyp	Beschreibung
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
mid	INTEGER	Die Matrix-ID
sid	BIGINT	Die Simulations-ID
matrix_a	BYTEA	Der komplette Speicherbereich der Matrix

Tabelle 51: Attribute von matrix_bin_data

matrix_bin_vectors

In dieser Tabelle werden die kompletten Speicherbereiche der beiden Vektoren x und b zu einer Pandas-Instanz eines Simulationszeitschrittes gespeichert.

Szenario: DUNE-Matrixlöser

Anwendungsfälle:

- Matrix speichern
- Matrix laden
- Matrix zyklisch speichern

Attributname	Datentyp	Beschreibung
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
mid	INTEGER	Die Matrix-ID
sid	BIGINT	Die Simulations-ID
vector_x	BYTEA	Der komplette Speicherbereich des Vektors x
vector_y	BYTEA	Der komplette Speicherbereich des Vektors b

Tabelle 52: Attribute von matrix_bin_vectors

ma_quality

Diese Tabelle enthält diverse Metriken der Matrixqualität.

Szenario: Data-Quality

Anwendungsfälle:

- Matrixqualität erfassen
- SQL-Query zur Matrixqualität

Attributname	Datentyp	Beschreibung
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
mid	INTEGER	Die Matrix-ID
sid	BIGINT	Die Simulations-ID
diag_min	DOUBLE PRECISION	Der minimale Wert auf der Hauptdiagonalen der Matrix
diag_max	DOUBLE PRECISION	Der maximale Wert auf der Hauptdiagonalen der Matrix
diag_quotient	DOUBLE PRECISION	Der Quotient aus minimalen und maximalen Wert
vecx_min	DOUBLE PRECISION	Der minimale Wert des Vektors x

vecx_max	DOUBLE PRECISION	Der maximale Wert des Vektors x
vecx_quotient	DOUBLE PRECISION	Der Quotient aus minimalen und maximalen Wert des Vektors x
vecb_min	DOUBLE PRECISION	Der minimale Wert des Vektors y
vecb_max	DOUBLE PRECISION	Der maximale Wert des Vektors y
vecb_quotient	DOUBLE PRECISION	Der Quotient aus minimalen und maximalen Wert des Vektors y
zero_vals	BIGINT	Die Anzahl der Matrixeinträge außerhalb der Hauptdiagonalen, die nicht Null sind.
non_zero_vals	BIGINT	Die Anzahl der Matrixeinträge außerhalb der Hauptdiagonalen, die Null sind.

Tabelle 53: Attribute von ma_quality

mesh_alg_data

Diese Tabelle enthält Freiheitsgrade eines Meshes in Form einer MeshFile, die über die interne Pandas Funktion **`MeshSave`** erzeugt wurde.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfälle:

- Freiheitsgrade an allen Mesh-Elementen speichern
- Gespeicherte Freiheitsgrade laden

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
data	BYTEA	MeshFile

Tabelle 54: Attribute von mesh_alg_data

mesh_obj_data

Diese Tabelle enthält alle Objekte eines Meshes in Form einer ObjectFile, die über die interne Pandas Funktion **`BisectSave`** erzeugt wurde. Diese Tabelle wird nur bei einer zweidimensionalen Simulation verwendet.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfälle:

- Freiheitsgrade an allen Mesh-Elementen speichern
- Gespeicherte Freiheitsgrade laden

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
data	BYTEA	ObjectFile

Tabelle 55: Attribute von mesh_obj_data

gausspunkte

Diese Tabelle enthält die Variablen von den Gausspunkten der Mesh-Elemente.

Szenario: Pandas-Matlab Kopplung

Anwendungsfälle:

- Alle Variablen der Gausspunkte speichern
- Alle Variablen der Gausspunkte laden
- Eine Variable der Gausspunkte speichern
- Eine Variable der Gausspunkte laden
- Anzahl der Gausspunkte abfragen
- Anzahl der Elemente abfragen
- Anzahl aller Variablen abfragen
- Matlab Input-Dateien erstellen
- Matlab Output-Datei laden

Attributname	Datentyp	Beschreibung
sid	BIGINT	Die Simulations-ID
stepnr	INTEGER	Der Simulationszeitschritt der Simulation
elementnr	INTEGER	Die Nummer des Mesh-Elementes
gaussnr	INTEGER	Die Nummer des Gausspunktes an einem Mesh-Element
name	TEXT	Der Kurzname der Variable
physidx	INTEGER	Der Index der Variablendefinition in der Physik
index	INTEGER	Elementindex: Index in elem->var bzw. elem->hist
hist	BOOLEAN	Flag, ob es sich um eine History Variable handelt
nick	TEXT	Der Langname der Variable
scale	DOUBLE PRECISION	Skalierung der Variable
useit	BOOLEAN	Ausgabe der Variable
value	DOUBLE PRECISION	Der Wert der Variable

Tabelle 56: Attribute von gausspunkte

5 Implementierung

In diesem Kapitel werden einige Details der Implementierung der Pandasmodifikation beschrieben. Hierzu wird zunächst die Integration des gSOAP Web Service-Servers und die Modifikation an Pandas anhand ihrer Abläufe beschrieben. Anschließend werden die genauen Änderungen an den Modulen und des Projektes aufgezeigt.

Bei der Implementierung des WSI-Pandas-Adapters und der Modifikation an Pandas wurde die Entwicklungsumgebung Eclipse 3.6 verwendet. Die Ausführung der Web Service-Operationen wurde mit dem Web Service Explorer von Eclipse vorgenommen. Während der Implementierung wurden der WSI_Pandas-Adapter und die Modifikation an Pandas aufgrund des modularen Aufbaus manuell getestet.

5.1 Anpassungen an Pandas

Mit dem WSI-Adapter, dem WSI-Pandas-Adapter und der Web Service Erweiterung an Pandas sind nun interaktive Simulationen möglich. Hierzu werden Web Service Anfragen, die an den WSI-Pandas-Adapter gesendet wurden, an die modifizierte Pandas-Anwendung weitergeleitet. Dieser Umweg war nötig, da Pandas selber nicht als Dienst laufen kann und für jedes Simulationsproblem neu kompiliert und gestartet werden muss.

Damit Pandas nach dem Start Web Service Anfragen bearbeiten kann, musste zunächst der Web Service-Server von gSOAP integriert werden. Für das bessere Verständnis werden der ursprüngliche und der modifizierten Startablauf miteinander verglichen.

5.1.1 Ursprünglicher Ablauf

Der ursprüngliche Ablauf von Pandas ist auf der Abbildung 32 zu sehen und wird im Folgenden beschrieben.

1. Der Benutzer startet Pandas, womit zunächst die main-Methode von **`pandas.c`** ausgeführt wird.
2. Die main-Methode aus **`pandas.c`** ruft sofort die **`MainMain`**-Methode aus der **`main.c`** auf.
   Zu Beginn der **`MainMain`**-Methode werden die Kommandozeilenparameter eingelesen
3. Nachdem die Kommandozeilenparameter eingelesen wurden, wird die komplette Pandas-Anwendung initialisiert. Dazu wird aus der **`pandas.c`** die Methode **`AppInitialize`** aufgerufen, die eine komplette Hierarchie von Initialisierungsmethoden diverser Module antriggert. In der Abbildung 32 ist dies mit dem Aufruf der Methode **`MainInit`** beispielhaft angedeutet.
4. Nach der Initialisierung wird die **`MainLoop`**-Methode aufgerufen, worin die Benutzerbefehle entgegengenommen und abgearbeitet werden.

5. Nach Eingabe des Beenden-Kommandos wird die **`MainLoop`** verlassen und alle Ressourcen wieder freigegeben.

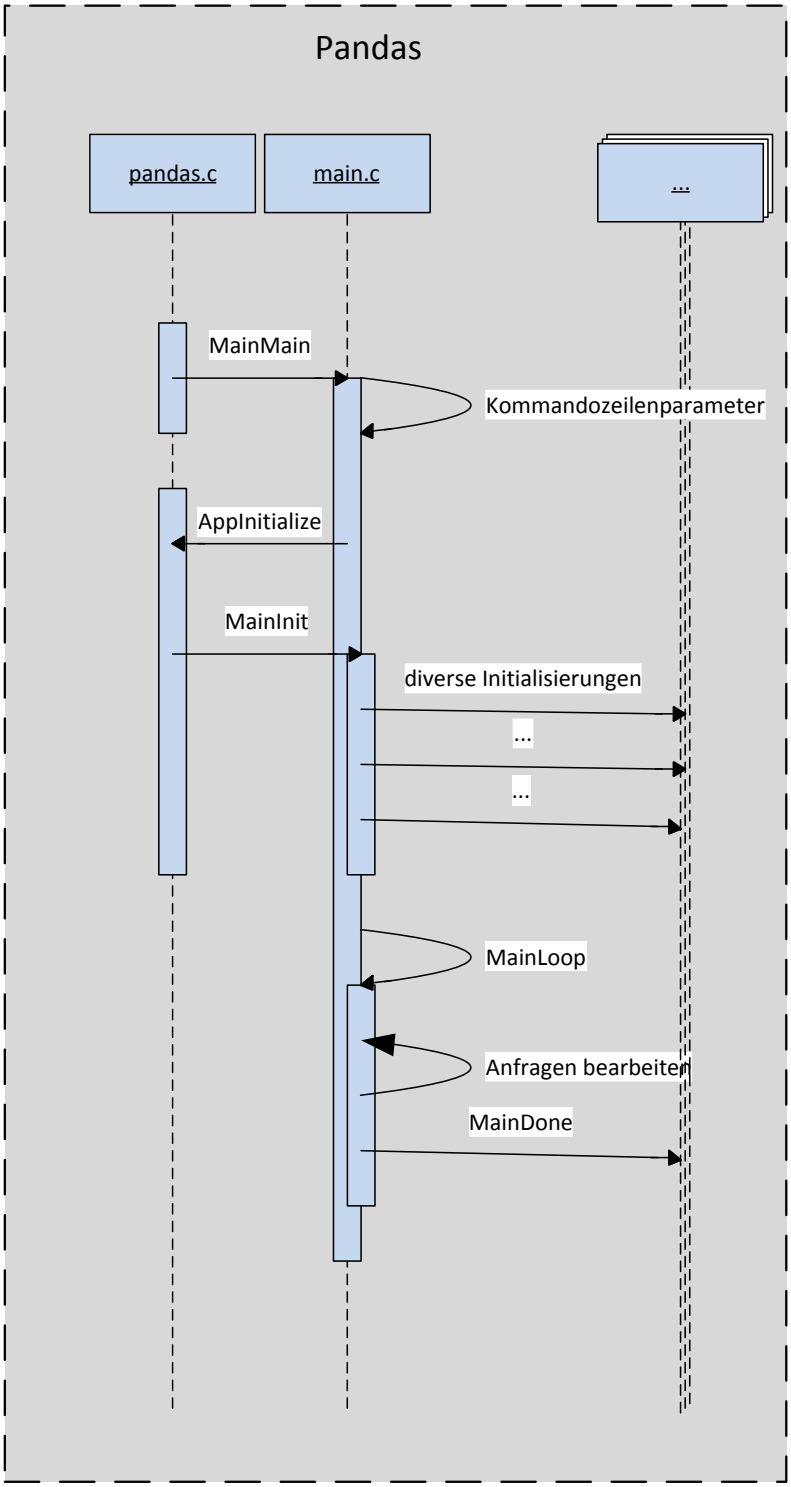

Abbildung 32: Ablauf der Pandas-Anwendung

5.1.2 Modifizierter Ablauf

Für den modifizierten Ablauf von Pandas ist der WSI-Pandas-Adapter nötig, der zunächst über den generischen WSI-Adapter eine neue Simulationsinstanz erzeugt, den Sourcecode von Pandas kompiliert und dann startet. Für die Web Service-Server Integration in Pandas wurde die **`main.c`** angepasst. Dieser modifizierte Ablauf wird in Abbildung 33 gezeigt und im Folgenden beschrieben.

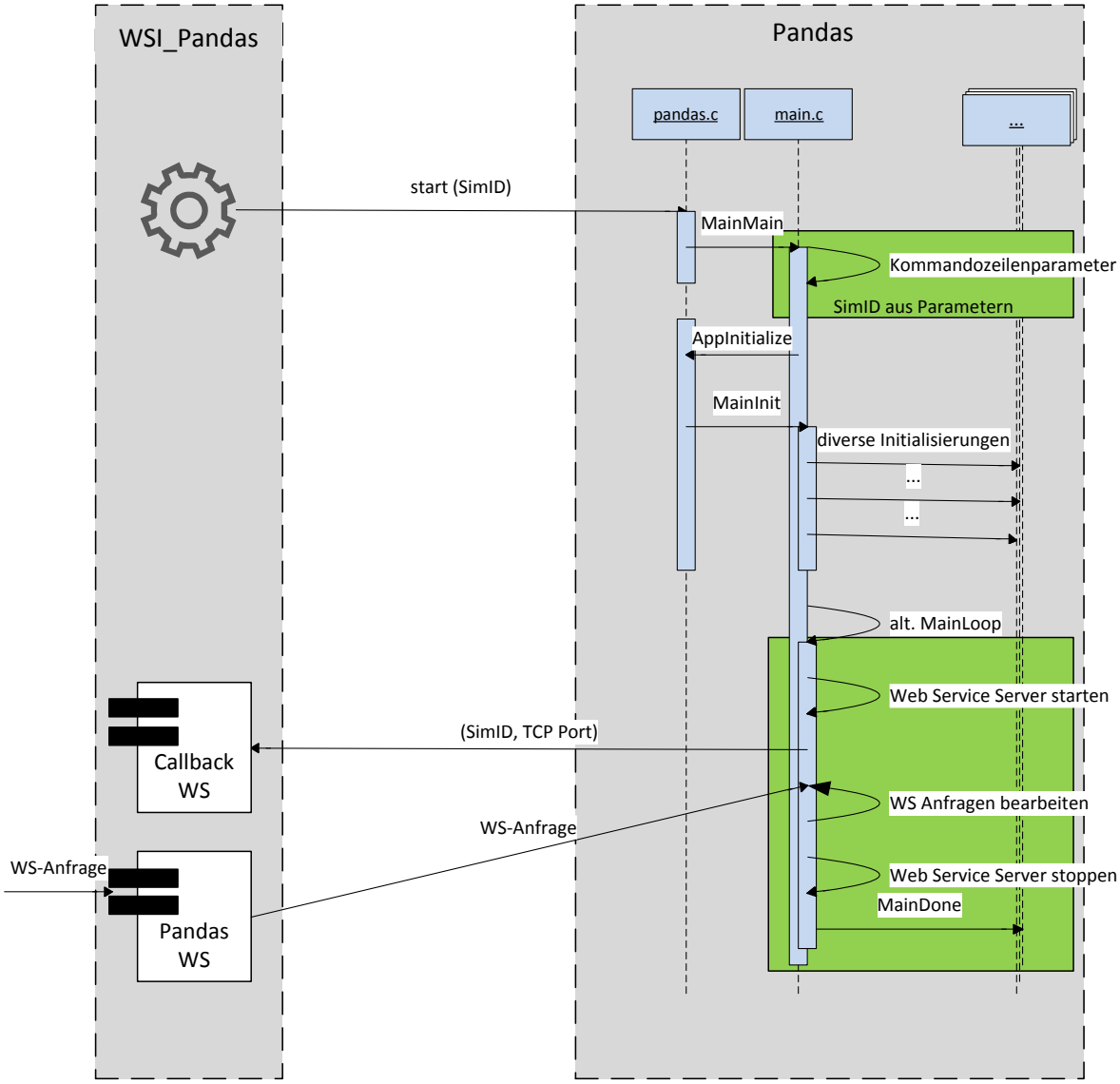

Abbildung 33: Ablauf der Pandas-Anwendung nach der Modifikation

1. Diesmal wird die Pandas-Anwendung über den WSI-Pandas-Adapter gestartet. Hierzu wird, von der zuvor neu erstellten Simulationsinstanz, die SimID als Kommandozeilenparameter übergeben.
2. Wie zuvor, ruft die **`main`**-Methode aus der **`pandas.c`** sofort die **`MainMain`**-Methode aus der **`main.c`** auf. Zu Beginn der **`MainMain`**-Methode wird nun aus den Kommandozeilenparametern die SimID ausgelesen.

3. Die Anwendungsinitialisierung durch die **`AppInitialize`**-Methode ist unverändert geblieben.
4. Da in den Kommandozeilenparametern die SimID übergeben wurde, wird nun nach der Initialisierung eine alternative **`MainLoop`**-Methode aufgerufen
    a. In der alternativen MainLoop-Methode wird zunächst der von gSOAP erstellte Web Service-Server gestartet und an einen freien TCP-Port gebunden. Es können jedoch noch keine Web Service Anfragen beantwortet werden.
    b. Die SimID und der TCP-Port werden über die Callback-Schnittstelle des generischen WSI-Adapters als asynchrone Nachricht geschickt.
    c. Der generische WSI-Adapter speichert den verwendeten TCP-Port bei der entsprechenden Simulationsinstanz ab und setzt den Zustand der Instanz auf *Ready*, wodurch nun der WSI-Pandas-Adapter Web Service Anfragen an Pandas weiterleiten kann.
    d. Der in Pandas integrierte Web Service-Server ist nun bereit Web Service-Anfragen zu bearbeiten und wartet auf eingehende Nachrichten.
    e. Eingehende Nachrichten werden bearbeitet und die Antwort an den WSI-Pandas-Adapter zurückgesendet und anschließend auf die nächste Anfrage gewartet.
    f. Bei dem Aufruf der **`stopApplication`**-Operation des WSI-Pandas-Adapters wird der integrierte Web Service-Server gestoppt und im Anschluss die von Pandas verwendeten Ressourcen wieder freigegeben.

5.1.3 Modifizierter Simulationsablauf

Damit man die Simulation in der Datenbank abbilden kann, war es nötig einen globalen Simulationszeitschrittzähler einzuführen. Dieser Simulationszeitschrittzähler stellt die Speicherungsgranularität in der Datenbank dar. Zusätzlich müssen während der Simulation die Schritte gezählt werden und, während der Berechnung eines Zeitschrittes, der Zugriff auf die Simulations-Daten ermöglicht werden. Hierfür wurde die Methode **`DaeSteps`** modifiziert, da diese von allen Kommandos zum Starten der Berechnung verwendet werden. Die Abbildung 34 zeigt den Ablauf der modifizierten Methode **`DaeSteps`** und wird im Folgenden beschrieben.

1. Die Simulationsberechnung wird über den Aufruf von **`DaeSteps`** gestartet
2. Zu Beginn finden diverse Initialisierungen für die Berechnung statt.
3. Nach den Initialisierungen wird die Schleife zur Ausführung der Simulationszeitschritte betreten.
    a. Ein Hook für Benutzer-Funktionen wird aufgerufen.
    b. Die Berechnung eines Simulationszeitschrittes wird angestoßen.
    c. Es wird geprüft, ob die Simulation schon in der Datenbank initialisiert wurde. Falls dies noch nicht geschehen ist, wird nun eine neue Simulation in der Datenbank angelegt sowie die Informationen zur verwendeten Matrix in der Datenbank abgespeichert.
    d. Nachdem sichergestellt wurde, dass die Simulation in der Datenbank vorhanden ist, wird der aktuelle Simulationszeitschritt in der Datenbank angelegt. Dadurch ist es nun möglich zu diesem Simulationszeitschritt Daten in der Datenbank abzulegen.

e. Wenn zyklisches Speichern der Matrix oder der Matrixqualität eingestellt wurde, wird nun geprüft, ob die Matrix oder die Matrixqualität gespeichert werden muss und unter Umständen gespeichert.
f. Der neue globale Simulationszeitschrittzähler wird erhöht.
g. Danach wird ein etwaiger Benutzerabbruch behandelt und es erfolgen diverse Bildschirmausgaben.
h. Am Ende der Schleife wird der temporär verwendete Speicher wieder freigegeben.

Nachdem die **`DaeSteps`**-Methode durchlaufen wurde, können folgende Daten aus der Simulation in der Datenbank gespeichert werden:

- Matrix
- Matrixqualität
- Pandas-Zustand
- Freiheitsgrade der Mesh-Elemente
- Variablen der Gausspunkte

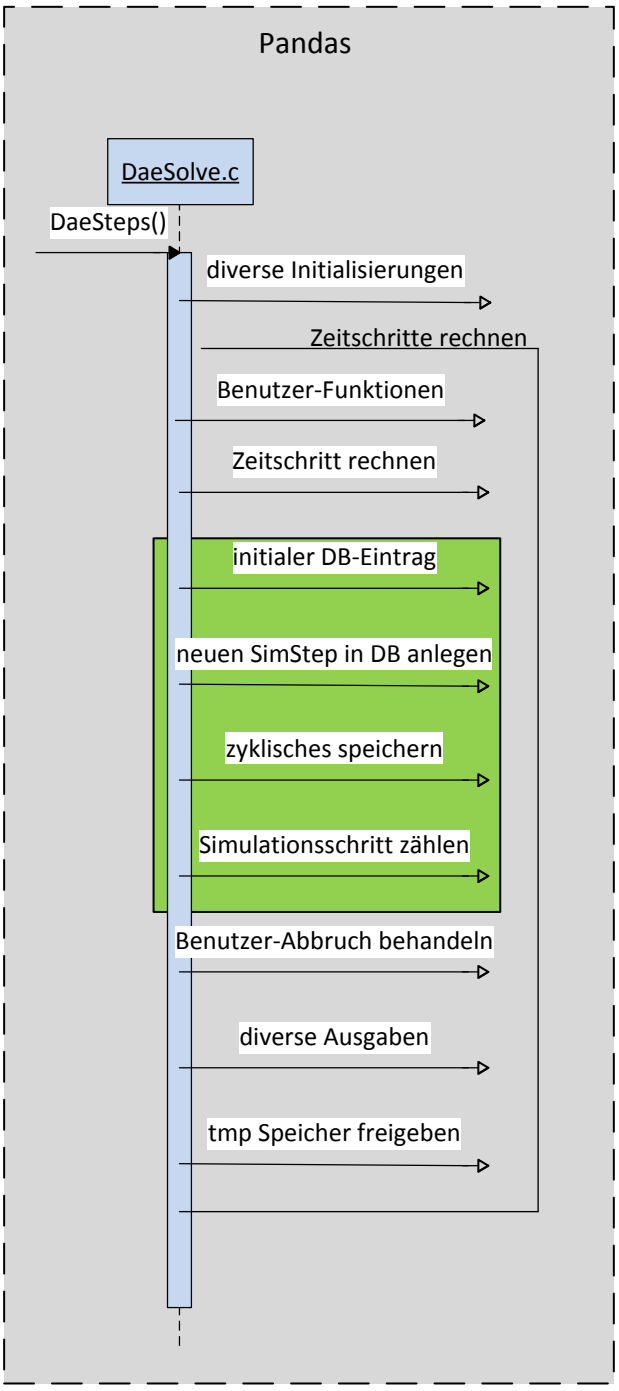

Abbildung 34: Ablauf der geänderten DaeSteps-Methode

5.1.4 Quellcodeänderungen

Die Service Bus Erweiterung an Pandas direkt betraf einerseits den Build-Prozess und andererseits die Module **`main`** und **`daesolve`**. Zusätzlich mussten neue Dateien in die Sourcen mit aufgenommen werden. Im Folgenden werden die betroffenen Stellen beschrieben

5.1.4.1 Build-Prozess

Der Build-Prozess von Pandas musste angepasst werden, damit der generierte Web Service-Server von gSOAP und die OpenDBX-Bibliothek für die Datenbankanbindung eingebunden werden. Da sich die Build-Prozesse von der Test- und der Entwicklerversion unterscheiden, werden beide Änderungen gezeigt.

