Institut für Parallele und Verteilte Systeme (IPVS)
Institut für Architektur von Anwendungssystemen (IAAS)
Institut für Technische und Numerische Mechanik (ITM)
Universität Stuttgart
D–70569 Stuttgart

Studienarbeit Nr. 2314

Ausführung einer Modellreduktion für Simulationen auf Basis der Workflow-Technologie

Simon Remppis

Studiengang:	Technische Kybnernetik
Prüfer:	PD Dr. rer. nat. Holger Schwarz (IPVS)
Betreuer:	Dipl.-Inf. Peter Reimann (IVPS) Dipl.-Math. Michael Reiter (IAAS) Dr.-Ing. Jörg Fehr (ITM)
begonnen am:	22. Dezember 2010
beendet am:	3. November 2011
CR-Klassifikation:	H.4.1, I.6.7, I.6.3

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Rechnergestützte Simulationen sind aus der modernen Wissenschaft nicht mehr wegzudenken, sie nehmen sowohl im industriellen als auch im akademischen Umfeld einen immer wichtigeren Stellenwert ein. Angesichts der stetigen Weiterentwicklung der Informationstechnik und der damit verbundenen Leistungssteigerung von Simulationssystemen ist ein Ende dieses Trends nicht absehbar (vgl. [HW10], [Glo11]).

Bei der Durchführung wissenschaftlicher Simulationen kommt eine Vielzahl unterschiedlicher Werkzeuge und Softwarelösungen zum Einsatz, die oftmals große Datenmengen verarbeiten müssen. Teilweise werden diese Tools gezielt für ein eng begrenztes Anwendungsszenario, also z.B. für die Simulation einer einzigen Klasse von dynamischen Systemen[1] oder gar eines einzigen Systems entwickelt. Als Beispiel sei Software genannt, die gezielt für die Wettervorhersage oder für die Verbrennungssimulation in einem bestimmtem Dampfkessel geschrieben wird. Andere Tools, beispielsweise das in [RRS+11] vorgestellte SIMPL-Framework, sind dagegen eher generischer Natur und übernehmen Aufgaben, die bei der Simulation ganz verschiedener Arten von Systemen auftreten können.

Wie auch in anderen Teilbereichen der Softwaretechnik ist die Wiederverwendbarkeit von Software eine häufige Anforderung an Simulationswerkzeuge, da sich dadurch der Aufwand späterer Neuentwicklungen erheblich senken lässt (vgl. [OPM+09]). Vor diesem Hintergrund ist es sinnvoll, komplexe Simulationsaufgaben in einzelne Teile aufzuspalten, welche möglichst von eher generischen und wiederverwendbaren Tools oder Softwarekomponenten erledigt werden können.

Im Bereich der Geschäftsanwendungen hat sich aus dieser Überlegung heraus das Prinzip der *Serviceorientierten Architektur (SOA)* entwickelt, bei der einzelne Aufgaben von unabhängigen Programmen beziehungsweise Diensten erledigt werden. Komplexere Arbeitsabläufe, man spricht hier von Geschäftsprozessen, können dann durch eine Kombination verschiedener Dienste abgearbeitet werden. Die Workflow-Technologie bietet Möglichkeiten, solche Geschäftsprozesse zu beschreiben, zu modellieren, automatisiert auszuführen und zu verwalten. Es liegt nahe, diese Vorgehensweise auch auf die Implementierung von Simulationsaufgaben zu übertragen.

[1] Der Begriff *System* ist hier im Sinne der Systemtheorie zu verstehen. Eine genauere Definition und Abgrenzung dieses Begriffs findet sich in Abschnitt 2.3.

1.1 Motivation

Die Betrachtung und Implementierung von wissenschaftlichen Simulationen als Workflow versprechen verschiedene Vorteile gegenüber monolithischen Simulationslösungen, die für einen einzigen Anwendungsfall entwickelt werden. So können neue Simulationsworkflows aus den Bestandteilen bereits implementierter Workflows zusammengesetzt werden, was den Entwicklungsaufwand für Simulationsaufgaben wesentlich reduziert.