In der Testversion musste lediglich die Datei `Makefile`, welche in `<Pandas_Home>/src` liegt geändert werden. Zum einen wurden bei der Variablendefinition **`OBJECTS`** die folgenden Dateien hinzugefügt: **`stdsoap2.o soapServer.o soapClient.o soapC.o db.o`**. Zum anderen wurde bei der Variablendefinition **`Netlibs`** die OpenDBX-Bibliothek **`-lopendbx`** hinzugefügt.

In der Entwicklerversion sind die Änderunge im Build-Prozess weitreichender. Hier wurden folgende Dateien geändert:

```
<Pandas_Home>/src/mak32/general.mak
<Pandas_Home>/src/mak64/general.mak
<Pandas_Home>/src/lib/Makefile
```

In der general.mak wurde eine Variable **`SOAP`** definiert und die Variablendefinition **`NETLIBS`** wurde um die OpenDBX-Bibliothek **`-lopendbx`** erweitert.

Die Makefile wurde um folgende Variablen erweitert:

```
SOAPSERVER  = stdsoap2.c soapServer.c soapClient.c soapC.c
HSOAPSERVER = $(SOAPSERVER:.c=.h)
DB      = db.c
HDB     = $(DB:.c=.h)
```

Über die neue Variable **`SOAP`** aus der `general.mak` wird nun in der Makefile entschieden, ob der Web Service-Server und die Datenbankerweiterung mitkompiliert werden.

```
ifeq ($(SOAP), TRUE)
  CFILES    = $(CUTIL) $(DB) $(CCORE) $(CMESH) me_ren.c  $(SOAPSERVER)
wrapper.c
else
  CFILES    = $(CUTIL) $(CCORE) $(CMESH) me_ren.c wrapper.c
endif

ifeq ($(SOAP), TRUE)
  HFILES    = $(HUTIL) $(HDB) $(HCORE) $(HMESH) $(HSOAPSERVER)
else
  HFILES    = $(HUTIL) $(HCORE) $(HMESH)
endif
```

5.1.4.2 main.c

In dieser Datei wurden die Grundlegenden Änderungen vorgenommen. Hierbei wurden für den gSOAP Web Service-Server die Dateien **`soapH.h, namespaces.nsmap`** und **`soapStub.h`** und für die Datenbankfunktionalität die Dateien **`db.h`** und **`odbx.h`** aufgenommen

My_MainLoop (A1)

Die alternative MainLoop Methode, welche den integrierten Web Service-Server startet, dem WSI den verwendeten TCP-Port mitteilt und Web Service Anfragen bearbeitet.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas starten

getLocalPort (A1)

Liefert den verwendeten TCP-Port des integrierten Web Service-Servers zurück

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas starten

__callback__reportStateReady (A1)

Diese Funktion wird nur für die Rückmeldung am WSI benötigt, um diesem den TCP-Port mitzuteilen.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas starten

__pandas__executeCommand (A1)

Führt ein Pandas-Kommando aus, indem der übergebene String an die interne Macro-Ausführung weitergeleitet wird. Da die interne Methode zur Macro-Ausführung keinen Status zurückgibt, wird hierüber immer ein fehlerfreier Status zurückgeliefert.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas Kommando ausführen

__pandas__getStepnr (A1)

Liefert den aktuellen Simulationszeitschritt zurück.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Aktuellen Simulationszeitschritt abfragen

__pandas__set_option (A2, A3)

Hierüber kann man das Verhalten der Matrixspeicherung steuern. Darüber hinaus lässt sich hierüber initial die Datenbank-Tables anlegen.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix zyklisch speichern

__pandas__stopApplication (A1)

Sofern Pandas mit einer Datenbank verbunden ist, wird diese Verbindung getrennt und anschließend wird Pandas beendet.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Pandas beenden

__pandas__do_step (A1)

Hierüber wird ein Simulationszeitschritt ausgeführt. Dazu wird intern der Befehl **step 1** ausgeführt.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Einen Simulationszeitschritt ausführen

__pandas__connect_db (A1, A2)

Stellt eine Verbindung zu einer Datenbank her.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: DB-Verbindung aufbauen

__pandas__disconnect_db (A1, A2)

Trennt eine bestehende Verbindung zu einer Datenbank.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: DB-Verbindung trennen

__pandas__save_state (A1, A2)

Speichert den Zustand von Pandas in der Datenbank. Dabei wird zunächst von Pandas der Befehl **save** verwendet um lokal eine Zustandsdatei zu erzeugen. Diese Zustandsdatei wird anschließend in der Datenbank abgelegt.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Zustand speichern

__pandas__load_state (A1, A2)

Lädt einen Zustand von Pandas aus der Datenbank. Hier wird zunächst die Zustandsdatei aus der Datenbank lokal erstellt und anschließend von Pandas über den Befehl **load** geladen.

Szenario: Pandas Service-Bus-Adapter

Anwendungsfall: Zustand laden

__pandas__save_dof (A2, A3)

Speichert die Freiheitsgrade der Mesh-Elemente ab. Dazu wird die Pandas-Funktion **MeshSave** verwendet, welche lokal eine Datei speichert. Diese Datei wird anschließend in der Datenbank abgelegt. Im 2-dimensionalen Fall wird zusätzlich noch die Pandas-Funktion **BisectSave** verwendet, damit beim Laden die Meshkoordinaten überprüfen zu können. Auch hier wird wieder lokal eine Datei erstellt, welche in der Datenbank abgelegt wird.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfall: Freiheitsgrade an allen Mesh-Elementen speichern

__pandas__load_dof (A2, A3)

Lädt die Freiheitsgrade der Mesh-Elemente aus der Datenbank. Im 2-dimensionalen Fall wird zunächst aus der Datenbank die von **BisectSave** erzeugte Datei lokal erstellt und temporär mittels **BisectLoad** geladen. Anschließend wird überprüft, ob die Koordinaten des aktuellen und des gespeicherten Meshes passen. Wenn die Koordinaten übereinstimmen, so wird aus der Datenbank die von **MeshSave** erzeugte Datei lokal erstellt und mit **MeshLoad** geladen.

Szenario: Mehrere Pandas-Instanzen

Anwendungsfall: Gespeicherte Freiheitsgrade laden

__pandas__save_matrix (A2, A3)

Speichert die Matrix von Pandas ab. Die Matrix kann dabei entweder werteweise oder der gesamte Speicherbereich binär in der Datenbank abgespeichert werden.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix speichern

__pandas__load_matrix (A2, A3)

Lädt die binär in der Datenbank abgelegte Matrix von Pandas.

Szenario: DUNE-Matrixlöser

Anwendungsfall: Matrix laden

__pandas__saveAllGauss (A2, A3)

Speichert alle Variablen von allen Gausspunkten an allen Elementen.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Alle Variablen der Gausspunkte speichern

__pandas__loadAllGauss (A2, A3)

Lädt aus der Datenbank alle Variablen von allen Gausspunkten an allen Elementen.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Alle Variablen der Gausspunkte laden

__pandas__saveGaussName (A2, A3)

Speichert eine angegebene Variable von allen Gausspunkten an allen Elementen.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Eine Variable der Gausspunkte speichern

__pandas__loadGaussName (A2, A3)

Lädt aus der Datenbank eine angegebene Variable von allen Gausspunkten an allen Elementen.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Eine Variable der Gausspunkte laden

__pandas__saveDataQuality (A2, A3)

Erfasst die aktuelle Matrix Qualität und legt diese in der Datenbank ab.

Szenario: Pandas-Matlab Kopplung

Anwendungsfall: Matrixqualität erfassen

__pandas__getDataQualityQuery (A3)

Gibt ein Query zur zuletzt abgespeicherten Matrix Qualität zurück.

Szenario: Data-Quality

Anwendungsfall: SQL-Query zur Matrixqualität

__pandas__getLastSavedStepnr (A3)

Gibt den Simulationszeitschritt zurück, bei dem zuletzt die Matrixqualität erhoben wurde.

Szenario: Data-Quality

Anwendungsfall: Zeitschritt der letzten Matrixqualität

__pandas__getMid (A3)

Liefert die Matrix-ID dieser Simulation zurück.

Szenario: Data-Quality

Anwendungsfall: Gespeicherte Matrix-ID abfragen

5.1.4.3 daesolve.c (A2)

Die Methode DaeSteps musste um folgende Parameter erweitert werden, damit der Simulationsverlauf in der Datenbank abgebildet werden kann.

Parametername	Typ	Beschreibung
SID	Long	globale Simulations-ID
mid	Integer	globale Matrix-ID
step_counter	Integer	globaler Simulationszeitschrittzähler
step_width	Integer	Schrittweite für zyklisches speichern der Matrix oder Matrixqualität
init	Boolean	wurde die Simulation schon in der Datenbank angelegt
db_state	DBstates	Handle für die Datenbank-Verbindung

Tabelle 57: zu der Methode DaeSteps hinzugefügte Parameter

5.1.4.4 Hinzugefügte Dateien (A1)

In der Testversion wurden in das Verzeichnis `<Pandas_Home>/src` und in der Entwicklerversion in das Verzeichnis `<Pandas_Home>/src/lib` neue Dateien für den Web Service-Server hinzugefügt.

Datei	Beschreibung
namespace.nsmap	Enthält die XML-Namespace-Definitionen
soapC.c soapClient.c soapH.h soapServer.c soapStub.h	Enthalten die aus dem gSOAP-Header generierten Structs und C-Funktionen, welche für die Skeletons und Stubs benötigt werden.
stdsoap2.c stdsoap.h	Enthalten den Web Service-Server von gSOAP.
db.c db.h	Enthalten für Pandas Datenbank-Funktionen, welche z.B. die Datenbank-Verbindung herstellen oder Queries absetzt.

Tabelle 58: hinzugefügte Dateien

6 Pandas basierte Workflows

In jedem Pandas Workflow gibt es zu Beginn und am Ende immer die gleichen Abläufe, welche die Simulation erstellen bzw. beenden. Diese wiederkehrenden Abläufe werden hiermit einmal erläutert und in den folgenden Workflows nur noch als *Pandas Prepare Steps* bzw. als *Pandas Post Processing* genannt.

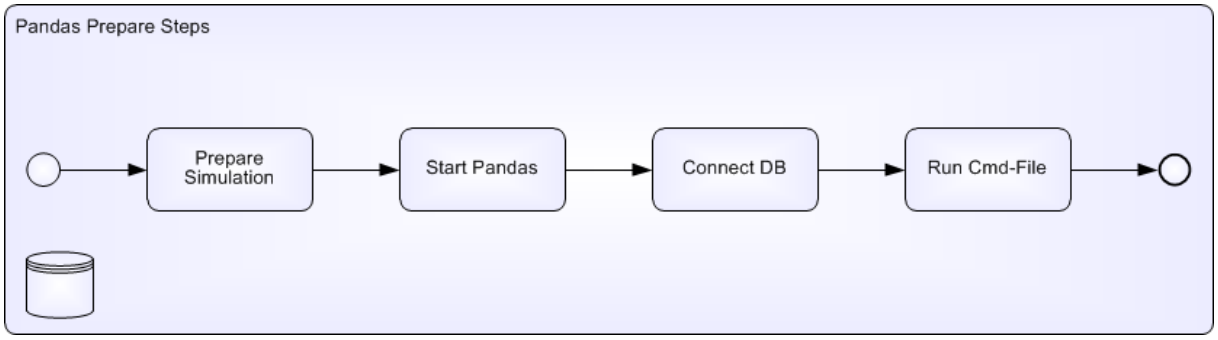

Abbildung 35: Pandas Prepare Steps

Auf der Abbildung 35 ist das *Pandas Prepare Steps* Workflowfragment zu sehen. Hier wird als erste Aktion die Operation *PrepareSimulation* aufgerufen, welche eine Simulationsinstanz erzeugt, den Pandas Sourcecode und die Simulationsdaten entpackt und dieses dann kompiliert. Nachdem Pandas erfolgreich kompiliert wurde, wird es mit der aktuellen Simulations-ID als Startparameter gestartet. Nachdem Pandas gestartet wurde und der integrierte Web Service-Server sich bei dem WSI_Pandas Adapter zurückgemeldet hat, kann es nun direkt Web Service Anfragen bearbeiten. Als erste eingehende Web Service Anfrage wird die Verbindung zu einer Datenbank etabliert. Als abschließende Aktion wird ein Initialisierungsfile von Pandas ausgeführt, welche mehrere simulationsspezifische Einstellungen vornimmt. Nach der Ausführung der Initialisierungsfile ist Pandas bereit die Simulation auszuführen.

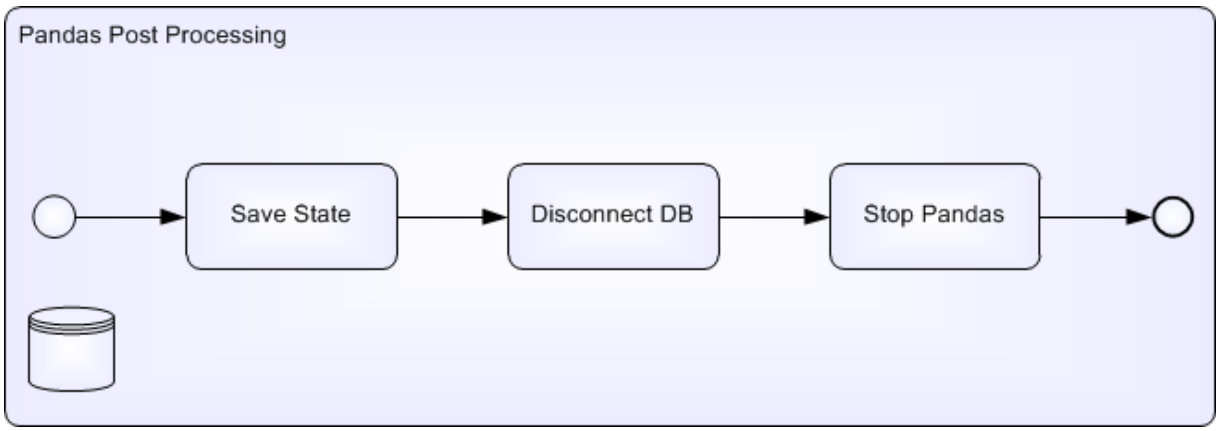

Abbildung 36: Pandas Post Processing

Auf der Abbildung 36 ist das *Pandas Post Processing* Workflowfragment zu sehen. Nachdem die Simulation durchgelaufen ist, wird zunächst der Endzustand der Simulation in der Datenbank gespeichert. Wenn der Zustand gespeichert ist, wird die Verbindung zu der Datenbank getrennt und abschließend wird Pandas beendet.

6.1 Data-Quality (A3)

Der Workflow zu dem *Data-Quality* Szenario testet den WSI_Pandas Web Service zusammen mit dem Data Quality Framework aus der Diplomarbeit [26]. Hierbei wurde im Workflow ein vom Data Quality Framework erstellter Wrapper des WSI_Pandas verwendet. Durch den Wrapper ist es dem Data-Quality-Framework möglich, die Ergebnisse aus der Simulation auszuwerten und gegebenenfalls grafisch darzustellen. Konkret wird die Parameter-File mit ihren Materialeigenschaften und die Werteverteilung auf der Lösungsmatrix von Pandas ausgewertet. In diesem Workflow wird einmal zu Beginn die Materialeigenschaften ausgewertet und dann während der Simulation zyklisch die Matrixqualität erfasst.

Der auf der Abbildung 37 zu sehende DataQuality Workflow führt zunächst das *Pandas Prepare Steps*-Workflowfragment aus, wobei zusätzlich noch eine zyklische Speicherung der Matrixqualität eingestellt wird. Nachdem Pandas erfolgreich seine Vorbereitungsschritte ausgeführt hat, wird der Pfad, der verwendeten Parameterfile, abgefragt und von dem Data Quality Framework analysiert. Als nächstes wird die Simulationsschleife betreten. Innerhalb der Schleife wird zunächst die angegebene Anzahl von Simulationszeitschritten ausgeführt und automatisch die Matrixqualität erfasst. Nachdem die Simulationszeitschritte ausgeführt wurden, wird die SQL-Query zur zuletzt gespeicherten Matrixqualität angefordert. Diese SQL-Query wird von dem Wrapper an das Data Quality Framework gesendet, welches damit die Matrixqualität abfragen, weiterverarbeiten und grafisch darstellen kann. Am Ende der Schleife wird überprüft, ob die Simulation das Ende erreicht hat. Wenn die Simulation ihr Ende erreicht hat wird abschließend das *Pandas Post Processing*-Workflowfragment ausgeführt.

Aus Provenance Gründen bleiben die verwendeten Instanz-Verzeichnisse und deren Inhalt erhalten.

Die WSDL des Workflows ist im Anhang **WSDL Data-Quality** zu finden. Der BPEL-Workflow ist auf der beigelegten DVD enthalten.

Die Startparameter des Data-Quality Workflows sind in der Tabelle 59 aufgelistet:

Parametername	Datentyp	Beschreibung
ProblemName	String	Bestimmt welches Simulationsproblem verwendet werden soll.
CmdFile	Integer	Bestimmt welche Cmd Datei zur Pandasinitialisierung verwendet werden soll
StepWidth	Integer	Bestimmt in welcher Schrittweite immer die Matrix Qualität erfasst wird

Tabelle 59: Startparameter des DataQuality Workflow

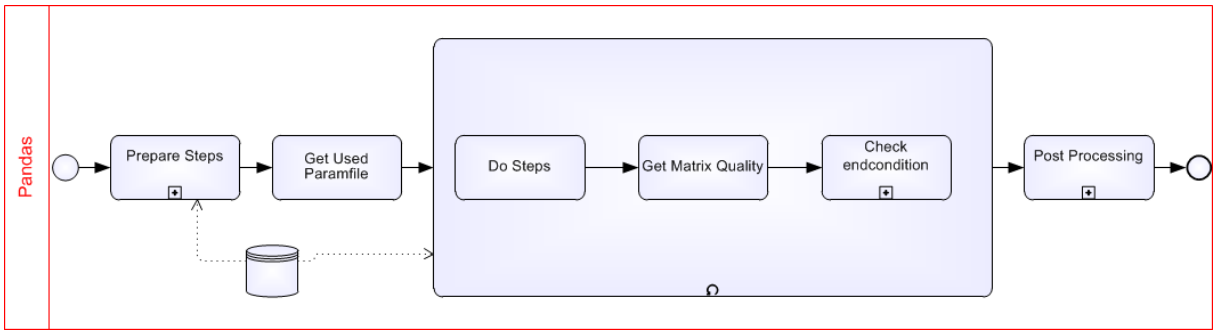

Abbildung 37: DataQuality Workflow

6.2 Simulation mit mehreren Pandas-Instanzen (A5)

Bei dem Workflow zu dem Szenario *Mehrere Pandas-Instanzen* wird der normale WSI_Pandas Web Service Adapter verwendet. Hierbei wird eine Simulation über zwei Pandas Instanzen geführt, indem beide Instanzen ein Objekt abwechselnd mit unterschiedlichen Physiken und Zeitintervallen berechnen. Der Datenaustausch erfolgt hierbei über die Datenbank. Konkret wird hier ein Knochen simuliert, bei dem die Belastungen und die Belastungsdauer variieren.

Bei dem Workflow auf Abbildung 38 werden gemäß des obigen *Pandas Prepare Steps*-Workflowfragments beide Pandas Instanzen erstellt und vorbereitet. Nach der Vorbereitung wird die Simulationsschleife betreten, in der sich beide Instanzen mit der Simulation abwechseln. Dabei beginnt die Pandas Instanz A seine Anzahl an Simulationszeitschritten zu rechnen und speichert dann im Anschluss die Freiheitsgrade der Mesh-Elemente in der Datenbank ab. Die Pandas Instanz B lädt im Anschluss die Freiheitsgrade der Instanz A aus der Datenbank und rechnet mit den geänderten Werten seine Anzahl an Simulationszeitschritten weiter. Nachdem die Pandas Instanz B mit seiner Berechnung fertig ist, werden dessen Freiheitsgrade in der Datenbank gespeichert und daraufhin gleich wieder von Instanz A geladen. Am Ende der Schleife wird überprüft, ob das Ende der Simulation, welches in dem Initialisierungsfile definiert wurde, erreicht wurde. Nachdem das Ende erreicht wurde, wird abschließend das *Pandas Post Processing*-Workflowfragment für beide Instanzen ausgeführt.

Aus Provenance Gründen bleiben die verwendeten Instanz-Verzeichnisse erhalten.

Die WSDL des Workflows ist im Anhang **WSDL TwoInstances** zu finden. Der BPEL-Workflow ist auf der beigelegten DVD enthalten.