Außerdem stehen vielfältige Werkzeuge zur Verfügung, die den Entwurf, die Ausführung und Verwaltung sowie die Analyse von Workflows vereinfachen. Da Simulationsaufgaben häufig lange Laufzeiten aufweisen und deshalb verschiedene Simulationsvorgänge parallel ausgeführt werden sollten, ist eine Verwaltung verschiedener Instanzen eines Workflows, wie sie von Workflow-Management-Systemen unterstützt wird, von großem Nutzen.

Die verschiedenen Teilprozesse in einer Serviceorientierten Architektur kommunizieren üblicherweise über netzwerktransparente Protokolle miteinander und können plattformunabhängig verwendet werden. Diese Eigenschaft bietet bei der Durchführung wissenschaflicher Simulationen große Vorteile, da bestimmte Simulationsprozesse oder Teile davon nicht auf gewöhnlichen Arbeitsplatzrechnern oder Servern, sondern auf spezialisierter Hardware und in heterogenen Netzen ausgeführt werden [Tay07].

Auf der anderen Seiten treten bei der Implementierung von wissenschaftlichen Simulationen als Workflow auch einige spezifische Probleme auf, für die Lösungen gefunden werden müssen. Für die Bewältigung einzelner Simulationsaufgaben wird häufig proprietäre Software eingesetzt, die nicht ohne weiteres über ein netzwerkfähiges Protokoll angesprochen und so als Dienst in einen Workflow integriert werden kann. Ein für diese Arbeit relevantes Beispiel für solche Software ist die Numerik- und Simulationsumgebung *Matlab*.

Eine Herausforderung besteht auch darin, dass Wissenschaftler, die Simulationsprozesse entwerfen, nicht unbedingt über viel Wissen und Erfahrung in der Softwareentwicklung verfügen [Tay07]. Die verwendeten Workflow-Tools und -Sprachen sollten also möglichst einfach zu erlernen und zu nutzen sein.

1.2 Aufgabe

In der technischen Mechanik werden Simulationen eingesetzt, um das dynamische Verhalten elastischer Mehrkörpersysteme zu untersuchen. In solchen Systemen sind die elastischen Verformungen der mechanischen Komponenten von entscheidender Bedeutung und müssen deshalb bei der Simulation berücksichtigt werden. Aufgrund der großen Zahl der Freiheitsgrade solcher Systeme kann die Dimension der aufgestellten mathematischen Modelle sehr groß werden, was lange Ausführungszeiten der Simulation zur Folge hat.

Durch eine Modellreduktion versucht man deshalb, die Komplexität der mathematischen Modelle zu verringern, um die Ausführungsgeschwindigkeit der nachfolgenden Simulation drastisch zu erhöhen. Im Rahmen dieser Arbeit soll eine Modellreduktion auf Basis der

Workflow-Technologie automatisiert ausgeführt werden, damit die oben genannten Vorteile dieser Technologie ausgenutzt werden können. Dazu wird der Vorgang der Reduktion in sinnvolle Workflow-Schritte zerlegt. Es werden verschiedene Verfahren untersucht, mit denen die einzelnen Workflow-Schritte – welche zum großen Teil mit der proprietären Software *Matlab* ausgeführt werden – als Webservice verfügbar gemacht und in den Workflow eingebunden werden können. Weiterhin wird darauf eingegangen, wie dem Workflow die Ausgangsdaten bereitgestellt und wie die reduzierten Modelldaten ausgewertet und weiterverarbeitet werden können.

1.3 Gliederung und Aufbau

Der Rest der Arbeit gliedert sich in folgende Kapitel:

Kapitel 2 – Grundlagen: Hier werden die Grundlagen dieser Arbeit beschrieben und wesentliche Technologien und Verfahren aus dem Bereich der Workflow-Technologie, der Simulationstechnik und der technischen Mechanik vorgestellt. Weiterhin werden wichtige Begriffe definiert und die in dieser Arbeit verwendete Software vorgestellt.

Kapitel 3 – Konzept eines Workflows zur Modellreduktion: Hier wird dargestellt, in welche Workflow-Schritte die Modellreduktion eines mechanischen Systems sinnvoll zerlegt werden kann.