Die Startparameter des TwoInstances Workflow sind in der Tabelle 60 aufgelistet:

Parametername	Datentyp	Beschreibung
Problem-SimA	String	Bestimmt welches Simulationsproblem für die Pandas-Instanz A verwendet werden soll.
CmdFile-SimA	String	Bestimmt welche Cmd Datei zur Initialisierung für die Pandas-Instanz A verwendet werden soll
StepWidth-SimA	Integer	Bestimmt wie viele Simulationszeitschritte von der Pandas-Instanz A gerechnet werden sollen
Problem-SimB	String	Bestimmt welches Simulationsproblem für die Pandas-Instanz B

		verwendet werden soll.
CmdFile-SimB	String	Bestimmt welche Cmd Datei zur Initialisierung für die Pandas-Instanz B verwendet werden soll
StepWidth-SimB	Integer	Bestimmt wie viele Simulationszeitschritte von der Pandas-Instanz B gerechnet werden sollen

Tabelle 60: Startparameter des TwoInstances Workflow

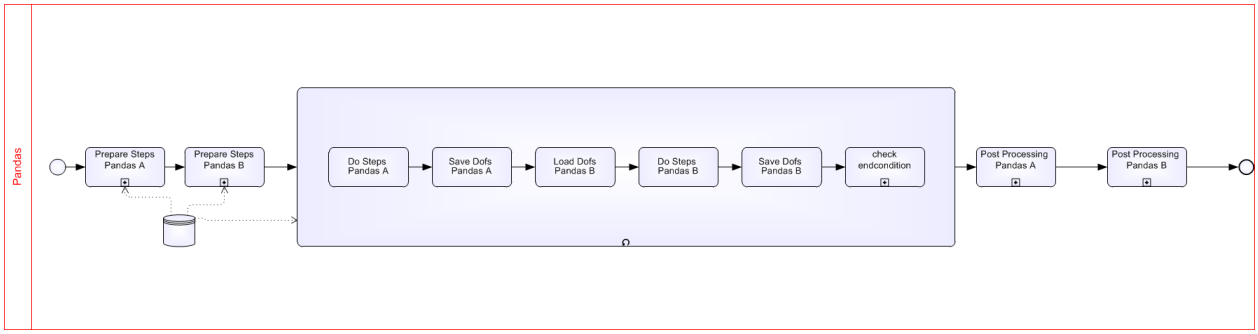

Abbildung 38: TwoInstances Workflow

6.3 Pandas-Matlab Kopplung (A3, A5)

Bei dem Szenario *Pandas-Matlab* werden die Strukturänderungen im Knochen simuliert. Dabei führen Pandas und Matlab gemeinsam die Simulation aus. Pandas berechnet hierbei das biomechanische Modell, die Belastungsverteilung im Organ oder Gewebe, und Matlab das systembiologische Modell, ein Zelleninteraktionsmodell mit Signalmolekülen.

Die Simulation wird über drei Workflows ausgeführt: Pandas-Bone, Data-Manager und Matlab-Bone. Der Workflow Pandas-Bone führt die Pandas-Simulation aus. Der Matlab-Bone Workflow führt die Matlab-Simulation aus und wird von Pandas-Bone über den Data-Manager Workflow angetriggert. Der Data-Manager Workflow hingegen koordiniert die Kommunikation und Datenaustausch zwischen Pandas und mehreren Matlab Instanzen. In den Workflows finden folgende Web Service Adapter Verwendung: WSI_Pandas, WSI_Matlab und WSI_PMConnector. Der Datenaustausch zwischen Pandas und Matlab erfolgt über die Datenbank und CSV-Dateien. Pandas speichert seine Daten in der Datenbank ab und der WSI_PMConnector erstellt aus der Datenbank eine CSV-Datei, welche von Matlab eingelesen werden kann. Matlab hingegen speichert sein Ergebnis wieder in einer CSV-Datei ab, die wiederum der WSI_PMConnector in der Datenbank speichert.

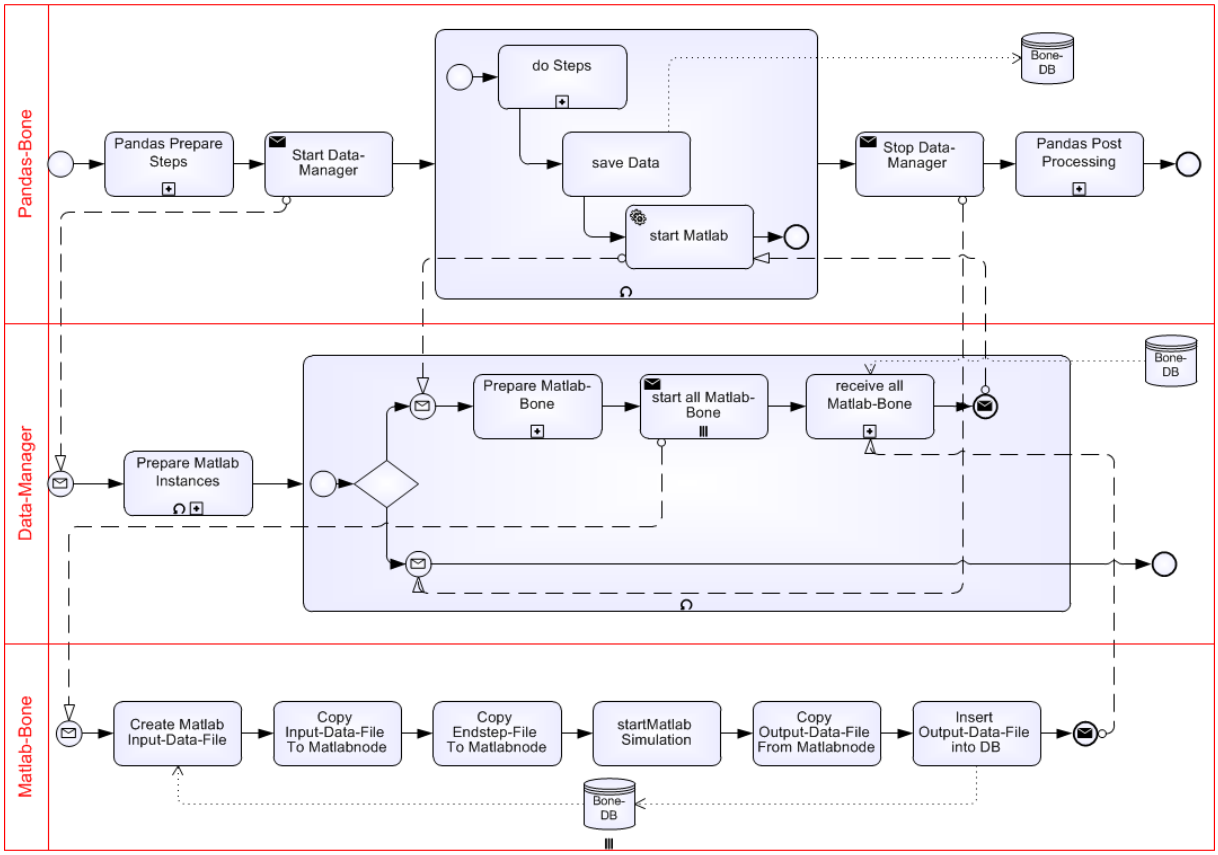

Abbildung 39: Workflows Strukturänderungen im Knochen

Auf Abbildung 39 ist das Zusammenspiel der drei Workflows zu sehen und wird nun exemplarisch durchgegangen. Die Simulation wird über den Pandas-Bone Workflow in Gang gesetzt. Nachdem Pandas-Bone gestartet wurde und Pandas über das Workflowfragment *Pandas Prepare Steps* vorbereitet wurde, wird zunächst der Data-Manager Workflow über die process-Operation gestartet. Durch das Starten des Data-Manager Workflows wird zunächst eine Schleife zur Vorbereitung der Matlab Instanzen gestartet. Auf der Abbildung 40 ist diese Vorbereitungsschleife zu sehen. Die erste Aktion *Create Matlab Instance* erzeugt eine Simulationsinstanz für Matlab und entpackt darin die benötigten Simulationsdateien. Da Matlab auf unterschiedlichen Rechnern ausgeführt wird und da auf jedem Rechner Matlab wo anders liegen kann, wird als nächste Aktion für diese Instanz der Matlab Aufruf festgelegt. Danach wird auf dem Rechner, auf dem Matlab ausgeführt wird, ein lokales Simulationsverzeichnis erstellt. Anschließend wird in dieses Verzeichnis das von Matlab auszuführende M-File kopiert. Als letzter Schritt in der Vorbereitungsschleife werden Initialisierungsdateien in dieses Verzeichnis kopiert. Diese Schleife wird so oft ausgeführt wie es Matlab-Rechner gibt.

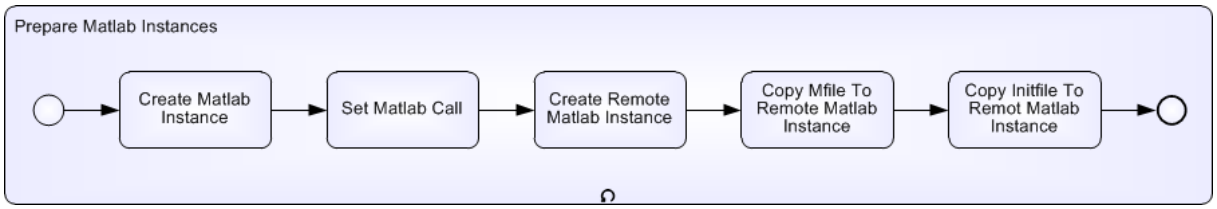

Abbildung 40: Prepare Matlab Instances

Nachdem alle Matlab Instanzen vorbereitet wurden, kommt der Pandas-Bone Workflow in seine Simulationsschleife und der Data-Manager Workflow wartet auf weitere Interaktionen. Jetzt werden von Pandas Simulationszeitschritte berechnet und anschließend seine Daten in der Datenbank gespeichert. Nachdem die Daten in der Datenbank abgespeichert wurden, wird über die Data-Manager Operation startMatlabSim die Matlab Berechnung angetriggert.

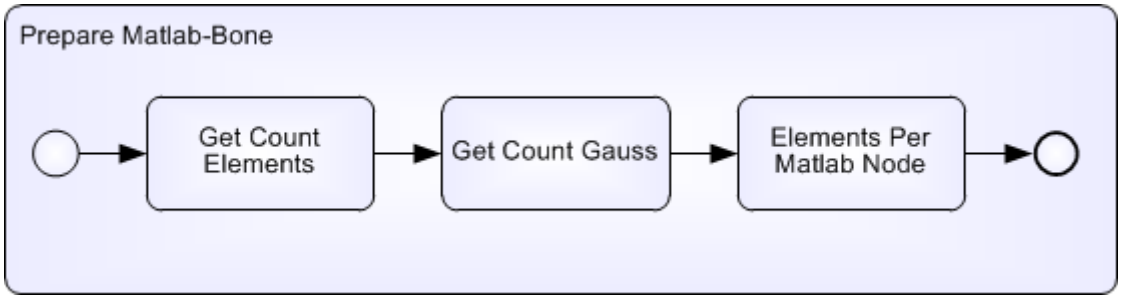

Abbildung 41: Prepare Matlab-Bone

Bevor die Matlab Berechnung anfangen kann, wird zunächst die Größe der zu verteilenden Datenmenge bestimmt. Dies geschieht über das Workflowfragment *Prepare Matlab-Bone* und ist auf der Abbildung 41 zu sehen. Hierbei werden aus den zuvor von Pandas abgespeicherten Daten die Anzahl der Elemente und die Anzahl der Gausspunkte je Element bestimmt. Die Anzahl der Elemente, die jeder Matlab-Rechner berechnen muss, wird durch die Anzahl der Matlab Instanzen und der Anzahl der aller Elemente bestimmt. Danach wird für jede Matlab Instanz eine Matlab-Bone Workflow Instanz erzeugt und anschließend auf alle Matlab-Bone Ergebnisse gewartet.

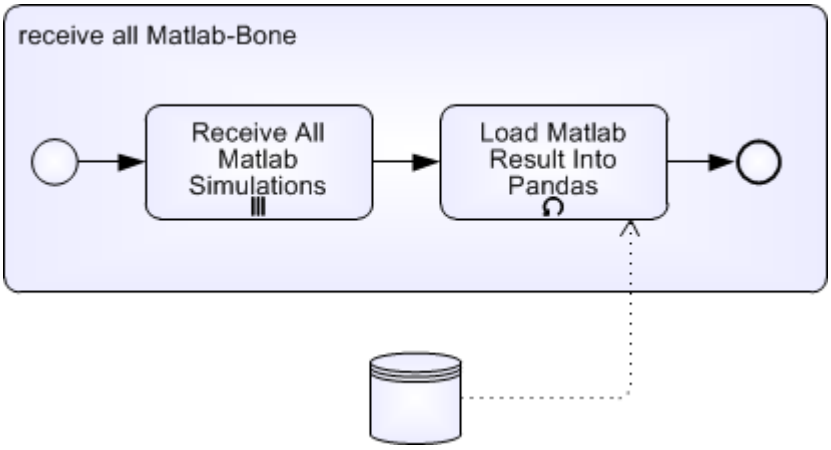

Abbildung 42: receive all Matlab-Bone

Beim Matlab-Bone Workflow werden zuerst über den WSI_PMConnector die Eingabedateien für Matlab in dem Matlab-Instanzverzeichnis erstellt. Nachdem die Eingabedateien erstellt wurden, werden diese auf den entfernten Matlab-Rechner kopiert. Nun wird die Matlab Berechnung gestartet, indem der Kommandozeilenaufruf zur direkten Ausführung eines M-File verwendet wird. Nach Beendigung der Matlab Berechnung steht das Ergebnis in Form einer Ausgabedatei zur Verfügung. Diese Ausgabedatei wird danach von dem Matlab-Rechner zurück in das Matlab Instanz Verzeichnis kopiert. Wenn sich die Ausgabedatei wieder im Matlab Instanz Verzeichnis befindet, wird diese von dem WSI_PMConnector eingelesen und in der Datenbank gespeichert.

Wie auf Abbildung 42 zu sehen ist, werden, nachdem alle Matlab-Bone Instanzen zurückgekehrt sind, alle Teilergebnisse von Matlab in Pandas geladen. Im Anschluss wird wieder zum Pandas-Bone Workflow zurückgekehrt und es wird überprüft, ob mit dem nächsten Simulationszeitschritt begonnen wird oder ob die Simulation beendet wird.

Wenn die Simulation ihr Ende erreicht hat, so wird zunächst die stop-Operation des Data-Manager Workflows aufgerufen, womit sich der Data-Manager Workflow beendet. Nachdem der Data-Manager Workflow beendet wurde, wird Abschließend das *Pandas Post Processing*-Workflowfragment ausgeführt.

Aus Provenance Gründen bleiben die verwendeten Instanz-Verzeichnisse und deren Inhalt erhalten.

Die WSDL-Dateien der Workflows sind in den Anhängen **WSDL Pandas-Bone**, **WSDL Data-Manager** und **WSDL Matlab-Bone** zu finden. Die BPEL-Workflows sind auf der beigelegten DVD enthalten.

In den nachfolgenden Tabellen sind die Parameter der Operationen der drei Workflows aufgelistet.

Parametername	**Datentyp**	**Beschreibung**
PandasData	ComplexType	enthält die Daten zur Pandas-Simulation
ProblemName	String	der Name des Simulationsarchives
CmdFile	String	der Name des Initialisierungsfiles
Step_Width	Integer	die Simulationszeitschrittweite für Pandas
End_Step	Integer	das Ende der Simulation
MatlabData	ComplexType	enthält die Daten zur Matlab-Simulation
MatlabNodes	ComplexType	enthält die Zugriffsdaten

User	String	der Benutzername am Matlabrechner
Host	String	der Hostname des Matlabrechners
SimPath	String	der Simulationspfad auf dem Matlabrechner
MatlabPath	String	der zu verwendende Matlab Aufruf
SimData	ComplexType	enthält die Daten für die Matlab Simulation
SrcArchive	String	der Name des Matlab Archives
mFile	String	der Name des M-Files
t_end	Integer	die Anzahl der Matlab Simulationszeitschritte
WSINodeData	ComplexType	Daten bezüglich des WSI-Rechners
User	String	der Benutzername am WSI Rechner
Host	String	der Hostname des WSI Rechners

Tabelle 61: Parameter der process-Operation des Pandas-Bone Workflows

Parametername	Datentyp	Beschreibung
SimID-Pandas	Long	Die Simulations-ID von der Pandas Instanz
MatlabNodes	ComplexType	enthält die Daten bezüglich des Matlab Rechners
User	String	der Benutzername am Matlab Rechner
Host	String	der Hostname des Matlab Rechners
SimPath	String	der Simulationsinstanzpfad auf dem Matlab Rechner
MatlabPath	String	der zu verwendende Matlab Aufruf
SimNode	ComplexType	enthält die Daten bezüglich des WSI Rechners
User	String	der Benutzername am WSI Rechner
Host	String	der Hostname des WSI Rechners
SimData	ComplexType	enthält die Daten für die Matlab Simulation
SrcArchive	String	der Name des zu verwendenden Matlab Archives
mFile	String	der Name der zu verwendenden M-File

Tabelle 62: Parameter der Initiate-Operation des Data-Manager Workflows

Parametername	Datentyp	Beschreibung
SimIDPandas	Long	Die Simulations-ID von der Pandas Instanz
t_end	Integer	die Anzahl der Matlab-Simulationszeitschritte
Stepnr	Integer	aus welchem Simulationszeitschritt die Variablen der Gausspunkte stammen

Tabelle 63: Parameter der startMatlabSim-Operatioon des Data-Manager Workflows

Parametername	Datentyp	Beschreibung
SimID-Pandas	Long	Die Simulations-ID von der Pandas Instanz

Tabelle 64: Parameter der stop-Operation des Data-Manager Workflows

Parametername	Datentyp	Beschreibung
SimID	Long	Die Simulations-ID der Matlab Instanz
FromSimID	Long	Die Simulations-ID von Pandas, von der die Variablen der Gausspunkte stammen
StartElement	Integer	Bei welchem Mesh-Element die Liste der Variablen der Gausspunkte beginnt

EndElement	Integer	Bei welchem Mesh-Element die Liste endet
Gaussount	Integer	Die Anzahl der Gausspunkte pro Mesh-Element
Stepnr	Integer	aus welchem Simulationszeitschritt die Variablen der Gausspunkte stammen
t_end	Integer	die Anzahl der Matlab-Simulationszeitschritte
mSimPath	String	der Pfad der Simulationsinstanz auf dem Matlabrechner
mFile	String	der Name der auszuführenden M-File
mUser	String	der Benutzername auf Matlabrechner
mHost	String	der Hostname des Matlabrechners
wUser	String	der Benutzername auf dem WSI Rechner
wHost	String	der Hostname des WSI Rechners

Tabelle 65: Parameter der Initiate-Operation des Matlab-Bone Workflows

7 Laufzeitumgebung

Im Rahmen dieser Diplomarbeit wurde eine Laufzeitumgebung erstellt, auf welcher die Pandas Testversion lauffähig ist, und welche die notwendige Software für die Pandas Service-Bus-Erweiterung enthält. Diese Umgebung wurde mit Oracle‘s Virtualbox 4.0 erstellt und als Appliance im Open Virtualization Format (OVF) exportiert. Aus Gründen der Portabilität und der Unabhängigkeit der tatsächlich zugrunde liegenden Hardware, wurde die Laufzeitumgebung vollständig virtualisiert und auf Formen wie z.B. der Paravirtualisierung verzichtet.

Die Appliance der Laufzeitumgebung besteht aus drei Dateien. Die ovf-Datei enthält die Beschreibung der virtualisierten Hardware-Komponenten und die vmdk-Datei enthält das Festplattenabbild der virtuellen Umgebung. Die mf-Datei wiederum enthält Fingerprints über die Appliance, um sicherzustellen dass die ausgelieferten Dateien der Appliance zusammenpassen und nicht verändert wurden.

7.1 Aufbau und Benutzung

Für die Laufzeitumgebung wurde das Betriebssystem Ubuntu 8.04 benötigt, da hierauf die Pandas Testversion lauffähig ist. Bei der Erstellung der virtuellen Maschine, wurde das von VirtualBox bereitgestellte Hardware Profil für das Ubuntu Betriebssystem verwendet.

HW-Komponente	Auswahl
Prozessoren	1
Hauptspeicher	512 MB
Grafikspeicher	12 MB
Festplatte	15 GB
IDE-Controller	Intel PII4
Netzwerk-Adapter	Intel PRO/100 MT Desktop

Tabelle 66: Hardware-Komponenten der Laufzeitumgebung

Neben der Standardinstallation von Ubuntu 8.04 wurden noch folgende Pakete installiert, welche zur Ausführung der Serverplattform bzw. Pandas und der Pandas-Erweiterung benötig wird:

- Java Runtime Environment
- Fortran g77 compiler
- OpenDBX Bibliothek

Die Serverplattform zur Ausführung des WSI_Pandas Adapters und der Workflows bildet sich aus dem Apache Tomcat Server mit Apache Axis und Apache ODE. Für die Datenbankanbindung von Pandas wurde Postgresql 8.3 installiert. Zusätzlich wurde für die Datenbankadministration pgAdmin installiert.

Die Umgebung kann man entweder direkt über die Virtualisierungssoftware oder per SSH-Verbindung verwenden. Da im Rahmen der Diplomarbeit diese Umgebung nicht für den produktiven Einsatz gedacht ist, spielen Sicherheitsaspekte nur eine untergeordnete Rolle. Dies äußert sich z.B. dadurch dass sämtliche Benutzernamen und Passwörter „pandas“ lauten. Damit nach dem Start der virtuellen Maschine das Web Service Interface und die darauf aufbauenden Erweiterungsadapter verwendet werden können, müssen zunächst der Tomcat Server und die Postgresql Datenbank gestartet werden.

Das Starten des Tomcat Servers erfolgt über den folgenden Aufruf des up.sh Skriptes:

```
/srv/tomcat/bin/up.sh
```

Dieses Skript setzt, bevor es den Tomcat Server startet, die LD_LIBRARY_PATH Variable auf den OpenDBX-Bibliothekspfad, damit dieser später der Pandasausführung zur Verfügung steht.