Kapitel 4 – Implementierung des Workflows: stellt die konkrete Implementierung einer Modellreduktion als Workflow vor. Dabei wird speziell auf die Verwaltung der Arbeitsdaten, auf die verschiedenen Möglichkeiten der Anbindung von Matlab an den Workflow-Prozess sowie auf mögliche Erweiterungen des Workflows eingegangen.

Kapitel 5 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen.

2 Grundlagen

Um eine Modellreduktion als Workflow automatisiert auszuführen, ist der Einsatz verschiedener Konzepte und Technologien aus den Bereichen SOA, Webservices, Workflows und Simulationstechnik notwendig. Diese werden in den Abschnitten 2.1 bis 2.3 vorgestellt. Danach vermittelt der Abschnitt 2.4 eine Übersicht über die mechanischen und mathematischen Grundlagen der durchzuführenden Modellreduktion. Abschnitt 2.5 stellt im Anschluss die für diese Arbeit wichtige Software vor. Hier erfolgt insbesondere eine Einführung in das Softwarepaket Matlab, welches für den in dieser Arbeit entwickelten Modellreduktionsworkflow essentiell ist.

2.1 SOA und Webservices

Das Thema Serviceorientierte Architektur ist ein Schlagwort, das besonders im Bereich der Unternehmens-IT auf viel Interesse gestoßen ist. Bei der Wahl der Strategie zur Modernisierung von IT-Landschaften fällt die Entscheidung häufig zugunsten dieses Architekturmodells. Der Kerngedanke von SOA besteht darin, IT-Funktionalitäten in einzelne, lose gekoppelte Serviceblöcke oder *Dienste* aufzuspalten und damit komplexe Strukturen aufzulösen [FZ09].

Die einzelnen Dienste lassen sich beliebig kombinieren und für unterschiedliche Anwendungen wiederverwenden. Auf diese Weise kann man der Nachfrage nach schnell entwickelbaren und anpassbaren Anwendungen besser gerecht werden als mit großen, monolithischen und damit weniger flexiblen Anwendungen (vgl. [PH07]). Beispielsweise könnten in einem System zur Verkaufsabwicklung das Aufnehmen einer Person in eine Kundendatenbank, das Erstellen einer Rechnung und die Vorbereitung des Versandes durch einzelne Dienste erledigt werden. Ein komplexerer Prozess wie die Bearbeitung einer Bestellung wird dann mit geringem Aufwand durch die Kombination dieser Dienste erledigt. Dieses Zusammensetzen verschiedener Dienste zu einer Anwendung wird im SOA-Umfeld *Orchestrierung* genannt.

Der Begriff SOA beschreibt lediglich das Architekturmodell einer IT-Infrastruktur und bezieht sich nicht auf eine konkrete Technologie. Zwar werden an die Bestandteile einer SOA-Implementierung gewisse Ansprüche gestellt, auf die im Abschnitt 2.1.1 näher eingegangen wird. Welche Protokolle, Sprachen und Frameworks aber letztendlich für die konkrete Umsetzung verwendet werden, wird nicht vorgeschrieben und bleibt dem Entwickler überlassen. In der Praxis wird für die Kommunikation oft auf die Technologien und Protokolle des *Webservice-Stacks*, insbesondere auf SOAP und HTTP gesetzt [FZ09]. Da diese Technologien und Protokolle auch für diese Arbeit von Bedeutung sind, werden sie im Kapitel 2.1.2 näher vorgestellt. Alternativ könnten für die Realisierung einer SOA aber auch andere plattformunabhängige Kommunikationsprotokolle wie *Representational State Transfer (REST)* oder die Common *Object Request Broker Architecture (CORBA)* eingesetzt werden (vgl. [FZ09]).

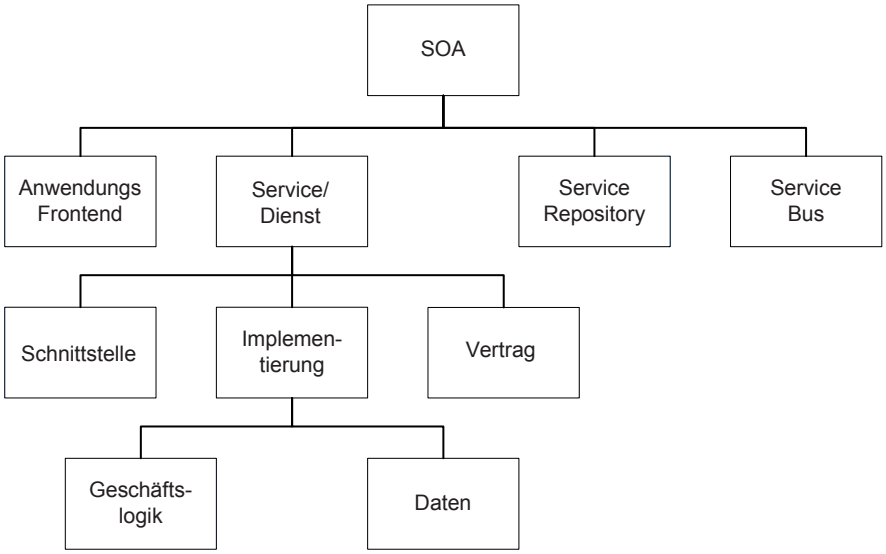

Abbildung 2.1: Bestandteile der SOA (vgl. [KBS05])

2.1.1 Bestandteile der SOA

Serviceorientierte Architekturen setzen sich aus verschiedenen, miteinander interagierenden Komponenten zusammen. Diese werden in Abbildung 2.1 hierarchisch dargestellt und in den folgenden Abschnitten näher besprochen.

Dienste

Zentrales Element einer SOA sind die Dienste bzw. Services. Dienste zeichnen sich im SOA-Kontext nach [PH07] durch folgende Eigenschaften aus

- Ein Dienst erledigt eine Aufgabe autonom und kann eigenständig genutzt werden. Er verwaltet seinen Zustand bis auf wenige Ausnahmen (Stichwort *WS-Resource Framework*) selbstständig.
- Er besitzt eine öffentliche Schnittstelle, die plattformunabhängig und netzwerktransparent genutzt werden kann.
- Ein Dienst kann genutzt werden, ohne seine Implementierung und seine innere Funktionsweise zu kennen. Zur Nutzung eines Dienstes ist es lediglich notwendig, seine Schnittstelle zu kennen.
- Ein Dienst kann dynamisch gebunden werden, d.h. bei der Erstellung einer Anwendung, die einen Dienst nutzt, braucht der Dienst nicht vorhanden zu sein. Es ist möglich, ihn erst bei der Ausführung zu lokalisieren und einzubinden.

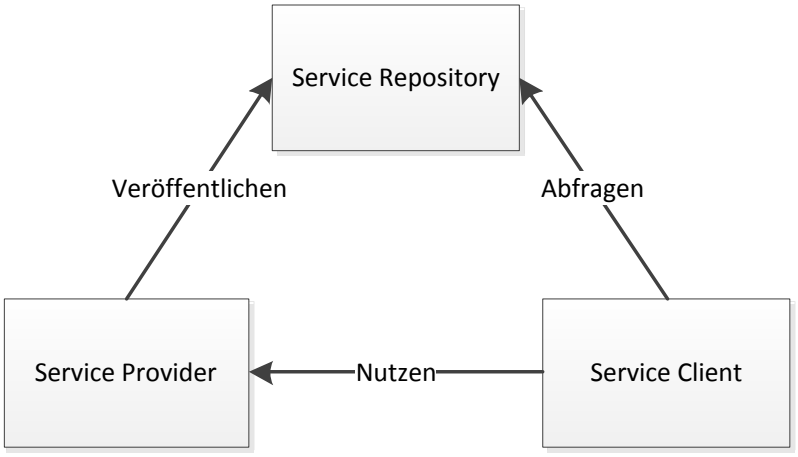

Abbildung 2.2: SOA-Dreieck (vgl. [PH07])

- Sämtliche Funktionalität in einer SOA wird als Dienst realisiert. Dies gilt sowohl für tatsächliche Geschäftsprozesse als auch Hilfsfunktionen und weiterer Funktionalität, die zum ordnungsgemäßen Ablauf der Programme notwendig ist.