Die Datenbank startet man hingegen mit folgendem Befehl:

```
/etc/init.d/postgresql-8.3 start
```

Nachdem der Tomcat Server und die Datenbank gestartet wurden, können sämtliche Web Services und BPEL-Prozesse über den Port 8080 der virtuellen Maschine erreicht werden. Für das Web Service Interface wurden folgende Verzeichnisse verwendet:

Pfad	**Verwendung**
`/srv/wsi/instances`	Verzeichnis der Simulationsinstanzen
`/srv/wsi/archives/problems`	Archiv-Verzeichnis für die Simulationsdaten von Pandas
`/srv/wsi/archives/src`	Archiv-Verzeichnis für den Pandas Sourcecode

8 Zusammenfassung und Ausblick

Die Nutzung und Bedeutung von Workflows im wissenschaftlichen Umfeld nimmt stetig zu. Dies zeigt sich auch an der Forschungsaktivität in Richtung Scientific Workflows. Daher sollte im Rahmen dieser Arbeit für das FEM-Simulationsprogramm Pandas eine Servicebus-Erweiterung erstellt werden. Hierfür wurde Pandas dahingehend erweitert, um Web Service Anfragen verarbeiten zu können. Damit Pandas als Service verfügbar ist, wurde ein stellvertretender Web Service Interface-Adapter erstellt. Durch die Erstellung eines Datenbank-Schemas und der Verwendung einer Datenbank-Bibliothek in Pandas, ist es nun möglich eine Pandas-Simulation in der Datenbank abzubilden. Hierbei können Pandas Simulationsdaten in der Datenbank abgelegt werden, die von anderen Pandas-Instanzen oder teilweise auch von anderen Programmen verwendet werden können.

Um eine Multi-* Simulation mit Matlab zusammen führen zu können, wurde zusätzlich noch ein stellvertretender Web Service Interface-Adapter für Matlab erstellt. Der Einfachheit halber, werden bei diesem Adapter zur Steuerung von Matlab ausschließlich SSH- und SCP-Befehle abgesetzt.

Im Laufe der Arbeit wurde über das Erfassen der Matrixqualität von Pandas, eine numerische Inkompatibilität zwischen den Matrizen von Pandas und DUNE festgestellt. Aus diesem Grunde schlug die Kopplung von Pandas mit DUNE fehl.

Für die Pandas-Service-Bus-Erweiterung wurden verschiedene Workflows erstellt, um einerseits die allgemeine Workflowfähigkeit von Pandas zu zeigen und andererseits die Kopplung mit anderen Instanzen, sei es Pandas selbst oder ein anderes Simulationsprogramm, zu zeigen. Der erste Workflow erfasst von Pandas in regelmäßigen Abständen die Matrixqualität, welche von dem Daten-Qualitäts-Framework visualisiert werden konnte. Darüber hinaus ließen sich diese Daten zur Steuerung der Simulation heranziehen. Bei dem zweiten Workflow führen zwei Pandas-Instanzen eine Multi-Skalen und eine Multi-Physien Simulation durch. Da im Laufe der Arbeit die unterschiedlichen Physiken noch nicht lauffähig waren, wurde hier zweimal dieselbe Physik verwendet. Für das nächste Szenario wurden drei Workflows erstellt, mit denen es möglich ist mit Pandas und Matlab zusammen eine Multi-Domänen, Multi-Skalen, Multi-Physiken und Multi-Tools Simulation auszuführen. Hierbei ergab sich die Einschränkung die Simulation nur mit einem Matlab-Knoten durchzuführen, da es nicht ohne weiteres möglich war innerhalb des Datenverwaltungs-Workflows Variablen mit Arrays zu erstellen.

Für den erstellten Matlab-Adapter bzw. auch den Pandas-Adapter könnten für den Datenzugriff das Framework SIMPLE oder WS-Ressource verwendet werden. Des Weiteren könnte der Datenverwaltungs-Workflow so erweitert werden, dass er mehrere Matlab-Knoten verwalten kann.

Literaturverzeichnis

[1] Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, Michael Reiter Katharina Görlach, "Conventional Workflow Technology for Scientific Simulation," 2011.

[2] Stephan Hartmann, "The World as a Process: Simulations in the Natural and Social Sciences," in *Simulation and Modelling in the Social Sciences from the Philosophy of Science Point of View*. Dordrecht: Kluwer, 1996, pp. 77-100.

[3] 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*. Upper Saddle River, New Jersey, USA: Prentice Hall PTR, March 22, 2005.

[4] David Booth et al. (2004, February) Web Services Architecture. [Online]. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[5] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. (2001, March) Web Services Description Language (WSDL) 1.1. [Online]. http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[6] Martin Gudgin et al. (2007, April) SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). [Online]. http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[7] Martin Gudgin, Marc Hadley, and Tony Rogers. (2006, May) Web Services Addressing 1.0 - Core. [Online]. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/

[8] Martin Gudgin, Marc Hadley, and Tony Rogers. (2006, May) Web Services Addressing 1.0 - SOAP Binding. [Online]. http://www.w3.org/TR/2006/REC-ws-addr-soap-20060509/

[9] Martin Gudgin, Marc Hadley, Tony Rogers, and Ümit Yalçinalp. (2007, September) Web Services Addressing 1.0 - Metadata. [Online]. http://www.w3.org/TR/2007/REC-ws-addr-metadata-20070904/

[10] Frank Leymann and Dieter Roller, *Production Workflow - Concepts and Techniques*. Upper Saddle River, New Jersey, USA: Prentice Hall PTR, 2000.

[11] David Hollingsworth. (1995, Januar) Workflow Management Coalition: The Workflow Reference Model. [Online]. http://www.wfmc.org/standards/docs/tc003v11.pdf

[12] Frank Leymann et al. (2003, May) Business Process Execution Language for Web Services version 1.1. [Online]. http://www.ibm.com/developerworks/library/specification/ws-bpel/

[13] Anders Berglund et al. (2009, April) XML Path Language (XPath) 2.0 (Second Edition). [Online]. http://www.w3.org/TR/2009/PER-xpath20-20090421/

[14] Nicholas W. Jankowski, "Exploring e-Science: An Introduction," *Journal of Computer-Mediated Communication, 12(2)*, pp. 549-562, 2007.

[15] Tony Hey, Stewart Tansley, and Krist in Tolle, *The Fourth Paradigm*. Redmond, 2009.

[16] Herbert Stachowiak, *Allgemeine Modelltheorie*. Wien: Springer-Verlag, 1973.

[17] Semendjajew Bronstein, *Taschenbuch der Mathematik*.: Verlag Harri Deutsch, 1972.

[18] O. C. Zienkiewicz, *Methode der finiten Elemente*.: Carl Hanser Verlag, 1975.

[19] Hans Rudolf Schwarz, *Methode der finiten Elemente*. Stuttgart: Teubner, 1991.

[20] W. Ehlers, "Grundlegende Konzepte in der Theorie Poröser Medien," in *Technische Mechanik, Band 16*., 1996, pp. 63-76.

[21] Michael Müller and Jochen Ruben, Adaptive dynamisch verteilte Baugrund-Tragwerk-Simulation auf der Grundlage der Theorie poröser Medien, Oktober 2004.

[22] W. Ehlers, "Challenges of Porous Media Models in Geo- and Biomechanical Engineering including Electro-Chemically Active Polymers and Gels," Stuttgart, 2009.

[23] Robert Krause, Wolfgang Ehlers, and Bernd Markert, Bone remodelling: A combined biomechanical and systems-biological model, March 25, 2011.

[24] Jens Rutschmann, Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden, August 17, 2009.

[25] Christoph Marian Müller, Development of an Integrated Database Architecture for a Runtime Environment for Simulation Workflows, February 5, 2010.

[26] Uwe Breitenbücher, Datenqualität in Simulation-Workflows, März 3, 2011.

Anhang

WSDL WSI_Pandas

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="WSI_Pandas"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
      xmlns:tns="http://wsi.simtech.de/extensions/pandas/"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:types="http://wsi.simtech.de/ws/types/"
      targetNamespace="http://wsi.simtech.de/extensions/pandas/">

      <wsdl:types>
            <xsd:schema
targetNamespace="http://wsi.simtech.de/extensions/pandas/">
                  <xsd:element name="ExecuteCommand">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1"
                                                maxOccurs="1" />
                                    <xsd:element name="Command"
type="xsd:string"
                                                minOccurs="1" maxOccurs="1" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="ExecuteCommandSyncFault">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="returnCode"
type="xsd:int"
                                                minOccurs="1" maxOccurs="1" />
                                    <xsd:element name="errorMessage"
type="xsd:string"
                                                minOccurs="1" maxOccurs="1" />
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="readProblem">
                <xsd:annotation>
                  <xsd:documentation>Die Geometrie-Datei enthält die
geometrischen Daten des zu beschreibenden Randwertproblems.
Mit Hilfe dieser Datei kann der Netzgenerator ein FE-Netz
erstellen.</xsd:documentation>
                </xsd:annotation>
                <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long"
                                                minOccurs="1" maxOccurs="1">
                              <xsd:annotation>
                                <xsd:documentation></xsd:documentation>
                              </xsd:annotation>
                                    </xsd:element>
                                    <xsd:element name="ShapeFile"
type="xsd:string"
```

```
                                        minOccurs="1" maxOccurs="1">
                              <xsd:annotation>
                                <xsd:documentation>In der Shape-Datei werden
die mathematischen Voraussetzungen f¨ur eine FE-Berechnung festgelegt.
So werden dort der Typ der finiten Elemente, die Anzahl der
Prim¨arvariablen und deren
Ansatz- und Testfunktionen, die Geometrietransformation und die
Quadraturordnungen f¨ur die
numerische Integration definiert.</xsd:documentation></xsd:annotation>
                                    </xsd:element>
                                    <xsd:element name="GeomFile"
                                        type="xsd:string" maxOccurs="1"
minOccurs="1">
                              <xsd:annotation>
                                <xsd:documentation>Die Geometrie-Datei enthält
die geometrischen Daten des zu beschreibenden Randwertproblems.
Mit Hilfe dieser Datei kann der Netzgenerator ein FE-Netz
erstellen.</xsd:documentation>
                              </xsd:annotation>
                                    </xsd:element>
                                    <xsd:choice minOccurs="1" maxOccurs="1">
                                        <xsd:sequence minOccurs="1"
maxOccurs="1">
                                            <xsd:element
name="createMesh"
                                                type="xsd:boolean"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="BaseName"
                                                type="xsd:string"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="minAngle"
                                                type="xsd:float"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="maxArea"
                                                type="xsd:long"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="o2"
                                                type="xsd:boolean"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                        </xsd:sequence>
                                        <xsd:sequence minOccurs="1"
maxOccurs="1">
                                            <xsd:element
name="createGrid"
                                                type="xsd:boolean"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element
name="ShapeType"
                                                type="xsd:string"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="PhysType"
                                                type="xsd:string"
minOccurs="1" maxOccurs="1">
                                            </xsd:element>
                                            <xsd:element name="nel_x"
type="xsd:int"
```

```
                                                    minOccurs="1"
maxOccurs="1">
                                                </xsd:element>
                                                <xsd:element name="nel_y"
type="xsd:int"
                                                    minOccurs="1"
maxOccurs="1">
                                                </xsd:element>
                                            </xsd:sequence>
                                            <xsd:sequence>
                                                <xsd:element
name="useNodesElems"
                                                    type="xsd:boolean"
minOccurs="1" maxOccurs="1">
                                                </xsd:element>
                                                <xsd:element
name="NodesFile"
                                                    type="xsd:string"
maxOccurs="1" minOccurs="1">
                                        <xsd:annotation>
                                            <xsd:documentation>In der Knoten-
Datei werden die Koordinaten aller Netz-Knoten als Fließkommazahlen
festgelegt. Die Indizes der Knoten sind über die jeweilige Zeilennummer
definiert, wobei die vier Kopfzeilen nicht mitzählen. Die Indizierung
beginnt mit 1.</xsd:documentation>
                                        </xsd:annotation>
                                                </xsd:element>
                                                <xsd:element
name="ElemsFile"
                                                    type="xsd:string"
maxOccurs="1" minOccurs="1">
                                        <xsd:annotation>
                                            <xsd:documentation>In der Element-
Datei wird festgelegt, welche Knoten ein Element bilden. Dabei wird pro
Zeile durch Angabe der Knotenindizes aus der Knoten-Datei genau ein Element
definiert. Die Anzahl und Reihenfolge der Elementknoten wird durch den
Elementtyp bestimmt.</xsd:documentation>
                                        </xsd:annotation>
                                                </xsd:element>
                                    <xsd:element name="BoundaryFile"
type="xsd:string" minOccurs="0" maxOccurs="1">
                                        <xsd:annotation>
                                            <xsd:documentation>In der
Randbedingungs-Datei können die Randbedingungen, die in der Boundary-
Condition-Datei definiert wurden, auf ihre Richtigkeit überprüft werden.
Sie wird durch den Befehl write von PANDAS erzeugt. Die Randbedingungen pro
Knoten werden in der Reihenfolge der Knotenfreiheitsgrade angegeben. Diese
Reihenfolge und auch die Anzahl der Knotenfreiheitsgrade ist von der
verwendeten Physik abhängig.</xsd:documentation>
                                        </xsd:annotation>
                                    </xsd:element>
                                </xsd:sequence>

                                    </xsd:choice>

                            <xsd:element name="DescriptionFile"
type="xsd:string" minOccurs="0" maxOccurs="1">
                                <xsd:annotation>
                                  <xsd:documentation>Durch das Lesen der
Problembeschreibungsdatei kann ein ganzes Problem beschrieben
werden.</xsd:documentation></xsd:annotation>
                            </xsd:element>
                            <xsd:element name="IvarsFile"
```

```
                                        type="xsd:string" maxOccurs="1"
minOccurs="1">
                              <xsd:annotation>
                                <xsd:documentation>Die Datei der internen
Variablen enthält die Namen (Strings) der internen Variablen, die später
visualisiert oder in eine Datei ausgegeben werden sollen. Es wird dabei
nicht zwischen Groß- und Kleinschreibung der Variablennamen
unterschieden.</xsd:documentation>
                              </xsd:annotation>
                           </xsd:element>
                                 <xsd:element name="ParamFile"
                                        type="xsd:string" maxOccurs="1"
minOccurs="1">
                              <xsd:annotation>
                                <xsd:documentation>In der Materialparameter-
Datei werden die Materialparameter für die jeweils verwendete Elementphysik
(z. B. linear elastisches Materialverhalten) angegeben. Die Anzahl, Art und
Reihenfolge der Materialparameter ist durch die Wahl der Elementphysik
gegeben. Für eine Elementphysik können auch mehrere Materialparametersätze
(matsets) angegeben werden. Damit ist es möglich, ein Problem mit gleicher
Elementphysik aber verschiedenen Materialparametern in unterschiedlichen
Gebieten zu berechnen.</xsd:documentation>
                              </xsd:annotation>
                                 </xsd:element>
                              </xsd:sequence>
                           </xsd:complexType>
                     </xsd:element>
                     <xsd:element name="readProblemResponse">
                           <xsd:complexType>
                              <xsd:sequence>
                                 <xsd:element name="ReturnCode"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                                 <xsd:element name="errorMessage"
                                        type="xsd:string" minOccurs="0"
maxOccurs="1">
                                 </xsd:element>
                              </xsd:sequence>
                           </xsd:complexType>
                     </xsd:element>

                     <xsd:element name="connect-db">
                           <xsd:complexType>
                              <xsd:sequence>
                                 <xsd:element name="SimID"
                                        type="xsd:long" minOccurs="1"
maxOccurs="1">
                                 </xsd:element>
                                 <xsd:element name="backend"
type="xsd:string"
                                        minOccurs="1" maxOccurs="1">
                                 </xsd:element>
                                 <xsd:element name="host"
type="xsd:string"
                                        minOccurs="1" maxOccurs="1">
                                 </xsd:element>
                                 <xsd:element name="port"
type="xsd:string"
                                        minOccurs="1" maxOccurs="1">
                                 </xsd:element>
                                 <xsd:element name="db" type="xsd:string"
                                        minOccurs="1" maxOccurs="1">
                                 </xsd:element>
```

```
                                    <xsd:element name="user"
type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="pw" type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="encryption"
                                          type="xsd:boolean" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="compression"
                                          type="xsd:boolean" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="connect-dbResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="returnCode"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="error-msg"
                                          type="xsd:string" minOccurs="0"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="disconnect-db">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
                                          type="xsd:long">
                                    </xsd:element>
                                    <xsd:element name="host"
type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="port"
type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="db" type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="disconnect-dbResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="returnCode"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="error-msg"
                                          type="xsd:string" minOccurs="0"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="set-option">
```

```
306                         <xsd:complexType>
307                               <xsd:sequence>
308                                     <xsd:element name="SimID"
309                                           type="xsd:long" minOccurs="1"
310   maxOccurs="1">
311                                     </xsd:element>
312                                     <xsd:element name="save-matrix"
313                                           type="xsd:boolean" minOccurs="1"
314   maxOccurs="1">
315                                     </xsd:element>
316                                     <xsd:element name="load-matrix"
317                                           type="xsd:boolean" minOccurs="1"
318   maxOccurs="1">
319                                     </xsd:element>
320                                     <xsd:element name="save-binary"
321                                           type="xsd:boolean" minOccurs="1"
322   maxOccurs="1">
323                                     </xsd:element>
324                                     <xsd:element name="create-tables"
325                                           type="xsd:boolean" minOccurs="1"
326   maxOccurs="1">
327                                     </xsd:element>
328                                     <xsd:element name="step-width"
329   type="xsd:int"
330                                           minOccurs="1" maxOccurs="1">
331                                     </xsd:element>
332                               </xsd:sequence>
333                         </xsd:complexType>
334                   </xsd:element>
335                   <xsd:element name="set-optionResponse">
336                         <xsd:complexType>
337                               <xsd:sequence>
338                                     <xsd:element name="returnCode"
339   type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
340                                     <xsd:element name="error-msg"
341                                           type="xsd:string" minOccurs="1"
342   maxOccurs="1">
343                                     </xsd:element>
344                               </xsd:sequence>
345                         </xsd:complexType>
346                   </xsd:element>
347                   <xsd:element name="do-step">
348                         <xsd:complexType>
349                               <xsd:sequence>
350                                     <xsd:element name="SimID"
351   type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
352                               </xsd:sequence>
353                         </xsd:complexType>
354                   </xsd:element>
355                   <xsd:element name="do-stepResponse">
356                         <xsd:complexType>
357                               <xsd:sequence>
358                                     <xsd:element name="stepnr"
359                                           type="xsd:int" minOccurs="1"
360   maxOccurs="1">
361                                     </xsd:element>
362                               </xsd:sequence>
363                         </xsd:complexType>
364                   </xsd:element>
365                   <xsd:element name="save-dof">
366                         <xsd:complexType>
367                               <xsd:sequence>
```

```
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="save-dofResponse">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="load-dof">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="instance_SimID"
type="xsd:long"
                                        minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="load_SimID"
                                        type="xsd:long" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Stepnr"
type="xsd:int"
                                        minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="run-cmd">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="sid" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="cmd-filename"
                                        type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="run-cmdResponse">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="do-step-pseudo">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="sid" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                        <xsd:element name="do-step-pseudoResponse">
                            <xsd:complexType>
                                <xsd:sequence>
```

```
                                    <xsd:element name="sql-stmt"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getDataQualityQuery">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getDataQualityQueryResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="sql-stmt"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getDataQualityQueryPseudo">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getDataQualityQueryPseudoResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="sql-stmt"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getStepnr">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getStepnrResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getStepnrPseudo">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getStepnrPseudoResponse">
                        <xsd:complexType>
```

```
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getLastSavedStepnr">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getLastSavedStepnrResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getLastSavedStepnrPseudo">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getLastSavedStepnrPseudoResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getMid">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getMidResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Mid" type="xsd:int"
minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getMidPseudo">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getMidPseudoResponse">
```

```
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Mid" type="xsd:int"
minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveDataQuality">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveDataQualityResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveMatrix">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="saveBinary"
                                          type="xsd:boolean" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveMatrixResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadMatrix">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="load_SimID"
                                          type="xsd:long" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Stepnr"
type="xsd:int"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="loadBinary"
                                          type="xsd:boolean" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
```

```
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadMatrixResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="prepareSimulation">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="ProblemName"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="prepareSimulationResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="SimID"
                                          type="xsd:long" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="startPandas">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="ProgramPath"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="WorkingDirectory"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Arguments"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="ExecutionLog"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="startPandasResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getUsedParamFile">
```

```
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="ProblemName"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="getUsedParamFileResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Path"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveState">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveStateResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadState">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="load_SimID"
type="xsd:long"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Stepnr"
type="xsd:int"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="load_stateResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveMeshTrans">
                        <xsd:complexType>
```

```
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveMeshTransResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="ElementCount"
                                          type="xsd:int" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadMeshTrans">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadMeshTransResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="ReadNextValFromMeshFile">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="ReadNextValFromMeshFileResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="WriteValToMeshFile">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="in"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="WriteValToMeshFileResponse">
                        <xsd:complexType>
                              <xsd:sequence>
```

```
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveAllGauss">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveAllGaussResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadAllGauss">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="loadSimID"
                                          type="xsd:long" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Stepnr"
                                          type="xsd:int" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadGaussResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveGaussName">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long" minOccurs="1" maxOccurs="1"></xsd:element>
                                    <xsd:element name="Name"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="saveGaussNameResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Stepnr"
type="xsd:int" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
```

```
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadGaussName">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID"
type="xsd:long"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="loadSimID"
type="xsd:long"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Stepnr"
type="xsd:int"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Name"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
                  <xsd:element name="loadGaussNameResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="out"
type="xsd:string"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:schema>
            <xsd:schema>
                  <xsd:import namespace="http://wsi.simtech.de/ws/types/"
                        schemaLocation="types.xsd" />
            </xsd:schema>
      </wsdl:types>