Ergänzend zu oben genannten der Schnittstelle können die Eigenschaften eines Dienstes durch einen Service-Vertrag genauer spezifiziert werden. In ihm werden Funktionen und ggf. Beschränkungen der Nutzung beschrieben, an die sich Anbieter und Nutzer von Diensten halten müssen. Durch diese informelle Spezifikation wird eine reibungslose Verständigung und Leistungserbringung des Dienstes gewährleistet [FZ09].

Rollen

Nutzer von Diensten werden *Service Client* oder *Service Consumer* genannt, Anbieter von Diensten *Service Provider*. Weiterhin existieren in einer Serviceorientierten Architektur Verzeichnisse der vorhandenen Dienste und ihrer Eigenschaften. Diese *Service Repositories* oder *Service Broker* ermöglichen dem Client das automatische Auffinden passender Dienste anhand der vom Dienst zu erfüllenden Anforderungen. Die Zusammenhänge dieser drei Rollen werden im SOA-Dreieck verdeutlicht (Abbildung 2.2).

Service Bus

Der Service Bus stellt die Kommunikationsinfrastruktur dar, die die verschiedenen Komponenten der SOA miteinander verbindet. Er ist unter anderem dafür zuständig, eine Verbindung zwischen einem Service Client und einem Service Provider herzustellen, wenn ein entsprechender Dienst angefordert wird [Ley05]. Dabei ist der Service Bus dem Grundgedanke der SOA folgend nicht an eine bestimmte Technologie gebunden, sondern ermöglicht eine freie Wahl der IT-Plattform. Die Implementierung der angesprochenen Dienste kann auf unterschiedlichen Technologien basieren, sie wird nicht durch den Service-Bus festgelegt.

<table>
<tr><td rowspan="2">Anwendung</td><td colspan="2">SOAP, WSDL...</td></tr>
<tr><td>HTTP</td><td>HTTPS</td></tr>
<tr><td rowspan="2">Transport</td><td rowspan="2">TCP</td><td>SSL/TLS</td></tr>
<tr><td>TCP</td></tr>
<tr><td>Internet</td><td colspan="2">IP</td></tr>
<tr><td>Netzzugang</td><td colspan="2">...</td></tr>
</table>

Tabelle 2.1: Kernprotokolle von Webservices im OSI-Schichtenmodell (vgl. [FZ09])

Die wichtigsten Aufgaben des Service Bus sind nach [FZ09]:

- die Nachrichtenübermittlung
- die Datentransformation
- das Routing der Nachrichten

Darüber hinaus stellt der Service Bus auch technische Dienste bereit. Diese können zum Beispiel die Protokollierung, die Gewährleistung der Kommunikationssicherheit oder die Formatierung von Nachrichten umfassen.

2.1.2 Webservices

Das W3C[1] definiert Webservices als ein Mittel, mit dem Softwareanwendungen, welche auf verschiedenen Plattformen und Frameworks betrieben werden, miteinander kommunizieren können [MBF+04].

Dabei ist der Begriff Webservices mit einer Reihe von Technologien und Protokollen assoziiert, welche in der Regel kombiniert werden, um eine nachrichtenbasierte, netzwerktransparente und plattformübergreifende Kommunikation unterschiedlicher Anwendungen zu gewährleisten. Die Gesamtheit dieser Technologien und Protokolle wird auch als *Webservice-Stack* bezeichnet.

Trotz der Namensähnlichkeit dürfen Webservices nicht mit dem oft nur als *Web* bezeichneten *World Wide Web (WWW)* verwechselt werden. Zwar verwenden Webservices ähnlich wie das WWW üblicherweise (allerdings nicht zwingend) das *Hyper Text Transfer Protocol (HTTP)* oder das verschlüsselte *HTTPS* als Kommunikationsprotokoll. Ebenso sind *Unified Resource Identifiers (URIs)* von entscheidender Bedeutung für die Funktionsweise von Webservices. Im Gegensatz zu Websites oder Webanwendungen richten sich Webservices allerdings nicht an menschliche Benutzer und verfügen deshalb über kein Frontend. Sie sind vielmehr für die Kommunikation von Softwaresystemen gedacht (welche natürlich als Webanwendung realisiert sein können). Diese können über den Webservice-Stack Daten austauschen oder Funktionen auf entfernten Rechnern aufrufen.

[1] Das World Wide Web Consortium (`http://www.w3c.org`) ist die wichtigste Standardisierungsorganisation für das World Wide Web und verwaltet eine ganze Reihe von Protokollen die mit dem Web oder Webservices in Verbindung stehen, darunter HTML, SOAP und WSDL

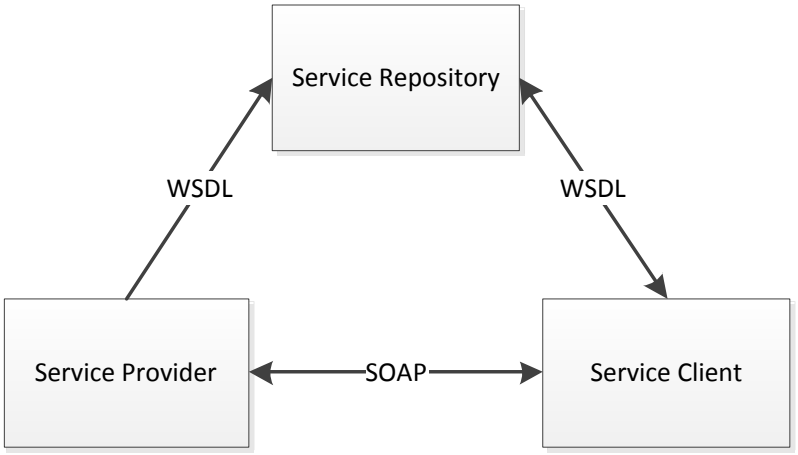

Abbildung 2.3: Einsatz von Webservices in einer SOA (vgl. [FZ09])

Da Webservices für die Verwendung durch Computerprogramme ausgelegt sind, werden für die Strukturierung der Nachrichten, die von und an Webservices übermittelt werden, maschienenlesbare Sprachen verwendet. So kommt für den Aufruf von Funktionen und den Austausch von Informationen zwischen Client und Server das XML-basierte Nachrichtenprotokoll *SOAP* zum Einsatz. Zur Beschreibung von Diensten und ihren Schnittstellen wird *WSDL*, eine andere XML-Sprache, eingesetzt. Beide Sprachen werden in den folgenden Abschnitten näher beschrieben. Zur besseren Übersicht über die verschiedenen Sprachen und Technologien, die mit Webservices assoziiert werden, sei auf die Tabelle 2.1 verwiesen, welche die wesentlichen Protokolle im OSI-Schichtenmodell[2] einordnet.

Webservices stellen die derzeit populärste Technologie dar, um Serviceorientierte Architekturen zu realisieren [PH07]. Gegenüber alternativen Kommunikationskonzepten wie CORBA oder *dem Distributed Component Object Model (DCOM)* von Microsoft haben Webservices den Vorteil, dass die genutzten Dienste bei der Entwicklung von Software nicht bekannt sein müssen, sondern zur Laufzeit dynamisch gebunden werden können [Wie04]. Webservices ermöglichen somit die in Serviceorienterten Architekturen geforderte lose Bindung der Komponenten. Außerdem werden die Spezifikationen der für Webservices wesentlichen Technologien und Protokolle von herstellerübergreifenden Konsortien wie dem W3C oder OASIS[3] erarbeitet und sind so nicht an bestimmte Plattformen gebunden. Abbildung 2.3 zeigt, wie die verschiedenen Webservice-Sprachen zur Realisierung der Kommunikation in einer SOA eingesetzt werden können.

[2]Das OSI-Schichtenmodell ordnet Netzwerkprotokolle hierarchisch in sieben Schichten ein. Es gibt einen Überblick darüber, wie verschiedene Kommunikationsprotkolle aufeinander aufbauen. Je höher ein Protokoll im Schichtenmodell angeordnet ist, desto weiter sind Hardwareeigenschaften abstrahiert und desto weniger übertragungstechnische Aufgaben wie Routing oder Fehlerkorrektur müssen von der entsprechenden Anwendung übernommen werden. Zum genaueren Verständnis sei auf Fachliteratur wie [Tan03] verwiesen.

[3]Die *Organization for the Advancement of Structured Information Standards (OASIS)*(`http://www.oasis-open.org/`) ist eine internationale, herstellerübergreifende Organisation, welche die Entwicklung und den Einsatz von Standards rund um die Themen E-Business und Webservices fördert.