      <wsdl:message name="ExecuteCommandSyncRequest">
            <wsdl:part element="tns:ExecuteCommand" name="parameters" />
      </wsdl:message>
      <wsdl:message name="EmptyMessage">
      </wsdl:message>
      <wsdl:message name="ExecuteCommandException">
            <wsdl:part name="fault"
element="tns:ExecuteCommandSyncFault"></wsdl:part>
      </wsdl:message>
      <wsdl:message name="InvalidStateException">
            <wsdl:part name="fault"
element="types:InvalidStateFault"></wsdl:part>
      </wsdl:message>
      <wsdl:message name="ExecuteCommandAsyncRequest">
            <wsdl:part name="parameters"
element="tns:ExecuteCommand"></wsdl:part>
      </wsdl:message>
      <wsdl:message name="SimIDMessage">
            <wsdl:part name="parameters" element="types:SimID"></wsdl:part>
      </wsdl:message>
      <wsdl:message name="readProblemRequest">
            <wsdl:part name="parameters"
element="tns:readProblem"></wsdl:part>
      </wsdl:message>
```

```
934         <wsdl:message name="readProblemResponse">
935         </wsdl:message>
936         <wsdl:message name="connect-dbRequest">
937               <wsdl:part name="parameters" element="tns:connect-
938   db"></wsdl:part>
939         </wsdl:message>
940         <wsdl:message name="connect-dbResponse">
941         </wsdl:message>
942         <wsdl:message name="disconnect-dbRequest">
943               <wsdl:part name="parameters" element="tns:disconnect-
944   db"></wsdl:part>
945         </wsdl:message>
946         <wsdl:message name="disconnect-dbResponse">
947         </wsdl:message>
948         <wsdl:message name="set-optionRequest">
949               <wsdl:part name="parameters" element="tns:set-
950   option"></wsdl:part>
951         </wsdl:message>
952         <wsdl:message name="set-optionResponse">
953         </wsdl:message>
954         <wsdl:message name="do-stepRequest">
955               <wsdl:part name="parameters" element="tns:do-step"></wsdl:part>
956         </wsdl:message>
957         <wsdl:message name="do-stepResponse">
958               <wsdl:part name="parameters" element="tns:do-
959   stepResponse"></wsdl:part>
960         </wsdl:message>
961         <wsdl:message name="save-dofRequest">
962               <wsdl:part name="parameters" element="tns:save-
963   dof"></wsdl:part>
964         </wsdl:message>
965         <wsdl:message name="save-dofResponse">
966               <wsdl:part name="parameters" element="tns:save-
967   dofResponse"></wsdl:part>
968         </wsdl:message>
969         <wsdl:message name="load-dofRequest">
970               <wsdl:part name="parameters" element="tns:load-
971   dof"></wsdl:part>
972         </wsdl:message>
973         <wsdl:message name="load-dofResponse">
974         </wsdl:message>
975         <wsdl:message name="run-cmdRequest">
976               <wsdl:part name="parameters" element="tns:run-cmd"></wsdl:part>
977         </wsdl:message>
978         <wsdl:message name="run-cmdResponse">
979         </wsdl:message>
980         <wsdl:message name="do-step-pseudoRequest">
981               <wsdl:part name="parameters" element="tns:do-step-
982   pseudo"></wsdl:part>
983         </wsdl:message>
984         <wsdl:message name="do-step-pseudoResponse">
985               <wsdl:part name="parameters" element="tns:do-step-
986   pseudoResponse"></wsdl:part>
987         </wsdl:message>
988         <wsdl:message name="getDataQualityQueryRequest">
989               <wsdl:part name="parameters"
990   element="tns:getDataQualityQuery"></wsdl:part>
991         </wsdl:message>
992         <wsdl:message name="getDataQualityQueryResponse">
993               <wsdl:part name="parameters"
994   element="tns:getDataQualityQueryResponse"></wsdl:part>
995         </wsdl:message>
996         <wsdl:message name="getDataQualityQueryPseudoRequest">
```

```
997               <wsdl:part name="parameters"
998   element="tns:getDataQualityQueryPseudo"></wsdl:part>
999         </wsdl:message>
1000        <wsdl:message name="getDataQualityQueryPseudoResponse">
1001              <wsdl:part name="parameters"
1002  element="tns:getDataQualityQueryPseudoResponse"></wsdl:part>
1003        </wsdl:message>
1004        <wsdl:message name="getStepnrRequest">
1005              <wsdl:part name="parameters"
1006  element="tns:getStepnr"></wsdl:part>
1007        </wsdl:message>
1008        <wsdl:message name="getStepnrResponse">
1009              <wsdl:part name="parameters"
1010  element="tns:getStepnrResponse"></wsdl:part>
1011        </wsdl:message>
1012        <wsdl:message name="getStepnrPseudoRequest">
1013              <wsdl:part name="parameters"
1014  element="tns:getStepnrPseudo"></wsdl:part>
1015        </wsdl:message>
1016        <wsdl:message name="getStepnrPseudoResponse">
1017              <wsdl:part name="parameters"
1018  element="tns:getStepnrPseudoResponse"></wsdl:part>
1019        </wsdl:message>
1020        <wsdl:message name="getLastSavedStepnrRequest">
1021              <wsdl:part name="parameters"
1022  element="tns:getLastSavedStepnr"></wsdl:part>
1023        </wsdl:message>
1024        <wsdl:message name="getLastSavedStepnrResponse">
1025              <wsdl:part name="parameters"
1026  element="tns:getLastSavedStepnrResponse"></wsdl:part>
1027        </wsdl:message>
1028        <wsdl:message name="getLastSavedStepnrPseudoRequest">
1029              <wsdl:part name="parameters"
1030  element="tns:getLastSavedStepnrPseudo"></wsdl:part>
1031        </wsdl:message>
1032        <wsdl:message name="getLastSavedStepnrPseudoResponse">
1033              <wsdl:part name="parameters"
1034  element="tns:getLastSavedStepnrPseudoResponse"></wsdl:part>
1035        </wsdl:message>
1036        <wsdl:message name="getMidRequest">
1037              <wsdl:part name="parameters" element="tns:getMid"></wsdl:part>
1038        </wsdl:message>
1039        <wsdl:message name="getMidResponse">
1040              <wsdl:part name="parameters"
1041  element="tns:getMidResponse"></wsdl:part>
1042        </wsdl:message>
1043        <wsdl:message name="getMidPseudoRequest">
1044              <wsdl:part name="parameters"
1045  element="tns:getMidPseudo"></wsdl:part>
1046        </wsdl:message>
1047        <wsdl:message name="getMidPseudoResponse">
1048              <wsdl:part name="parameters"
1049  element="tns:getMidPseudoResponse"></wsdl:part>
1050        </wsdl:message>
1051        <wsdl:message name="saveDataQualityRequest">
1052              <wsdl:part name="parameters"
1053  element="tns:saveDataQuality"></wsdl:part>
1054        </wsdl:message>
1055        <wsdl:message name="saveDataQualityResponse">
1056        </wsdl:message>
1057        <wsdl:message name="saveMatrixRequest">
1058              <wsdl:part name="parameters"
1059  element="tns:saveMatrix"></wsdl:part>
```

```
        </wsdl:message>
        <wsdl:message name="saveMatrixResponse">
              <wsdl:part name="parameters"
element="tns:saveMatrixResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadMatrixRequest">
              <wsdl:part name="parameters"
element="tns:loadMatrix"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadMatrixResponse">
        </wsdl:message>
        <wsdl:message name="prepareSimulationRequest">
              <wsdl:part name="parameters"
element="tns:prepareSimulation"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="prepareSimulationResponse">
              <wsdl:part name="parameters"
element="tns:prepareSimulationResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="startPandasRequest">
              <wsdl:part name="parameters"
element="tns:startPandas"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="startPandasResponse">
              <wsdl:part name="parameters"
element="tns:startPandasResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="getUsedParamFileRequest">
              <wsdl:part name="parameters"
element="tns:getUsedParamFile"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="getUsedParamFileResponse">
              <wsdl:part name="parameters"
element="tns:getUsedParamFileResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="saveStateRequest">
              <wsdl:part name="parameters"
element="tns:saveState"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="saveStateResponse">
              <wsdl:part name="parameters"
element="tns:saveStateResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadStateRequest">
              <wsdl:part name="parameters"
element="tns:loadState"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadStateResponse">
        </wsdl:message>
        <wsdl:message name="saveMeshTransRequest">
              <wsdl:part name="parameters"
element="tns:saveMeshTrans"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="saveMeshTransResponse">
              <wsdl:part name="parameters"
element="tns:saveMeshTransResponse"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadMeshTransRequest">
              <wsdl:part name="parameters"
element="tns:loadMeshTrans"></wsdl:part>
        </wsdl:message>
        <wsdl:message name="loadMeshTransResponse">
        </wsdl:message>
```

```
1123        <wsdl:message name="ReadNextValFromMeshFileRequest">
1124              <wsdl:part name="parameters"
1125  element="tns:ReadNextValFromMeshFile"></wsdl:part>
1126        </wsdl:message>
1127        <wsdl:message name="ReadNextValFromMeshFileResponse">
1128              <wsdl:part name="parameters"
1129  element="tns:ReadNextValFromMeshFileResponse"></wsdl:part>
1130        </wsdl:message>
1131        <wsdl:message name="WriteValToMeshFileRequest">
1132              <wsdl:part name="parameters"
1133  element="tns:WriteValToMeshFile"></wsdl:part>
1134        </wsdl:message>
1135        <wsdl:message name="WriteValToMeshFileResponse">
1136              <wsdl:part name="parameters"
1137  element="tns:WriteValToMeshFileResponse"></wsdl:part>
1138        </wsdl:message>
1139
1140        <wsdl:message name="saveAllGaussRequest">
1141              <wsdl:part name="parameters"
1142  element="tns:saveAllGauss"></wsdl:part>
1143        </wsdl:message>
1144        <wsdl:message name="saveAllGaussResponse">
1145              <wsdl:part name="parameters"
1146  element="tns:saveAllGaussResponse"></wsdl:part>
1147        </wsdl:message>
1148        <wsdl:message name="loadAllGaussRequest">
1149              <wsdl:part name="parameters"
1150  element="tns:loadAllGauss"></wsdl:part>
1151        </wsdl:message>
1152        <wsdl:message name="loadAllGaussResponse">
1153        </wsdl:message>
1154        <wsdl:message name="saveGaussNameRequest">
1155              <wsdl:part name="parameters"
1156  element="tns:saveGaussName"></wsdl:part>
1157        </wsdl:message>
1158        <wsdl:message name="saveGaussNameResponse">
1159              <wsdl:part name="parameters"
1160  element="tns:saveGaussNameResponse"></wsdl:part>
1161        </wsdl:message>
1162        <wsdl:message name="loadGaussNameRequest">
1163              <wsdl:part name="parameters"
1164  element="tns:loadGaussName"></wsdl:part>
1165        </wsdl:message>
1166        <wsdl:message name="loadGaussNameResponse">
1167        </wsdl:message>
1168        <wsdl:portType name="WSI_Pandas">
1169              <wsdl:operation name="executeCommandSync">
1170                    <wsdl:documentation>Executes a Pandas command
1171  synchronously.
1172                    </wsdl:documentation>
1173                    <wsdl:input message="tns:ExecuteCommandSyncRequest" />
1174                    <wsdl:output message="tns:EmptyMessage" />
1175                    <wsdl:fault name="CommandFault"
1176  message="tns:ExecuteCommandException" />
1177                    <wsdl:fault name="InvalidStateFault"
1178  message="tns:InvalidStateException" />
1179              </wsdl:operation>
1180              <wsdl:operation name="stopApplication">
1181                    <wsdl:documentation>Shuts down the simulation
1182  application.
1183                    </wsdl:documentation>
1184                    <wsdl:input message="tns:SimIDMessage"></wsdl:input>
1185                    <wsdl:output message="tns:EmptyMessage"></wsdl:output>
```

```
                    <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException" />
                    <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException" />
                </wsdl:operation>
                <wsdl:operation name="readProblem">
                    <wsdl:documentation>Read a problem to initialize
Pandas</wsdl:documentation>
                    <wsdl:input
message="tns:readProblemRequest"></wsdl:input>
                    <wsdl:output
message="tns:readProblemResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="connect-db">
                    <wsdl:documentation>Connect Pandas with a
DB</wsdl:documentation>
                    <wsdl:input message="tns:connect-dbRequest"></wsdl:input>
                    <wsdl:output message="tns:connect-
dbResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="disconnect-db">
                    <wsdl:documentation>Disconnects Pandas from a
DB</wsdl:documentation>
                    <wsdl:input message="tns:disconnect-
dbRequest"></wsdl:input>
                    <wsdl:output message="tns:disconnect-
dbResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="set-option">
                    <wsdl:input message="tns:set-optionRequest"></wsdl:input>
                    <wsdl:output message="tns:set-
optionResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="do-step">
                    <wsdl:documentation>execute one simulation
step</wsdl:documentation>
                    <wsdl:input message="tns:do-stepRequest"></wsdl:input>
                    <wsdl:output message="tns:do-stepResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="save-dof">
                    <wsdl:documentation>saves algebraic mesh data at the
actual simulation step</wsdl:documentation>
                    <wsdl:input message="tns:save-dofRequest"></wsdl:input>
```

```
                    <wsdl:output message="tns:save-
dofResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="load-dof">
                    <wsdl:documentation>loades algebraic mesh data from a
certain simulation step</wsdl:documentation>
                    <wsdl:input message="tns:load-dofRequest"></wsdl:input>
                    <wsdl:output message="tns:load-
dofResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="run-cmd">
                    <wsdl:documentation>executes a command batch
file</wsdl:documentation>
                    <wsdl:input message="tns:run-cmdRequest"></wsdl:input>
                    <wsdl:output message="tns:run-cmdResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="do-step-pseudo">
                    <wsdl:input message="tns:do-step-
pseudoRequest"></wsdl:input>
                    <wsdl:output message="tns:do-step-
pseudoResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="getDataQualityQuery">
                    <wsdl:input
message="tns:getDataQualityQueryRequest"></wsdl:input>
                    <wsdl:output
message="tns:getDataQualityQueryResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="getDataQualityQueryPseudo">
                    <wsdl:input
message="tns:getDataQualityQueryPseudoRequest"></wsdl:input>
                    <wsdl:output
message="tns:getDataQualityQueryPseudoResponse"></wsdl:output>
                <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
                <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            </wsdl:operation>
                <wsdl:operation name="getStepnr">
                    <wsdl:input message="tns:getStepnrRequest"></wsdl:input>
                    <wsdl:output
message="tns:getStepnrResponse"></wsdl:output>
```

```
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="getStepnrPseudo">
                <wsdl:input
message="tns:getStepnrPseudoRequest"></wsdl:input>
                <wsdl:output
message="tns:getStepnrPseudoResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="getLastSavedStepnr">
                <wsdl:input
message="tns:getLastSavedStepnrRequest"></wsdl:input>
                <wsdl:output
message="tns:getLastSavedStepnrResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="getLastSavedStepnrPseudo">
                <wsdl:input
message="tns:getLastSavedStepnrPseudoRequest"></wsdl:input>
                <wsdl:output
message="tns:getLastSavedStepnrPseudoResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="getMid">
                <wsdl:input message="tns:getMidRequest"></wsdl:input>
                <wsdl:output message="tns:getMidResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="getMidPseudo">
                <wsdl:input
message="tns:getMidPseudoRequest"></wsdl:input>
                <wsdl:output
message="tns:getMidPseudoResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="saveDataQuality">
                <wsdl:input
message="tns:saveDataQualityRequest"></wsdl:input>
                <wsdl:output
message="tns:saveDataQualityResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
```

```
1374            <wsdl:operation name="saveMatrix">
1375                <wsdl:input message="tns:saveMatrixRequest"></wsdl:input>
1376                <wsdl:output
1377  message="tns:saveMatrixResponse"></wsdl:output>
1378            <wsdl:fault name="CommandFault"
1379  message="tns:ExecuteCommandException"></wsdl:fault>
1380            <wsdl:fault name="InvalidStateFault"
1381  message="tns:InvalidStateException"></wsdl:fault>
1382        </wsdl:operation>
1383            <wsdl:operation name="loadMatrix">
1384                <wsdl:input message="tns:loadMatrixRequest"></wsdl:input>
1385                <wsdl:output
1386  message="tns:loadMatrixResponse"></wsdl:output>
1387            <wsdl:fault name="CommandFault"
1388  message="tns:ExecuteCommandException"></wsdl:fault>
1389            <wsdl:fault name="InvalidStateFault"
1390  message="tns:InvalidStateException"></wsdl:fault>
1391        </wsdl:operation>
1392            <wsdl:operation name="prepareSimulation">
1393                <wsdl:input
1394  message="tns:prepareSimulationRequest"></wsdl:input>
1395                <wsdl:output
1396  message="tns:prepareSimulationResponse"></wsdl:output>
1397            <wsdl:fault name="CommandFault"
1398  message="tns:ExecuteCommandException"></wsdl:fault>
1399            <wsdl:fault name="InvalidStateFault"
1400  message="tns:InvalidStateException"></wsdl:fault>
1401        </wsdl:operation>
1402            <wsdl:operation name="startPandas">
1403                <wsdl:input
1404  message="tns:startPandasRequest"></wsdl:input>
1405                <wsdl:output
1406  message="tns:startPandasResponse"></wsdl:output>
1407            <wsdl:fault name="CommandFault"
1408  message="tns:ExecuteCommandException"></wsdl:fault>
1409            <wsdl:fault name="InvalidStateFault"
1410  message="tns:InvalidStateException"></wsdl:fault>
1411        </wsdl:operation>
1412            <wsdl:operation name="getUsedParamFile">
1413                <wsdl:input
1414  message="tns:getUsedParamFileRequest"></wsdl:input>
1415                <wsdl:output
1416  message="tns:getUsedParamFileResponse"></wsdl:output>
1417            <wsdl:fault name="CommandFault"
1418  message="tns:ExecuteCommandException"></wsdl:fault>
1419            <wsdl:fault name="InvalidStateFault"
1420  message="tns:InvalidStateException"></wsdl:fault>
1421        </wsdl:operation>
1422            <wsdl:operation name="saveState">
1423                <wsdl:input message="tns:saveStateRequest"></wsdl:input>
1424                <wsdl:output
1425  message="tns:saveStateResponse"></wsdl:output>
1426            <wsdl:fault name="CommandFault"
1427  message="tns:ExecuteCommandException"></wsdl:fault>
1428            <wsdl:fault name="InvalidStateFault"
1429  message="tns:InvalidStateException"></wsdl:fault>
1430        </wsdl:operation>
1431            <wsdl:operation name="loadState">
1432                <wsdl:input message="tns:loadStateRequest"></wsdl:input>
1433                <wsdl:output
1434  message="tns:loadStateResponse"></wsdl:output>
1435            <wsdl:fault name="CommandFault"
1436  message="tns:ExecuteCommandException"></wsdl:fault>
```

```
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="saveMeshTrans">
                <wsdl:input
message="tns:saveMeshTransRequest"></wsdl:input>
                <wsdl:output
message="tns:saveMeshTransResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="loadMeshTrans">
                <wsdl:input
message="tns:loadMeshTransRequest"></wsdl:input>
                <wsdl:output
message="tns:loadMeshTransResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="ReadNextValFromMeshFile">
                <wsdl:input
message="tns:ReadNextValFromMeshFileRequest"></wsdl:input>
                <wsdl:output
message="tns:ReadNextValFromMeshFileResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="WriteValToMeshFile">
                <wsdl:input
message="tns:WriteValToMeshFileRequest"></wsdl:input>
                <wsdl:output
message="tns:WriteValToMeshFileResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="saveAllGauss">
                <wsdl:input
message="tns:saveAllGaussRequest"></wsdl:input>
                <wsdl:output
message="tns:saveAllGaussResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="loadAllGauss">
                <wsdl:input
message="tns:loadAllGaussRequest"></wsdl:input>
                <wsdl:output
message="tns:loadAllGaussResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
```

```
            <wsdl:operation name="saveGaussName">
                  <wsdl:input
message="tns:saveGaussNameRequest"></wsdl:input>
                  <wsdl:output
message="tns:saveGaussNameResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
            <wsdl:operation name="loadGaussName">
                  <wsdl:input
message="tns:loadGaussNameRequest"></wsdl:input>
                  <wsdl:output
message="tns:loadGaussNameResponse"></wsdl:output>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
            <wsdl:fault name="InvalildStateFault"
message="tns:InvalidStateException"></wsdl:fault>
        </wsdl:operation>
      </wsdl:portType>


      <wsdl:binding name="WSI_PandasSOAP" type="tns:WSI_Pandas">
            <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http" />
            <wsdl:operation name="executeCommandSync">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/executeCommandSyn
c" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="stopApplication">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/stopApplication"
/>
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
```

```
1563              </wsdl:operation>
1564              <wsdl:operation name="readProblem">
1565                    <soap:operation
1566
1567        soapAction="http://wsi.simtech.de/extensions/pandas/readProblem" />
1568                    <wsdl:input>
1569                          <soap:body use="literal" />
1570                    </wsdl:input>
1571                    <wsdl:output>
1572                          <soap:body use="literal" />
1573                    </wsdl:output>
1574                    <wsdl:fault name="CommandFault">
1575                          <soap:fault use="literal" name="CommandFault" />
1576                    </wsdl:fault>
1577                    <wsdl:fault name="InvalidStateFault">
1578                          <soap:fault use="literal" name="InvalidStateFault"
1579    />
1580                    </wsdl:fault>
1581              </wsdl:operation>
1582              <wsdl:operation name="connect-db">
1583                    <soap:operation
1584
1585        soapAction="http://wsi.simtech.de/extensions/pandas/connect-db" />
1586                    <wsdl:input>
1587                          <soap:body use="literal" />
1588                    </wsdl:input>
1589                    <wsdl:output>
1590                          <soap:body use="literal" />
1591                    </wsdl:output>
1592                    <wsdl:fault name="CommandFault">
1593                          <soap:fault use="literal" name="CommandFault" />
1594                    </wsdl:fault>
1595                    <wsdl:fault name="InvalidStateFault">
1596                          <soap:fault use="literal" name="InvalidStateFault"
1597    />
1598                    </wsdl:fault>
1599              </wsdl:operation>
1600              <wsdl:operation name="disconnect-db">
1601                    <soap:operation
1602
1603        soapAction="http://wsi.simtech.de/extensions/pandas/disconnect-db" />
1604                    <wsdl:input>
1605                          <soap:body use="literal" />
1606                    </wsdl:input>
1607                    <wsdl:output>
1608                          <soap:body use="literal" />
1609                    </wsdl:output>
1610                    <wsdl:fault name="CommandFault">
1611                          <soap:fault use="literal" name="CommandFault" />
1612                    </wsdl:fault>
1613                    <wsdl:fault name="InvalidStateFault">
1614                          <soap:fault use="literal" name="InvalidStateFault"
1615    />
1616                    </wsdl:fault>
1617              </wsdl:operation>
1618              <wsdl:operation name="set-option">
1619                    <soap:operation
1620
1621        soapAction="http://wsi.simtech.de/extensions/pandas/set-option" />
1622                    <wsdl:input>
1623                          <soap:body use="literal" />
1624                    </wsdl:input>
1625                    <wsdl:output>
```

```
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="do-step">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/do-step" />
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="save-dof">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/save-dof" />
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="load-dof">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/load-dof" />
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
```

```
1689              </wsdl:operation>
1690              <wsdl:operation name="run-cmd">
1691                    <soap:operation
1692
1693        soapAction="http://wsi.simtech.de/extensions/pandas/run-cmd" />
1694                    <wsdl:input>
1695                          <soap:body use="literal" />
1696                    </wsdl:input>
1697                    <wsdl:output>
1698                          <soap:body use="literal" />
1699                    </wsdl:output>
1700                    <wsdl:fault name="CommandFault">
1701                          <soap:fault use="literal" name="CommandFault" />
1702                    </wsdl:fault>
1703                    <wsdl:fault name="InvalidStateFault">
1704                          <soap:fault use="literal" name="InvalidStateFault"
1705    />
1706                    </wsdl:fault>
1707              </wsdl:operation>
1708              <wsdl:operation name="do-step-pseudo">
1709                    <soap:operation
1710
1711        soapAction="http://wsi.simtech.de/extensions/pandas/do-step-pseudo"
1712    />
1713                    <wsdl:input>
1714                          <soap:body use="literal" />
1715                    </wsdl:input>
1716                    <wsdl:output>
1717                          <soap:body use="literal" />
1718                    </wsdl:output>
1719                    <wsdl:fault name="CommandFault">
1720                          <soap:fault use="literal" name="CommandFault" />
1721                    </wsdl:fault>
1722                    <wsdl:fault name="InvalidStateFault">
1723                          <soap:fault use="literal" name="InvalidStateFault"
1724    />
1725                    </wsdl:fault>
1726              </wsdl:operation>
1727              <wsdl:operation name="getDataQualityQuery">
1728                    <soap:operation
1729
1730        soapAction="http://wsi.simtech.de/extensions/pandas/getDataQualityQue
1731    ry" />
1732                    <wsdl:input>
1733                          <soap:body use="literal" />
1734                    </wsdl:input>
1735                    <wsdl:output>
1736                          <soap:body use="literal" />
1737                    </wsdl:output>
1738                    <wsdl:fault name="CommandFault">
1739                          <soap:fault use="literal" name="CommandFault" />
1740                    </wsdl:fault>
1741                    <wsdl:fault name="InvalidStateFault">
1742                          <soap:fault use="literal" name="InvalidStateFault"
1743    />
1744                    </wsdl:fault>
1745              </wsdl:operation>
1746              <wsdl:operation name="getDataQualityQueryPseudo">
1747                    <soap:operation
1748
1749        soapAction="http://wsi.simtech.de/extensions/pandas/getDataQualityQue
1750    ryPseudo" />
1751                    <wsdl:input>
```

```
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="getStepnr">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/getStepnr" />
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="getStepnrPseudo">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/getStepnrPseudo"
/>
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                    </wsdl:fault>
                    <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                    </wsdl:fault>
                </wsdl:operation>
                <wsdl:operation name="getLastSavedStepnr">
                    <soap:operation

        soapAction="http://wsi.simtech.de/extensions/pandas/getLastSavedStepn
r" />
                    <wsdl:input>
                        <soap:body use="literal" />
                    </wsdl:input>
                    <wsdl:output>
                        <soap:body use="literal" />
                    </wsdl:output>
                    <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
```

```
1815                    </wsdl:fault>
1816                    <wsdl:fault name="InvalidStateFault">
1817                          <soap:fault use="literal" name="InvalidStateFault"
1818    />
1819                    </wsdl:fault>
1820              </wsdl:operation>
1821              <wsdl:operation name="getLastSavedStepnrPseudo">
1822                    <soap:operation
1823
1824          soapAction="http://wsi.simtech.de/extensions/pandas/getLastSavedStepn
1825    rPseudo" />
1826                    <wsdl:input>
1827                          <soap:body use="literal" />
1828                    </wsdl:input>
1829                    <wsdl:output>
1830                          <soap:body use="literal" />
1831                    </wsdl:output>
1832                    <wsdl:fault name="CommandFault">
1833                          <soap:fault use="literal" name="CommandFault" />
1834                    </wsdl:fault>
1835                    <wsdl:fault name="InvalidStateFault">
1836                          <soap:fault use="literal" name="InvalidStateFault"
1837    />
1838                    </wsdl:fault>
1839              </wsdl:operation>
1840              <wsdl:operation name="getMid">
1841                    <soap:operation
1842
1843          soapAction="http://wsi.simtech.de/extensions/pandas/getMid" />
1844                    <wsdl:input>
1845                          <soap:body use="literal" />
1846                    </wsdl:input>
1847                    <wsdl:output>
1848                          <soap:body use="literal" />
1849                    </wsdl:output>
1850                    <wsdl:fault name="CommandFault">
1851                          <soap:fault use="literal" name="CommandFault" />
1852                    </wsdl:fault>
1853                    <wsdl:fault name="InvalidStateFault">
1854                          <soap:fault use="literal" name="InvalidStateFault"
1855    />
1856                    </wsdl:fault>
1857              </wsdl:operation>
1858              <wsdl:operation name="getMidPseudo">
1859                    <soap:operation
1860
1861          soapAction="http://wsi.simtech.de/extensions/pandas/getMidPseudo" />
1862                    <wsdl:input>
1863                          <soap:body use="literal" />
1864                    </wsdl:input>
1865                    <wsdl:output>
1866                          <soap:body use="literal" />
1867                    </wsdl:output>
1868                    <wsdl:fault name="CommandFault">
1869                          <soap:fault use="literal" name="CommandFault" />
1870                    </wsdl:fault>
1871                    <wsdl:fault name="InvalidStateFault">
1872                          <soap:fault use="literal" name="InvalidStateFault"
1873    />
1874                    </wsdl:fault>
1875              </wsdl:operation>
1876              <wsdl:operation name="saveDataQuality">
1877                    <soap:operation
```

```
1878
1879        soapAction="http://wsi.simtech.de/extensions/pandas/saveDataQuality"
1880    />
1881                    <wsdl:input>
1882                        <soap:body use="literal" />
1883                    </wsdl:input>
1884                    <wsdl:output>
1885                        <soap:body use="literal" />
1886                    </wsdl:output>
1887                    <wsdl:fault name="CommandFault">
1888                        <soap:fault use="literal" name="CommandFault" />
1889                    </wsdl:fault>
1890                    <wsdl:fault name="InvalidStateFault">
1891                        <soap:fault use="literal" name="InvalidStateFault"
1892    />
1893                    </wsdl:fault>
1894                </wsdl:operation>
1895                <wsdl:operation name="saveMatrix">
1896                    <soap:operation
1897
1898        soapAction="http://wsi.simtech.de/extensions/pandas/saveMatrix" />
1899                    <wsdl:input>
1900                        <soap:body use="literal" />
1901                    </wsdl:input>
1902                    <wsdl:output>
1903                        <soap:body use="literal" />
1904                    </wsdl:output>
1905                    <wsdl:fault name="CommandFault">
1906                        <soap:fault use="literal" name="CommandFault" />
1907                    </wsdl:fault>
1908                    <wsdl:fault name="InvalidStateFault">
1909                        <soap:fault use="literal" name="InvalidStateFault"
1910    />
1911                    </wsdl:fault>
1912                </wsdl:operation>
1913                <wsdl:operation name="loadMatrix">
1914                    <soap:operation
1915
1916        soapAction="http://wsi.simtech.de/extensions/pandas/loadMatrix" />
1917                    <wsdl:input>
1918                        <soap:body use="literal" />
1919                    </wsdl:input>
1920                    <wsdl:output>
1921                        <soap:body use="literal" />
1922                    </wsdl:output>
1923                    <wsdl:fault name="CommandFault">
1924                        <soap:fault use="literal" name="CommandFault" />
1925                    </wsdl:fault>
1926                    <wsdl:fault name="InvalidStateFault">
1927                        <soap:fault use="literal" name="InvalidStateFault"
1928    />
1929                    </wsdl:fault>
1930                </wsdl:operation>
1931                <wsdl:operation name="prepareSimulation">
1932                    <soap:operation
1933
1934        soapAction="http://wsi.simtech.de/extensions/pandas/prepareSimulation
1935    " />
1936                    <wsdl:input>
1937                        <soap:body use="literal" />
1938                    </wsdl:input>
1939                    <wsdl:output>
1940                        <soap:body use="literal" />
```

```
1941                    </wsdl:output>
1942                    <wsdl:fault name="CommandFault">
1943                          <soap:fault use="literal" name="CommandFault" />
1944                    </wsdl:fault>
1945                    <wsdl:fault name="InvalidStateFault">
1946                          <soap:fault use="literal" name="InvalidStateFault"
1947    />
1948                    </wsdl:fault>
1949              </wsdl:operation>
1950              <wsdl:operation name="startPandas">
1951                    <soap:operation
1952
1953          soapAction="http://wsi.simtech.de/extensions/pandas/startPandas" />
1954                    <wsdl:input>
1955                          <soap:body use="literal" />
1956                    </wsdl:input>
1957                    <wsdl:output>
1958                          <soap:body use="literal" />
1959                    </wsdl:output>
1960                    <wsdl:fault name="CommandFault">
1961                          <soap:fault use="literal" name="CommandFault" />
1962                    </wsdl:fault>
1963                    <wsdl:fault name="InvalidStateFault">
1964                          <soap:fault use="literal" name="InvalidStateFault"
1965    />
1966                    </wsdl:fault>
1967              </wsdl:operation>
1968              <wsdl:operation name="getUsedParamFile">
1969                    <soap:operation
1970
1971          soapAction="http://wsi.simtech.de/extensions/pandas/getUsedParamFile"
1972    />
1973                    <wsdl:input>
1974                          <soap:body use="literal" />
1975                    </wsdl:input>
1976                    <wsdl:output>
1977                          <soap:body use="literal" />
1978                    </wsdl:output>
1979                    <wsdl:fault name="CommandFault">
1980                          <soap:fault use="literal" name="CommandFault" />
1981                    </wsdl:fault>
1982                    <wsdl:fault name="InvalidStateFault">
1983                          <soap:fault use="literal" name="InvalidStateFault"
1984    />
1985                    </wsdl:fault>
1986              </wsdl:operation>
1987              <wsdl:operation name="saveState">
1988                    <soap:operation
1989
1990          soapAction="http://wsi.simtech.de/extensions/pandas/saveState" />
1991                    <wsdl:input>
1992                          <soap:body use="literal" />
1993                    </wsdl:input>
1994                    <wsdl:output>
1995                          <soap:body use="literal" />
1996                    </wsdl:output>
1997                    <wsdl:fault name="CommandFault">
1998                          <soap:fault use="literal" name="CommandFault" />
1999                    </wsdl:fault>
2000                    <wsdl:fault name="InvalidStateFault">
2001                          <soap:fault use="literal" name="InvalidStateFault"
2002    />
2003                    </wsdl:fault>
```

```
2004              </wsdl:operation>
2005              <wsdl:operation name="loadState">
2006                    <soap:operation
2007
2008        soapAction="http://wsi.simtech.de/extensions/pandas/loadState" />
2009                    <wsdl:input>
2010                          <soap:body use="literal" />
2011                    </wsdl:input>
2012                    <wsdl:output>
2013                          <soap:body use="literal" />
2014                    </wsdl:output>
2015                    <wsdl:fault name="CommandFault">
2016                          <soap:fault use="literal" name="CommandFault" />
2017                    </wsdl:fault>
2018                    <wsdl:fault name="InvalidStateFault">
2019                          <soap:fault use="literal" name="InvalidStateFault"
2020    />
2021                    </wsdl:fault>
2022              </wsdl:operation>
2023              <wsdl:operation name="saveMeshTrans">
2024                    <soap:operation
2025
2026        soapAction="http://wsi.simtech.de/extensions/pandas/saveMeshTrans" />
2027                    <wsdl:input>
2028                          <soap:body use="literal" />
2029                    </wsdl:input>
2030                    <wsdl:output>
2031                          <soap:body use="literal" />
2032                    </wsdl:output>
2033                    <wsdl:fault name="CommandFault">
2034                          <soap:fault use="literal" name="CommandFault" />
2035                    </wsdl:fault>
2036                    <wsdl:fault name="InvalidStateFault">
2037                          <soap:fault use="literal" name="InvalidStateFault"
2038    />
2039                    </wsdl:fault>
2040              </wsdl:operation>
2041              <wsdl:operation name="loadMeshTrans">
2042                    <soap:operation
2043
2044        soapAction="http://wsi.simtech.de/extensions/pandas/loadMeshTrans" />
2045                    <wsdl:input>
2046                          <soap:body use="literal" />
2047                    </wsdl:input>
2048                    <wsdl:output>
2049                          <soap:body use="literal" />
2050                    </wsdl:output>
2051                    <wsdl:fault name="CommandFault">
2052                          <soap:fault use="literal" name="CommandFault" />
2053                    </wsdl:fault>
2054                    <wsdl:fault name="InvalidStateFault">
2055                          <soap:fault use="literal" name="InvalidStateFault"
2056    />
2057                    </wsdl:fault>
2058              </wsdl:operation>
2059              <wsdl:operation name="ReadNextValFromMeshFile">
2060                    <soap:operation
2061
2062        soapAction="http://wsi.simtech.de/extensions/pandas/ReadNextValFromMe
2063    shFile" />
2064                    <wsdl:input>
2065                          <soap:body use="literal" />
2066                    </wsdl:input>
```

```
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="WriteValToMeshFile">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/WriteValToMeshFil
e" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="saveAllGauss">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/saveAllGauss" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="loadAllGauss">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/loadAllGauss" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
```

```
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="saveGaussName">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/saveGaussName" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalidStateFault">
                        <soap:fault use="literal" name="InvalidStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
            <wsdl:operation name="loadGaussName">
                  <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pandas/loadGaussName" />
                  <wsdl:input>
                        <soap:body use="literal" />
                  </wsdl:input>
                  <wsdl:output>
                        <soap:body use="literal" />
                  </wsdl:output>
                  <wsdl:fault name="CommandFault">
                        <soap:fault use="literal" name="CommandFault" />
                  </wsdl:fault>
                  <wsdl:fault name="InvalildStateFault">
                        <soap:fault use="literal" name="InvalildStateFault"
/>
                  </wsdl:fault>
            </wsdl:operation>
      </wsdl:binding>
      <wsdl:service name="WSI_Pandas">
            <wsdl:port binding="tns:WSI_PandasSOAP" name="WSI_PandasSOAP">
                  <soap:address
location="http://localhost:8080/axis2/services/WSI_Pandas/" />
            </wsdl:port>
      </wsdl:service>

</wsdl:definitions>
```

WSDL WSI_Matlab

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <wsdl:definitions name="WSI_Matlab"
3      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4      xmlns:tns="http://wsi.simtech.de/extensions/matlab/"
5      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
6      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
      xmlns:types="http://wsi.simtech.de/ws/types/"
      targetNamespace="http://wsi.simtech.de/extensions/matlab/">

    <wsdl:types>
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://wsi.simtech.de/extensions/matlab/">

            <xsd:element name="ExecuteCommandSyncFault">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="returnCode" type="xsd:int"
minOccurs="1"
                                    maxOccurs="1" />
                              <xsd:element name="errorMessage"
type="xsd:string"
                                    minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="prepareSimulation">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="Name" type="xsd:string"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="prepareSimulationResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              <xsd:element name="SimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="start">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="User" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Host" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Path" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Program" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
```

```
            </xsd:element>

            <xsd:element name="startResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="copy">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="SrcUser" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="SrcHost" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="SrcFile" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="DstUser"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="DstHost"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="DstFile"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="copyResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="mkdir">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="User" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Host" type="xsd:string"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
```

```
                              <xsd:element name="Dir"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="mkdirResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="setMatlabPath">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                              <xsd:element name="MatlabPath"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="setMatlabPathResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getMatlabPath">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getMatlabPathResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="MatlabPath"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="deleteFile">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                              <xsd:element name="User"
```

```
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Host"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="File"
                                    type="xsd:string" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="deleteFileResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

      </xsd:schema>

      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://wsi.simtech.de/ws/types/"
                  schemaLocation="types.xsd">
            </xsd:import>
      </xsd:schema>
  </wsdl:types>

  <wsdl:message name="prepareSimulationRequest">
      <wsdl:part name="parameters"
element="tns:prepareSimulation"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="prepareSimulationResponse">
      <wsdl:part name="parameters"
element="tns:prepareSimulationResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="startRequest">
      <wsdl:part name="parameters" element="tns:start"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="startResponse">
      <wsdl:part name="parameters" element="tns:startResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="copyRequest">
      <wsdl:part name="parameters" element="tns:copy"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="copyResponse">
      <wsdl:part name="parameters" element="tns:copyResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="mkdirRequest">
      <wsdl:part name="parameters" element="tns:mkdir"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="mkdirResponse">
      <wsdl:part name="parameters" element="tns:mkdirResponse"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="setMatlabPathRequest">
      <wsdl:part name="parameters" element="tns:setMatlabPath"></wsdl:part>
  </wsdl:message>
  <wsdl:message name="setMatlabPathResponse">
```

```
259         <wsdl:part name="parameters"
260   element="tns:setMatlabPathResponse"></wsdl:part>
261     </wsdl:message>
262     <wsdl:message name="getMatlabPathRequest">
263         <wsdl:part name="parameters" element="tns:getMatlabPath"></wsdl:part>
264     </wsdl:message>
265     <wsdl:message name="getMatlabPathResponse">
266         <wsdl:part name="parameters"
267   element="tns:getMatlabPathResponse"></wsdl:part>
268     </wsdl:message>
269     <wsdl:message name="InvalidStateException">
270       <wsdl:part name="fault" element="types:InvalidStateFault"></wsdl:part>
271     </wsdl:message>
272     <wsdl:message name="ExecuteCommandException">
273       <wsdl:part name="fault"
274   element="tns:ExecuteCommandSyncFault"></wsdl:part>
275     </wsdl:message>
276     <wsdl:message name="deleteFileRequest">
277         <wsdl:part name="parameters" element="tns:deleteFile"></wsdl:part>
278     </wsdl:message>
279     <wsdl:message name="deleteFileResponse">
280         <wsdl:part name="parameters"
281   element="tns:deleteFileResponse"></wsdl:part>
282     </wsdl:message>
283
284
285     <wsdl:portType name="WSI_Matlab">
286
287       <wsdl:operation name="prepareSimulation">
288           <wsdl:input message="tns:prepareSimulationRequest"></wsdl:input>
289           <wsdl:output message="tns:prepareSimulationResponse"></wsdl:output>
290               <wsdl:fault name="InvalidStateFault"
291   message="tns:InvalidStateException"></wsdl:fault>
292               <wsdl:fault name="CommandFault"
293   message="tns:ExecuteCommandException"></wsdl:fault>
294           </wsdl:operation>
295
296       <wsdl:operation name="start">
297         <wsdl:input message="tns:startRequest"></wsdl:input>
298         <wsdl:output message="tns:startResponse"></wsdl:output>
299               <wsdl:fault name="InvalidStateFault"
300   message="tns:InvalidStateException"></wsdl:fault>
301               <wsdl:fault name="CommandFault"
302   message="tns:ExecuteCommandException"></wsdl:fault>
303           </wsdl:operation>
304
305       <wsdl:operation name="copy">
306         <wsdl:input message="tns:copyRequest"></wsdl:input>
307         <wsdl:output message="tns:copyResponse"></wsdl:output>
308               <wsdl:fault name="InvalidStateFault"
309   message="tns:InvalidStateException"></wsdl:fault>
310               <wsdl:fault name="CommandFault"
311   message="tns:ExecuteCommandException"></wsdl:fault>
312           </wsdl:operation>
313
314       <wsdl:operation name="mkdir">
315         <wsdl:input message="tns:mkdirRequest"></wsdl:input>
316         <wsdl:output message="tns:mkdirResponse"></wsdl:output>
317               <wsdl:fault name="InvalidStateFault"
318   message="tns:InvalidStateException"></wsdl:fault>
319               <wsdl:fault name="CommandFault"
320   message="tns:ExecuteCommandException"></wsdl:fault>
321           </wsdl:operation>
```

```
322
323         <wsdl:operation name="deleteFile">
324             <wsdl:input message="tns:deleteFileRequest"></wsdl:input>
325             <wsdl:output message="tns:deleteFileResponse"></wsdl:output>
326                 <wsdl:fault name="InvalidStateFault"
327     message="tns:InvalidStateException"></wsdl:fault>
328                 <wsdl:fault name="CommandFault"
329     message="tns:ExecuteCommandException"></wsdl:fault>
330             </wsdl:operation>
331
332         <wsdl:operation name="setMatlabPath">
333           <wsdl:input message="tns:setMatlabPathRequest"></wsdl:input>
334           <wsdl:output message="tns:setMatlabPathResponse"></wsdl:output>
335                 <wsdl:fault name="InvalidStateFault"
336     message="tns:InvalidStateException"></wsdl:fault>
337                 <wsdl:fault name="CommandFault"
338     message="tns:ExecuteCommandException"></wsdl:fault>
339             </wsdl:operation>
340
341         <wsdl:operation name="getMatlabPath">
342           <wsdl:input message="tns:getMatlabPathRequest"></wsdl:input>
343           <wsdl:output message="tns:getMatlabPathResponse"></wsdl:output>
344                 <wsdl:fault name="InvalidStateFault"
345     message="tns:InvalidStateException"></wsdl:fault>
346                 <wsdl:fault name="CommandFault"
347     message="tns:ExecuteCommandException"></wsdl:fault>
348             </wsdl:operation>
349       </wsdl:portType>
350
351
352       <wsdl:binding name="WSI_MatlabSOAP" type="tns:WSI_Matlab">
353
354
355           <soap:binding style="document"
356                 transport="http://schemas.xmlsoap.org/soap/http" />
357           <wsdl:operation name="prepareSimulation">
358
359                 <soap:operation
360
361           soapAction="http://wsi.simtech.de/extensions/matlab/prepareSimulation
362     " />
363                 <wsdl:input>
364                       <soap:body use="literal" />
365                 </wsdl:input>
366                 <wsdl:output>
367                       <soap:body use="literal" />
368                 </wsdl:output>
369                 <wsdl:fault name="InvalidStateFault">
370                       <soap:fault use="literal" name="InvalidStateFault" />
371                 </wsdl:fault>
372                 <wsdl:fault name="CommandFault">
373                       <soap:fault use="literal" name="CommandFault" />
374                 </wsdl:fault>
375           </wsdl:operation>
376
377           <wsdl:operation name="start">
378                 <soap:operation
379
380           soapAction="http://wsi.simtech.de/extensions/matlab/start" />
381                 <wsdl:input>
382                       <soap:body use="literal" />
383                 </wsdl:input>
384                 <wsdl:output>
```

```
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="copy">
            <soap:operation
                  soapAction="http://wsi.simtech.de/extensions/matlab/copy"
/>
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="mkdir">
            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/matlab/mkdir" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="setMatlabPath">
            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/matlab/setMatlabPath" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>
```

```
448
449         <wsdl:operation name="getMatlabPath">
450               <soap:operation
451
452         soapAction="http://wsi.simtech.de/extensions/matlab/getMatlabPath" />
453               <wsdl:input>
454                     <soap:body use="literal" />
455               </wsdl:input>
456               <wsdl:output>
457                     <soap:body use="literal" />
458               </wsdl:output>
459               <wsdl:fault name="InvalidStateFault">
460                     <soap:fault use="literal" name="InvalidStateFault" />
461               </wsdl:fault>
462               <wsdl:fault name="CommandFault">
463                     <soap:fault use="literal" name="CommandFault" />
464               </wsdl:fault>
465         </wsdl:operation>
466
467         <wsdl:operation name="deleteFile">
468               <soap:operation
469
470         soapAction="http://wsi.simtech.de/extensions/matlab/deleteFile" />
471               <wsdl:input>
472                     <soap:body use="literal" />
473               </wsdl:input>
474               <wsdl:output>
475                     <soap:body use="literal" />
476               </wsdl:output>
477               <wsdl:fault name="InvalidStateFault">
478                     <soap:fault use="literal" name="InvalidStateFault" />
479               </wsdl:fault>
480               <wsdl:fault name="CommandFault">
481                     <soap:fault use="literal" name="CommandFault" />
482               </wsdl:fault>
483         </wsdl:operation>
484     </wsdl:binding>
485
486     <wsdl:service name="WSI_Matlab">
487       <wsdl:port binding="tns:WSI_MatlabSOAP" name="WSI_MatlabSOAP">
488         <soap:address
489   location="http://localhost:8080/axis2/services/WSI_Matlab"/>
490       </wsdl:port>
491     </wsdl:service>
492
493   </wsdl:definitions>
```