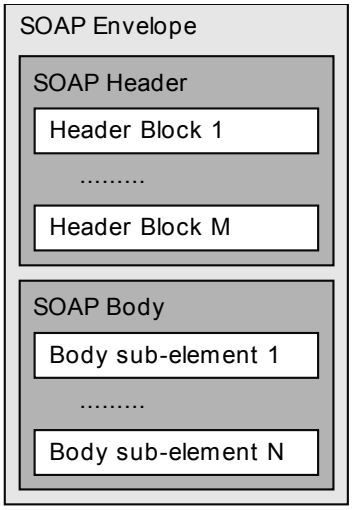

Abbildung 2.4: Aufbau von SOAP-Nachrichten (vgl. [GHMJJM07])

SOAP

SOAP ist ein leichtgewichtiges, nachrichtenbasiertes Protokoll, das für den Austausch strukturierter Informationen zwischen verschiedenen Softwaresystemen in einer dezentralen, verteilten Umgebung gedacht ist. Das Protokoll wird vom W3C unter [GHMJJM07] spezifiziert, die Informationen in diesem Abschnitt basieren, wo nicht anders angegeben, auf dieser Spezifikation.

SOAP darf nicht nur als Nachrichtenformat verstanden werden, sondern stellt vielmehr eine Spezifikation dar, die verschiedene Aspekte der Kommunikation zwischen Sender und Empfänger einer Nachricht über beliebig viele Zwischenstationen beschreibt. Die Zwischenstationen können dabei in die Kommunikation eingreifen und die verschickten Nachrichten beispielsweise protokollieren oder auch transformieren.

Das SOAP-Protokoll deckt folgende Bereiche ab [ND07]:

- Das XML-Format der verschickten Nachrichten.
- Die Bindung an ein Übertragungsprotokoll. Wie bereits erwähnt kommt hier meistens HTTP oder HTTPS zum Einsatz, es existieren aber auch Bindings für andere Protokolle wie *Java Message Service (JMS)* [AEJ+10] oder *Simple Mail Transfer Protocol (SMTP)*.
- Regeln für den Versand, die Weiterleitung und die Verarbeitung der Nachrichten durch die oben genannten Zwischenstationen.

SOAP-Nachrichten bestehen wie in Abbildung 2.4 dargestellt aus einem als *Envelope* bezeichneten Wurzelelement, welches einen *Header* und einen *Body* enthält. Header und Body wiederum können jeweils beliebig viele, XML-formatierte Informationsblöcke enthalten.

Der Body einer SOAP-Nachricht enthält die eigentlichen Nutzdaten, oft auch *payload* genannt. Liegen diese in einem XML-Dialekt vor, können sie direkt in die Body-Blöcke geschrieben werden. Hier profitiert das SOAP Protokoll vom Namespace-Konzept des XML-Formats, das eine

semantisch saubere Einbettung eines XML-Dialekts in einen anderen ermöglicht. Binärdaten müssen dagegen erst *Base64*-kodiert[4] oder der Nachricht als *SOAP-Attachment* angehängt werden. Darüber hinaus dient der Body einer Nachricht auch dazu, im Fehlerfall andere Kommunikationsteilnehmer über den Fehler und dessen Ursache zu informieren. Dazu sieht die Spezifikation ein fault-Element im Body vor, das einen Fehlercode und eine Fehlerbeschreibung enthalten muss.

Der Header ist optional und wird üblicherweise dafür verwendet, Metainformationen zu verpacken, die für die richtige Weiterverarbeitung oder -leitung der Nachricht notwendig sind. Headerblöcke werden von allen Stationen einer Nachricht auf dem Weg vom Sender zum letztendlichen Empfänger verarbeitet und danach normalerweise entfernt. Mit den XML-Attributen `role`, `mustUnderstand` und `relay` kann das gewünschte Verhalten der Kommunikationsteilnehmer genauer spezifiziert werden und unter anderem auch festgelegt werden, dass bestimmte Header-Blöcke an nachfolgende Stationen weitergeleitet werden. Typische Inhalte des Headers sind zum Beispiel Transaktionsnummern oder Sicherheitszertifikate. Für einige häufige Fälle wie die Zuordnung der Nachrichten zu vorangegangenen Anfragen oder die Signierung und Verschlüsselung der Nachrichten existieren weiterreichende standardisierte Spezifikationen, die bestimmte Header-Blöcke vorsehen. Dazu zählen beispielsweise WS-Adressing[5] und WS-Security[6] [FZ09].