WSDL WSI_PMConnector

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <wsdl:definitions name="WSI_PMConnector"
3       xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
4       xmlns:tns="http://wsi.simtech.de/extensions/pmconn/"
5       xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
6       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7       xmlns:types="http://wsi.simtech.de/ws/types/"
8       targetNamespace="http://wsi.simtech.de/extensions/pmconn/">
9
```

```
  <wsdl:types>
      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://wsi.simtech.de/extensions/pmconn/">

            <xsd:element name="ExecuteCommandSyncFault">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="returnCode" type="xsd:int"
minOccurs="1"
                                    maxOccurs="1" />
                              <xsd:element name="errorMessage"
type="xsd:string"
                                    minOccurs="1" maxOccurs="1" />
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>


            <xsd:element name="getCountGauss">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="FromSimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Stepnr" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getCountGaussResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="Count" type="xsd:int"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="createList">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="FromSimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Stepnr" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="StartElement"
type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="EndElement" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
```

```
                              <xsd:element name="Time"
                                    type="xsd:int" maxOccurs="1"
minOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="createListResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="insertList">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    maxOccurs="1" minOccurs="1">
                              </xsd:element>
                              <xsd:element name="Stepnr" type="xsd:int"
                                    maxOccurs="1" minOccurs="1">
                              </xsd:element>
                              <xsd:element name="StartElement"
type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="EndElement" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="CountGauss" type="xsd:int"
                                    maxOccurs="1" minOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="insertListResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="prepareSimulation">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="Name" type="xsd:string"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="prepareSimulationResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
```

```
                              <xsd:element name="SimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getCountElement">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    maxOccurs="1" minOccurs="1">
                              </xsd:element>
                              <xsd:element name="FromSimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Stepnr" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getCountElementResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="Count" type="xsd:int"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getCountAllValues">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="FromSimID"
                                    type="xsd:long" minOccurs="1"
maxOccurs="1">
                              </xsd:element>
                              <xsd:element name="Stepnr" type="xsd:int"
                                    minOccurs="1" maxOccurs="1">
                              </xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="getCountAllValuesResponse">
                  <xsd:complexType>
                        <xsd:sequence>
                              <xsd:element name="Count" type="xsd:int"
minOccurs="1" maxOccurs="1"></xsd:element>
                        </xsd:sequence>
                  </xsd:complexType>
            </xsd:element>

            <xsd:element name="setDBConn">
                  <xsd:complexType>
                        <xsd:sequence>
```

```
                                    <xsd:element name="SimID" type="xsd:long"
                                    minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="Host" type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="DB" type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="User"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="PW"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>

                  <xsd:element name="setDBConnResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="ReturnMessage"
type="xsd:string" minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>

                  <xsd:element name="getDBConn">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="SimID" type="xsd:long"
minOccurs="1" maxOccurs="1"></xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>

                  <xsd:element name="getDBConnResponse">
                        <xsd:complexType>
                              <xsd:sequence>
                                    <xsd:element name="Host" type="xsd:string"
                                          minOccurs="1" maxOccurs="1">
                                    </xsd:element>
                                    <xsd:element name="DB"
                                          type="xsd:string" minOccurs="1"
maxOccurs="1">
                                    </xsd:element>
                              </xsd:sequence>
                        </xsd:complexType>
                  </xsd:element>

            </xsd:schema>

            <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                  <xsd:import namespace="http://wsi.simtech.de/ws/types/"
                        schemaLocation="types.xsd">
                  </xsd:import>
            </xsd:schema>
      </wsdl:types>
```

```
<wsdl:message name="getCountGaussRequest">
    <wsdl:part name="parameters" element="tns:getCountGauss"></wsdl:part>
</wsdl:message>
<wsdl:message name="getCountGaussResponse">
    <wsdl:part name="parameters"
element="tns:getCountGaussResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="createListRequest">
    <wsdl:part name="parameters" element="tns:createList"></wsdl:part>
</wsdl:message>
<wsdl:message name="createListResponse">
    <wsdl:part name="parameters"
element="tns:createListResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="insertListRequest">
    <wsdl:part name="parameters" element="tns:insertList"></wsdl:part>
</wsdl:message>
<wsdl:message name="insertListResponse">
    <wsdl:part name="parameters"
element="tns:insertListResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="prepareSimulationRequest">
    <wsdl:part name="parameters"
element="tns:prepareSimulation"></wsdl:part>
</wsdl:message>
<wsdl:message name="prepareSimulationResponse">
    <wsdl:part name="parameters"
element="tns:prepareSimulationResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getCountElementRequest">
    <wsdl:part name="parameters"
element="tns:getCountElement"></wsdl:part>
</wsdl:message>
<wsdl:message name="getCountElementResponse">
    <wsdl:part name="parameters"
element="tns:getCountElementResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getCountAllValuesRequest">
    <wsdl:part name="parameters"
element="tns:getCountAllValues"></wsdl:part>
</wsdl:message>
<wsdl:message name="getCountAllValuesResponse">
    <wsdl:part name="parameters"
element="tns:getCountAllValuesResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="InvalidStateException">
  <wsdl:part name="fault" element="types:InvalidStateFault"></wsdl:part>
</wsdl:message>
<wsdl:message name="ExecuteCommandException">
  <wsdl:part name="fault"
element="tns:ExecuteCommandSyncFault"></wsdl:part>
</wsdl:message>

<wsdl:message name="setDBConnRequest">
    <wsdl:part name="parameters" element="tns:setDBConn"></wsdl:part>
</wsdl:message>
<wsdl:message name="setDBConnResponse">
    <wsdl:part name="parameters"
element="tns:setDBConnResponse"></wsdl:part>
</wsdl:message>
<wsdl:message name="getDBConnRequest">
```

```
325         <wsdl:part name="parameters" element="tns:getDBConn"></wsdl:part>
326       </wsdl:message>
327       <wsdl:message name="getDBConnResponse">
328           <wsdl:part name="parameters"
329   element="tns:getDBConnResponse"></wsdl:part>
330       </wsdl:message>
331
332
333
334
335       <wsdl:portType name="WSI_PMConnector">
336
337         <wsdl:operation name="prepareSimulation">
338             <wsdl:input message="tns:prepareSimulationRequest"></wsdl:input>
339             <wsdl:output message="tns:prepareSimulationResponse"></wsdl:output>
340                 <wsdl:fault name="InvalidStateFault"
341   message="tns:InvalidStateException"></wsdl:fault>
342                 <wsdl:fault name="CommandFault"
343   message="tns:ExecuteCommandException"></wsdl:fault>
344           </wsdl:operation>
345
346         <wsdl:operation name="getCountGauss">
347           <wsdl:input message="tns:getCountGaussRequest"></wsdl:input>
348           <wsdl:output message="tns:getCountGaussResponse"></wsdl:output>
349                 <wsdl:fault name="InvalidStateFault"
350   message="tns:InvalidStateException"></wsdl:fault>
351                 <wsdl:fault name="CommandFault"
352   message="tns:ExecuteCommandException"></wsdl:fault>
353           </wsdl:operation>
354
355         <wsdl:operation name="getCountElement">
356             <wsdl:input message="tns:getCountElementRequest"></wsdl:input>
357             <wsdl:output message="tns:getCountElementResponse"></wsdl:output>
358                 <wsdl:fault name="InvalidStateFault"
359   message="tns:InvalidStateException"></wsdl:fault>
360                 <wsdl:fault name="CommandFault"
361   message="tns:ExecuteCommandException"></wsdl:fault>
362           </wsdl:operation>
363
364         <wsdl:operation name="createList">
365           <wsdl:input message="tns:createListRequest"></wsdl:input>
366           <wsdl:output message="tns:createListResponse"></wsdl:output>
367                 <wsdl:fault name="InvalidStateFault"
368   message="tns:InvalidStateException"></wsdl:fault>
369                 <wsdl:fault name="CommandFault"
370   message="tns:ExecuteCommandException"></wsdl:fault>
371           </wsdl:operation>
372
373         <wsdl:operation name="insertList">
374           <wsdl:input message="tns:insertListRequest"></wsdl:input>
375           <wsdl:output message="tns:insertListResponse"></wsdl:output>
376                 <wsdl:fault name="InvalidStateFault"
377   message="tns:InvalidStateException"></wsdl:fault>
378                 <wsdl:fault name="CommandFault"
379   message="tns:ExecuteCommandException"></wsdl:fault>
380           </wsdl:operation>
381
382         <wsdl:operation name="getCountAllValues">
383           <wsdl:input message="tns:getCountAllValuesRequest"></wsdl:input>
384           <wsdl:output message="tns:getCountAllValuesResponse"></wsdl:output>
385                 <wsdl:fault name="InvalidStateFault"
386   message="tns:InvalidStateException"></wsdl:fault>
```

```
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
        </wsdl:operation>
    <wsdl:operation name="setDBConn">
      <wsdl:input message="tns:setDBConnRequest"></wsdl:input>
      <wsdl:output message="tns:setDBConnResponse"></wsdl:output>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
        </wsdl:operation>
    <wsdl:operation name="getDBConn">
      <wsdl:input message="tns:getDBConnRequest"></wsdl:input>
      <wsdl:output message="tns:getDBConnResponse"></wsdl:output>
            <wsdl:fault name="InvalidStateFault"
message="tns:InvalidStateException"></wsdl:fault>
            <wsdl:fault name="CommandFault"
message="tns:ExecuteCommandException"></wsdl:fault>
        </wsdl:operation>
  </wsdl:portType>


  <wsdl:binding name="WSI_PMConnectorSOAP" type="tns:WSI_PMConnector">

      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="getCountGauss">

            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pmconn/getCountGauss" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="createList">
            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pmconn/createList" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

```

```
450         <wsdl:operation name="insertList">
451               <soap:operation
452
453         soapAction="http://wsi.simtech.de/extensions/pmconn/insertList" />
454               <wsdl:input>
455                     <soap:body use="literal" />
456               </wsdl:input>
457               <wsdl:output>
458                     <soap:body use="literal" />
459               </wsdl:output>
460               <wsdl:fault name="InvalidStateFault">
461                     <soap:fault use="literal" name="InvalidStateFault" />
462               </wsdl:fault>
463               <wsdl:fault name="CommandFault">
464                     <soap:fault use="literal" name="CommandFault" />
465               </wsdl:fault>
466         </wsdl:operation>
467
468
469         <wsdl:operation name="prepareSimulation">
470               <soap:operation
471
472         soapAction="http://wsi.simtech.de/extensions/pmconn/prepareSimulation
473   " />
474               <wsdl:input>
475                     <soap:body use="literal" />
476               </wsdl:input>
477               <wsdl:output>
478                     <soap:body use="literal" />
479               </wsdl:output>
480               <wsdl:fault name="InvalidStateFault">
481                     <soap:fault use="literal" name="InvalidStateFault" />
482               </wsdl:fault>
483               <wsdl:fault name="CommandFault">
484                     <soap:fault use="literal" name="CommandFault" />
485               </wsdl:fault>
486         </wsdl:operation>
487
488         <wsdl:operation name="getCountElement">
489               <soap:operation
490
491         soapAction="http://wsi.simtech.de/extensions/pmconn/getCountElement"
492   />
493               <wsdl:input>
494                     <soap:body use="literal" />
495               </wsdl:input>
496               <wsdl:output>
497                     <soap:body use="literal" />
498               </wsdl:output>
499               <wsdl:fault name="InvalidStateFault">
500                     <soap:fault use="literal" name="InvalidStateFault" />
501               </wsdl:fault>
502               <wsdl:fault name="CommandFault">
503                     <soap:fault use="literal" name="CommandFault" />
504               </wsdl:fault>
505         </wsdl:operation>
506
507         <wsdl:operation name="getCountAllValues">
508               <soap:operation
509
510         soapAction="http://wsi.simtech.de/extensions/pmconn/getCountAllValues
511   " />
512               <wsdl:input>
```

```
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="setDBConn">
            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pmconn/setDBConn" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>

      <wsdl:operation name="getDBConn">
            <soap:operation

      soapAction="http://wsi.simtech.de/extensions/pmconn/getDBConn" />
            <wsdl:input>
                  <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                  <soap:body use="literal" />
            </wsdl:output>
            <wsdl:fault name="InvalidStateFault">
                  <soap:fault use="literal" name="InvalidStateFault" />
            </wsdl:fault>
            <wsdl:fault name="CommandFault">
                  <soap:fault use="literal" name="CommandFault" />
            </wsdl:fault>
      </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="WSI_PMConnector">
    <wsdl:port binding="tns:WSI_PMConnectorSOAP"
name="WSI_PMConnectorSOAP">
      <soap:address
location="http://localhost:8080/axis2/services/WSI_PMConnector"/>
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

WSDL Data-Quality

```
1  <?xml version="1.0"?>
2  <definitions name="DQ-WF"
3          targetNamespace="http://wsi.simtech.de/processes/samples/DQ-WF"
4          xmlns:tns="http://wsi.simtech.de/processes/samples/DQ-WF"
5          xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
6          xmlns="http://schemas.xmlsoap.org/wsdl/"
7          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
8
9  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
10       TYPE DEFINITION - List of types participating in this BPEL process
11       The BPEL Designer will generate default request and response types
12       but you can define or import any XML Schema type and use them as part
13       of the message types.
14       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
15 -->
16      <types>
17          <schema attributeFormDefault="unqualified"
18 elementFormDefault="qualified"
19
20 targetNamespace="http://wsi.simtech.de/processes/samples/DQ-WF"
21                  xmlns="http://www.w3.org/2001/XMLSchema">
22
23              <element name="DQ-WFRequest">
24                  <complexType>
25                      <sequence>
26                          <element name="ProblemName" type="string"
27                                minOccurs="1" maxOccurs="1" />
28                          <element name="CmdFile" type="string"
29                                minOccurs="1" maxOccurs="1">
30                          </element>
31                          <element name="StepWidth" type="int"
32                                minOccurs="1" maxOccurs="1">
33                          </element>
34
35                      </sequence>
36                  </complexType>
37              </element>
38
39              <element name="DQ-WFResponse">
40                  <complexType>
41                      <sequence>
42                          <element name="result" type="string"/>
43                      </sequence>
44                  </complexType>
45              </element>
46          </schema>
47      </types>
48
49
50 <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
51       MESSAGE TYPE DEFINITION - Definition of the message types used as
52       part of the port type defintions
53       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
54 -->
55      <message name="DQ-WFRequestMessage">
56          <part name="payload" element="tns:DQ-WFRequest"/>
57      </message>
58      <message name="DQ-WFResponseMessage">
59          <part name="payload" element="tns:DQ-WFResponse"/>
```

```
    </message>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
     a logical service unit.
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->

    <!-- portType implemented by the DQ-WF BPEL process -->
    <portType name="DQ-WF">
        <operation name="process">
            <input  message="tns:DQ-WFRequestMessage" />
            <output message="tns:DQ-WFResponseMessage"/>
        </operation>
    </portType>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="DQ-WF">
        <plnk:role name="DQ-WFProvider" portType="tns:DQ-WF"/>
    </plnk:partnerLinkType>

    <binding name="DQ-WFBinding" type="tns:DQ-WF">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="process">
            <soap:operation
                  soapAction="http://wsi.simtech.de/processes/samples/DQ-
WF/process" />
            <input>
                  <soap:body use="literal" />
            </input>
            <output>
                  <soap:body use="literal" />
            </output>
      </operation>
    </binding>
    <service name="DQ-WFService">
      <port name="DQ-WF" binding="tns:DQ-WFBinding">
            <soap:address location="http://localhost:8080/ode/processes/DQ-
WF" />
      </port>
    </service>
</definitions>
```

WSDL TwoInstances

```
<?xml version="1.0"?>
<definitions name="twoinstances"

targetNamespace="http://wsi.simtech.de/processes/samples/twoinstances"
        xmlns:tns="http://wsi.simtech.de/processes/samples/twoinstances"
        xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
        xmlns="http://schemas.xmlsoap.org/wsdl/"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
11        TYPE DEFINITION - List of types participating in this BPEL process
12        The BPEL Designer will generate default request and response types
13        but you can define or import any XML Schema type and use them as part
14        of the message types.
15        ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
16    -->
17        <types>
18            <schema attributeFormDefault="unqualified"
19    elementFormDefault="qualified"
20
21    targetNamespace="http://wsi.simtech.de/processes/samples/twoinstances"
22                    xmlns="http://www.w3.org/2001/XMLSchema">
23
24                <element name="twoinstancesRequest">
25                    <complexType>
26                        <sequence>
27                            <element name="Problem-SimA" type="string"
28                                  minOccurs="1" maxOccurs="1" />
29                            <element name="CmdFile-SimA" type="string"
30                                  minOccurs="1" maxOccurs="1">
31                            </element>
32                            <element name="StepWidth-SimA" type="int"
33                                  minOccurs="1" maxOccurs="1">
34                            </element>
35                            <element name="Problem-SimB" type="string"
36                                  minOccurs="1" maxOccurs="1">
37                            </element>
38                            <element name="CmdFile-SimB" type="string"
39    minOccurs="1" maxOccurs="1"></element>
40                            <element name="StepWidth-SimB" type="int"
41                                  minOccurs="1" maxOccurs="1">
42                            </element>
43                        </sequence>
44                    </complexType>
45                </element>
46
47                <element name="twoinstancesResponse">
48                    <complexType>
49                        <sequence>
50                            <element name="result" type="string"/>
51                        </sequence>
52                    </complexType>
53                </element>
54            </schema>
55        </types>
56
57
58    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
59         MESSAGE TYPE DEFINITION - Definition of the message types used as
60         part of the port type defintions
61         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
62    -->
63        <message name="twoinstancesRequestMessage">
64            <part name="payload" element="tns:twoinstancesRequest"/>
65        </message>
66        <message name="twoinstancesResponseMessage">
67            <part name="payload" element="tns:twoinstancesResponse"/>
68        </message>
69
70    <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
71         PORT TYPE DEFINITION - A port type groups a set of operations into
72         a logical service unit.
```

```
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->

    <!-- portType implemented by the twoinstances BPEL process -->
    <portType name="twoinstances">
        <operation name="process">
            <input  message="tns:twoinstancesRequestMessage" />
            <output message="tns:twoinstancesResponseMessage"/>
        </operation>
    </portType>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="twoinstances">
        <plnk:role name="twoinstancesProvider"
portType="tns:twoinstances"/>
    </plnk:partnerLinkType>

    <binding name="twoinstancesBinding" type="tns:twoinstances">
      <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
      <operation name="process">
            <soap:operation

      soapAction="http://wsi.simtech.de/processes/samples/twoinstances/proc
ess" />
            <input>
                  <soap:body use="literal" />
            </input>
            <output>
                  <soap:body use="literal" />
            </output>
      </operation>
    </binding>
    <service name="twoinstances">
      <port name="twoinstancesPort" binding="tns:twoinstancesBinding">
            <soap:address
location="http://localhost:8080/ode/processes/twoinstances" />
      </port>
    </service>

    <service name="twoinstances_vm">
        <port name="twoinstancesPort_vm" binding="tns:twoinstancesBinding">
            <soap:address
location="http://192.168.1.112:8080/ode/processes/twoinstances" />
        </port>
    </service>

</definitions>
```

WSDL Pandas-Bone

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://wsi.simtech.de/processes/samples/Pandas-Bone"
```

```
6   xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
7   xmlns:wsdl="http://wsi.simtech.de/extensions/pandas/"
8   xmlns:wsdl1="http://wsi.simtech.de/processes/samples/Matlab-Dispatcher"
9   xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Pandas-Bone"
10  targetNamespace="http://wsi.simtech.de/processes/samples/Pandas-Bone">
11
12  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
13       TYPE DEFINITION - List of types participating in this BPEL process
14       The BPEL Designer will generate default request and response types
15       but you can define or import any XML Schema type and use them as part
16       of the message types.
17       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
18  -->
19      <plnk:partnerLinkType name="WSI_Pandas">
20      <plnk:role name="WSI_Pandas" portType="wsdl:WSI_Pandas"/>
21    </plnk:partnerLinkType>
22      <plnk:partnerLinkType name="Matlab-Dispatcher">
23      <plnk:role name="Matlab-Dispatcher" portType="wsdl1:Matlab-
24  Dispatcher"/>
25    </plnk:partnerLinkType>
26      <import location="WSI_Pandas.wsdl"
27  namespace="http://wsi.simtech.de/extensions/pandas/"/>
28      <import location="Matlab-DispatcherArtifacts.wsdl"
29  namespace="http://wsi.simtech.de/processes/samples/Matlab-Dispatcher"/>
30      <types>
31          <schema xmlns="http://www.w3.org/2001/XMLSchema"
32  attributeFormDefault="unqualified" elementFormDefault="qualified"
33  targetNamespace="http://wsi.simtech.de/processes/samples/Pandas-Bone">
34
35              <element name="Pandas-BoneRequest">
36                  <complexType>
37                      <sequence>
38                          <element maxOccurs="1" minOccurs="1"
39  name="PandasData" type="tns:PandasDataType"/>
40                          <element maxOccurs="1" minOccurs="1"
41  name="MatlabData" type="tns:MatlabDataType"/>
42                          <element maxOccurs="1" minOccurs="1"
43  name="WSINodeData" type="tns:WSINodeDataType"/>
44                      </sequence>
45                  </complexType>
46              </element>
47
48              <element name="Pandas-BoneResponse">
49                  <complexType>
50                      <sequence>
51                          <element name="result" type="string"/>
52                      </sequence>
53                  </complexType>
54              </element>
55
56              <complexType name="PandasDataType">
57                    <sequence>
58                          <element maxOccurs="1" minOccurs="1"
59  name="ProblemName" type="string">
60                          </element>
61                          <element maxOccurs="1" minOccurs="1" name="CmdFile"
62  type="string">
63                          </element>
64                          <element maxOccurs="1" minOccurs="1"
65  name="Step_Width" type="int">
66                          </element>
67                          <element maxOccurs="1" minOccurs="1"
68  name="End_Step" type="int"/>
```

```
                    </sequence>
               </complexType>

               <complexType name="MatlabDataType">
                    <sequence>
                         <element maxOccurs="unbounded" minOccurs="1"
name="MatlabNodes" type="tns:MatlabNodeType"/>
                         <element maxOccurs="1" minOccurs="1" name="SimData"
type="tns:SimDataType"/>
                         <element maxOccurs="1" minOccurs="1" name="t_end"
type="int"/>
                    </sequence>
               </complexType>

               <complexType name="WSINodeDataType">
                    <sequence>
                         <element maxOccurs="1" minOccurs="1" name="User"
type="string"/>
                         <element maxOccurs="1" minOccurs="1" name="Host"
type="string"/>
                    </sequence>
               </complexType>

               <complexType name="MatlabNodeType">
                    <sequence>
                         <element maxOccurs="1" minOccurs="1" name="User"
type="string"/>
                         <element maxOccurs="1" minOccurs="1" name="Host"
type="string"/>
                         <element maxOccurs="1" minOccurs="1" name="SimPath"
type="string"/>
                         <element maxOccurs="1" minOccurs="1"
name="MatlabPath" type="string"/>
                    </sequence>
               </complexType>

               <complexType name="SimDataType">
                    <sequence>
                         <element maxOccurs="1" minOccurs="1"
name="SrcArchive" type="string"/>
                         <element maxOccurs="1" minOccurs="1" name="mFile"
type="string"/>
                    </sequence>
               </complexType>
          </schema>
     </types>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      MESSAGE TYPE DEFINITION - Definition of the message types used as
      part of the port type defintions
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
     <message name="Pandas-BoneRequestMessage">
          <part element="tns:Pandas-BoneRequest" name="payload"/>
     </message>
     <message name="Pandas-BoneResponseMessage">
          <part element="tns:Pandas-BoneResponse" name="payload"/>
     </message>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      PORT TYPE DEFINITION - A port type groups a set of operations into
      a logical service unit.
```

```
132         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
133     -->
134
135         <!-- portType implemented by the Pandas-Bone BPEL process -->
136         <portType name="Pandas-Bone">
137             <operation name="process">
138                 <input message="tns:Pandas-BoneRequestMessage"/>
139                 <output message="tns:Pandas-BoneResponseMessage"/>
140             </operation>
141         </portType>
142
143
144     <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
145          PARTNER LINK TYPE DEFINITION
146          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
147     -->
148         <plnk:partnerLinkType name="Pandas-Bone">
149             <plnk:role name="Pandas-BoneProvider" portType="tns:Pandas-Bone"/>
150         </plnk:partnerLinkType>
151
152         <binding name="Pandas-BoneBinding" type="tns:Pandas-Bone">
153           <soap:binding style="document"
154     transport="http://schemas.xmlsoap.org/soap/http"/>
155           <operation name="process">
156                 <soap:operation
157     soapAction="http://wsi.simtech.de/processes/samples/Pandas-Bone/process"/>
158                 <input>
159                       <soap:body use="literal"/>
160                 </input>
161                 <output>
162                       <soap:body use="literal"/>
163                 </output>
164           </operation>
165         </binding>
166         <service name="Pandas-BoneService">
167           <port binding="tns:Pandas-BoneBinding" name="Pandas-BonePort">
168                 <soap:address
169     location="http://localhost:8080/ode/processes/Pandas-Bone"/>
170           </port>
171         </service>
172     </definitions>
```

WSDL Data-Manager

```
1   <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2   <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
3   xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
4   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
5   xmlns:tns="http://wsi.simtech.de/processes/samples/Matlab-Dispatcher"
6   xmlns:types="http://wsi.simtech.de/ws/types/"
7   xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
8   xmlns:wsdl="http://wsi.simtech.de/extensions/pmconn/"
9   xmlns:wsdl1="http://wsi.simtech.de/extensions/matlab/"
10  xmlns:wsdl2="http://wsi.simtech.de/processes/samples/Matlab-Bone"
11  xmlns:wsdl4="http://wsi.simtech.de/extensions/pandas/"
12  xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Matlab-Dispatcher"
13  targetNamespace="http://wsi.simtech.de/processes/samples/Matlab-
14  Dispatcher">
15
16  <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
      TYPE DEFINITION - List of services participating in this BPEL process
      The default output of the BPEL designer uses strings as input and
      output to the BPEL Process. But you can define or import any XML
      Schema type and us them as part of the message types.
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="WSI_PMConnector">
      <plnk:role name="WSI_PMConnector" portType="wsdl:WSI_PMConnector"/>
    </plnk:partnerLinkType>
    <plnk:partnerLinkType name="WSI_Matlab">
      <plnk:role name="WSI_Matlab" portType="wsdl1:WSI_Matlab"/>
    </plnk:partnerLinkType>
    <plnk:partnerLinkType name="WSI_Pandas">
      <plnk:role name="WSI_Pandas" portType="wsdl4:WSI_Pandas"/>
    </plnk:partnerLinkType>

    <vprop:property name="PandasSimID" type="xsd:long"/>

    <vprop:propertyAlias messageType="tns:Matlab-DispatcherResponseMessage"
part="payload" propertyName="tns:PandasSimID">
    <vprop:query><![CDATA[/tns:SimID-Pandas]]></vprop:query>
  </vprop:propertyAlias>
    <vprop:propertyAlias messageType="tns:startMatlabSimRequest"
part="parameters" propertyName="tns:PandasSimID">
    <vprop:query><![CDATA[/tns:SimIDPandas]]></vprop:query>
  </vprop:propertyAlias>
    <vprop:propertyAlias messageType="tns:stopRequest" part="parameters"
propertyName="tns:PandasSimID">
    <vprop:query><![CDATA[/tns:SimID-Pandas]]></vprop:query>
  </vprop:propertyAlias>
    <vprop:propertyAlias messageType="wsdl2:Matlab-BoneResponseMessage"
part="payload" propertyName="tns:PandasSimID">
    <vprop:query><![CDATA[/wsdl2:PandasSimID]]></vprop:query>
  </vprop:propertyAlias>
    <import location="WSI_PMConnector.wsdl"
namespace="http://wsi.simtech.de/extensions/pmconn/"/>
    <import location="WSI_Matlab.wsdl"
namespace="http://wsi.simtech.de/extensions/matlab/"/>
    <import location="WSI_Pandas.wsdl"
namespace="http://wsi.simtech.de/extensions/pandas/"/>
    <import location="Matlab-BoneArtifacts.wsdl"
namespace="http://wsi.simtech.de/processes/samples/Matlab-Bone"/>
    <types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://wsi.simtech.de/processes/samples/Matlab-
Dispatcher">

            <element name="Matlab-DispatcherRequest">
                <complexType>
                    <sequence>
                        <element maxOccurs="1" minOccurs="1" name="SimID-
Pandas" type="long"/>
                        <element maxOccurs="unbounded" minOccurs="1"
name="MatlabNode" type="tns:MatlabNodeType">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="SimNode"
type="tns:SimNodeType">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="SimData"
type="tns:SimDataType"/>
                    </sequence>
                </complexType>
```

```
            </element>

            <element name="Matlab-DispatcherResponse">
                <complexType>
                    <sequence>
                        <element name="result" type="string"/>
                        <element maxOccurs="1" minOccurs="1" name="SimID-
Pandas" type="long"/>
                    </sequence>
                </complexType>
            </element>

            <complexType name="MatlabNodeType">
                  <sequence>
                        <element maxOccurs="1" minOccurs="1" name="User"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="Host"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="SimPath"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="0"
name="MatlabPath" type="string"/>
                  </sequence>
            </complexType>

            <complexType name="SimNodeType">
                  <sequence>
                        <element maxOccurs="1" minOccurs="1" name="User"
type="string"/>
                        <element maxOccurs="1" minOccurs="1" name="Host"
type="string"/>
                  </sequence>
            </complexType>

            <complexType name="SimDataType">
                  <sequence>

                    <element maxOccurs="1" minOccurs="1" name="SrcArchive"
type="string"/>
                    <element maxOccurs="1" minOccurs="1" name="mFile"
type="string">
                        </element>

                  </sequence>
            </complexType>

            <complexType name="InstanceType">
                  <sequence>
                        <element maxOccurs="1" minOccurs="1" name="SimID"
type="long">
                        </element>
                        <element maxOccurs="1" minOccurs="1"
name="MatlabNode" type="tns:MatlabNodeType"/>
                  </sequence>
            </complexType>

            <complexType name="InstancesType">
                  <sequence>
                        <element maxOccurs="unbounded" minOccurs="1"
name="Instance" type="tns:InstanceType"/>
```

```
                    </sequence>
                </complexType>

                <element name="startMatlabSim">
                    <complexType>
                        <sequence>
                            <element maxOccurs="1" minOccurs="1"
name="SimIDPandas" type="long">
                            </element>
                            <element maxOccurs="1" minOccurs="1"
name="t_end" type="int">
                            </element>
                            <element maxOccurs="1" minOccurs="1"
name="Stepnr" type="int"/>
                        </sequence>
                    </complexType>
                </element>

                <element name="startMatlabSimResponse">
                    <complexType>
                        <sequence>
                            <element name="out" type="string"/>
                            <element maxOccurs="1" minOccurs="1"
name="SimID-Pandas" type="long"/>
                        </sequence>
                    </complexType>
                </element>

                <element name="stop">
                    <complexType>
                        <sequence>
                            <element maxOccurs="1" minOccurs="1"
name="SimID-Pandas" type="long"/>
                        </sequence>
                    </complexType>
                </element>

                <element name="stopResponse">
                    <complexType>
                        <sequence>
                            <element name="out" type="string"/>
                            <element maxOccurs="1" minOccurs="1"
name="SimID-Pandas" type="long"/>
                        </sequence>
                    </complexType>
                </element>

                <element name="Instances" type="tns:InstancesType"/>

            </schema>
        </types>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
         MESSAGE TYPE DEFINITION - Definition of the message types used as
         part of the port type defintions
         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
        <message name="Matlab-DispatcherRequestMessage">
            <part element="tns:Matlab-DispatcherRequest" name="payload"/>
        </message>
        <message name="Matlab-DispatcherResponseMessage">
            <part element="tns:Matlab-DispatcherResponse" name="payload"/>
        </message>
```

```
206         <message name="startMatlabSimRequest">
207             <part element="tns:startMatlabSim" name="parameters"/>
208         </message>
209         <message name="startMatlabSimResponse">
210             <part element="tns:startMatlabSimResponse" name="parameters"/>
211         </message>
212         <message name="stopRequest">
213             <part element="tns:stop" name="parameters"/>
214         </message>
215         <message name="stopResponse">
216             <part element="tns:stopResponse" name="parameters"/>
217         </message>
218         <message name="InstancesMsg">
219             <part element="tns:Instances" name="parameters"/>
220         </message>
221
222     <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
223          PORT TYPE DEFINITION - A port type groups a set of operations into
224          a logical service unit.
225          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
226     -->
227         <!-- portType implemented by the Matlab-Dispatcher BPEL process -->
228
229         <portType name="Matlab-Dispatcher">
230             <operation name="initiate">
231                 <input message="tns:Matlab-DispatcherRequestMessage"/>
232                 <output message="tns:Matlab-DispatcherResponseMessage"/>
233             </operation>
234             <operation name="startMatlabSim">
235                 <input message="tns:startMatlabSimRequest"/>
236                 <output message="tns:startMatlabSimResponse"/>
237             </operation>
238             <operation name="stop">
239                 <input message="tns:stopRequest"/>
240                 <output message="tns:stopResponse"/>
241             </operation>
242         </portType>
243
244         <!-- portType implemented by the requester of Matlab-Dispatcher BPEL
245     process
246              for asynchronous callback purposes
247              -->
248
249
250
251     <!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
252          PARTNER LINK TYPE DEFINITION
253              the Matlab-Dispatcher partnerLinkType binds the provider and
254              requester portType into an asynchronous conversation.
255          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
256     -->
257         <plnk:partnerLinkType name="Matlab-Dispatcher">
258             <plnk:role name="Matlab-DispatcherProvider" portType="tns:Matlab-
259     Dispatcher"/>
260         </plnk:partnerLinkType>
261         <binding name="Matlab-DispatcherBinding" type="tns:Matlab-Dispatcher">
262
263           <soap:binding style="document"
264     transport="http://schemas.xmlsoap.org/soap/http"/>
265           <operation name="initiate">
266                 <soap:operation
267     soapAction="http://wsi.simtech.de/processes/samples/Matlab-
268     Dispatcher/initiate"/>
```

```
            <input>
                  <soap:body use="literal"/>
            </input>
            <output>
                  <soap:body use="literal"/>
            </output>
      </operation>
      <operation name="startMatlabSim">
            <soap:operation
soapAction="http://wsi.simtech.de/processes/samples/Matlab-
Dispatcher/startMatlabSim"/>
            <input>
                  <soap:body use="literal"/>
            </input>
            <output>
                  <soap:body use="literal"/>
            </output>
      </operation>
      <operation name="stop">
            <soap:operation
soapAction="http://wsi.simtech.de/processes/samples/Matlab-
Dispatcher/stop"/>
            <input>
                  <soap:body use="literal"/>
            </input>
            <output>
                  <soap:body use="literal"/>
            </output>
      </operation>
    </binding>

    <service name="Matlab-DispatcherService">
      <port binding="tns:Matlab-DispatcherBinding" name="Matlab-
DispatcherPort">
            <soap:address
location="http://localhost:8080/ode/processes/Matlab-Dispatcher"/>
      </port>
    </service>

</definitions>
```

WSDL Matlab-Bone

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://wsi.simtech.de/processes/samples/Matlab-Bone"
xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
xmlns:wsdl1="http://wsi.simtech.de/extensions/matlab/"
xmlns:wsdl2="http://wsi.simtech.de/extensions/pmconn/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="Matlab-Bone"
targetNamespace="http://wsi.simtech.de/processes/samples/Matlab-Bone">

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     TYPE DEFINITION - List of services participating in this BPEL process
     The default output of the BPEL designer uses strings as input and
     output to the BPEL Process. But you can define or import any XML
     Schema type and us them as part of the message types.
```

```
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="WSI_PMConnector">
    <plnk:role name="WSI_PMConnector" portType="wsdl2:WSI_PMConnector"/>
  </plnk:partnerLinkType>
    <plnk:partnerLinkType name="WSI_Matlab">
    <plnk:role name="WSI_Matlab" portType="wsdl1:WSI_Matlab"/>
  </plnk:partnerLinkType>
    <plnk:partnerLinkType name="WSI_Matlab">
    <plnk:role name="WSI_Matlab" portType="wsdl1:WSI_Matlab"/>
  </plnk:partnerLinkType>
    <import location="WSI_PMConnector.wsdl"
namespace="http://wsi.simtech.de/extensions/pmconn/"/>
    <import location="WSI_Matlab.wsdl"
namespace="http://wsi.simtech.de/extensions/matlab/"/>
    <types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://wsi.simtech.de/processes/samples/Matlab-Bone">

            <element name="Matlab-BoneRequest">
                <complexType>
                    <sequence>
                        <element maxOccurs="1" minOccurs="1" name="SimID"
type="long"/>
                        <element maxOccurs="1" minOccurs="1"
name="FromSimID" type="long">
                        </element>
                        <element maxOccurs="1" minOccurs="1"
name="StartElement" type="int">
                        </element>
                        <element maxOccurs="1" minOccurs="1"
name="EndElement" type="int">
                        </element>
                        <element maxOccurs="1" minOccurs="1"
name="Gausscount" type="int">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="Stepnr"
type="int">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="t_end"
type="int">
                        </element>
                        <element maxOccurs="1" minOccurs="1"
name="mSimPath" type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="mFile"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="mUser"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="mHost"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="wUser"
type="string">
                        </element>
                        <element maxOccurs="1" minOccurs="1" name="wHost"
type="string"/>
                    </sequence>
                </complexType>
            </element>
```

```
            <element name="Matlab-BoneResponse">
                <complexType>
                    <sequence>
                        <element maxOccurs="1" minOccurs="1" name="result"
type="string"/>
                        <element maxOccurs="1" minOccurs="1" name="SimID"
type="long"/>
                        <element maxOccurs="1" minOccurs="1"
name="PandasSimID" type="long"/>
                    </sequence>
                </complexType>
            </element>

        </schema>
    </types>

<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     MESSAGE TYPE DEFINITION - Definition of the message types used as
     part of the port type defintions
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <message name="Matlab-BoneRequestMessage">
        <part element="tns:Matlab-BoneRequest" name="payload"/>
    </message>

    <message name="Matlab-BoneResponseMessage">
        <part element="tns:Matlab-BoneResponse" name="payload"/>
    </message>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PORT TYPE DEFINITION - A port type groups a set of operations into
     a logical service unit.
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <!-- portType implemented by the Matlab-Bone BPEL process -->
    <portType name="Matlab-Bone">
        <operation name="initiate">
            <input message="tns:Matlab-BoneRequestMessage"/>
        </operation>
    </portType>

    <!-- portType implemented by the requester of Matlab-Bone BPEL process
         for asynchronous callback purposes
         -->
    <portType name="Matlab-BoneCallback">
        <operation name="onResult">
            <input message="tns:Matlab-BoneResponseMessage"/>
        </operation>
    </portType>


<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
     PARTNER LINK TYPE DEFINITION
         the Matlab-Bone partnerLinkType binds the provider and
         requester portType into an asynchronous conversation.
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-->
    <plnk:partnerLinkType name="Matlab-Bone">
        <plnk:role name="Matlab-BoneProvider" portType="tns:Matlab-Bone"/>
        <plnk:role name="Matlab-BoneRequester" portType="tns:Matlab-
BoneCallback"/>
```

```
143       </plnk:partnerLinkType>
144       <binding name="Matlab-BoneBinding" type="tns:Matlab-Bone">
145         <soap:binding style="document"
146   transport="http://schemas.xmlsoap.org/soap/http"/>
147         <operation name="initiate">
148               <soap:operation
149   soapAction="http://wsi.simtech.de/processes/samples/Matlab-Bone/initiate"/>
150               <input>
151                     <soap:body use="literal"/>
152               </input>
153         </operation>
154       </binding>
155       <binding name="Matlab-BoneCallbackBinding" type="tns:Matlab-
156   BoneCallback">
157         <soap:binding style="document"
158   transport="http://schemas.xmlsoap.org/soap/http"/>
159         <operation name="onResult">
160               <soap:operation
161   soapAction="http://wsi.simtech.de/processes/samples/Matlab-Bone/onResult"/>
162               <input>
163                     <soap:body use="literal"/>
164               </input>
165         </operation>
166       </binding>
167       <service name="Matlab-BoneService">
168         <port binding="tns:Matlab-BoneBinding" name="Matlab-BonePort">
169               <soap:address
170   location="http://localhost:8080/ode/processes/Matlab-Bone"/>
171         </port>
172       </service>
173       <service name="Matlab-BoneCallbackService">
174         <port binding="tns:Matlab-BoneCallbackBinding" name="Matlab-
175   BoneCallbackPort">
176               <soap:address
177   location="http://localhost:8080/ode/processes/Matlab-BoneCallback"/>
178         </port>
179       </service>
180   </definitions>
```

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Raymond Dormien)