Zur Veranschaulichung zeigt Listing 2.1 beispielhaft zwei SOAP-Nachrichten, die zwischen einem Client und einem Server ausgetauscht werden könnten. In der ersten fragt der Client anhand einer Matrikelnummer nach dem Namen des Studenten, in der zweiten teilt der Server ihm diesen Namen mit.

WSDL

Um die lose Kopplung und die späte Bindung von Webservices zu ermöglichen, müssen Dienste bzw. ihre Schnittstellen formell beschrieben sein, so dass die Client-Software zur Laufzeit anhand dieser Beschreibungen die passenden Services aussuchen und mit den bereitgestellten Schnittstellen kommunizieren kann. Zu diesem Zweck wurde die *Webservice Description Language (WSDL)* entwickelt. Will ein Client einen Webservice nutzen, kann er die jeweilige WSDL-Datei beim Service-Verzeichnis abrufen und erhält so alle für die Nutzung des Dienstes notwendigen Informationen. Wird der verwendete Dienst schon bei der Entwicklung fest mit der Client-Software verknüpft, kann die WSDL-Datei auch vom Entwickler des Clients manuell bezogen werden. Es existieren Werkzeuge welche dann anhand dieser WSDL-Beschreibung Vorlagen für die zu verschickenden SOAP-Nachrichten und die lokalen Anknüpfungspunkte erzeugt.

[4] Base64 ist ein verbreitetes Verfahren um Binärdaten in Textformaten wie XML unterzubringen. Bei der Base64-Kodierung werden 3 Bytes in 4 Zeichen aus dem ASCII-Zeichensatz umgewandelt. Erwartungsgemäß nimmt dabei der Speicherplatzverbrauch der Daten zu [Jos06].

[5] `http://www.w3.org/TR/2006/REC-ws-addr-core-20060509`

[6] `http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf`

Listing 2.1 Beispiel einer Anfrage und der entsprechenden Antwort im SOAP-Nachrichtenformat

```
<!-- Anforderung der Noten eines Studenten von einem (imaginären) Informationssystem -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <soap:Header>
              <t:TransactionId xmlns:t="http://example.com/transactions"
                  soap:mustUnderstand="1">5</t:TransactionId>
       </soap:Header>
       <soap:Body>
              <s:GetStudentName xmlns:s="http://example.com/studentInformation.xsd"">
                     <s:MatrNo>1234567</s:MatrNo>
              </s:>
       </soap:Body>
</soap:Envelope>
<!-- Antwort des Systems -->
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
       <soap:Header>
              <t:TransactionId xmlns:t="http://example.com/transactions"
                  soap:mustUnderstand="1">5</t:TransactionId>
       </soap:Header>
       <soap:Body>
              <s:StudentName xmlns:s="http://example.com/studentInformation.xsd">
                     Max Mustermann
              </s:StudentName>
       </soap:Body>
</soap:Envelope>
```

Wie in Abbildung 2.5 dargestellt, besteht eine WSDL-Datei im Allgemeinen aus zwei Teilen. Der abstrakte Teil beschreibt, was ein Dienst an Funktionalität bietet. Er beschreibt, welche Schnittstellen der Dienst enthält (`wsdl:portType`), welche Operationen die jeweiligen Schnittstellen unterstützen (`wsdl:operation`) und welche Nachrichten die Operationen erwarten bzw ausgeben (`wsdl:input`,`wsdl:output`). Außerdem wird hier definiert, wie diese Ein- und Ausgabenachrichten der Operationen aufgebaut sind (`wsdl:message`). Im abstrakten Teil der WSDL-Datei können auch Datentypen definiert (`wsdl:types`) werden, die in den Nachrichten verwendet werden. Hier bedient sich die WSDL bei der vom W3C empfohlenen Schemasprache *XML Schema* ([WF04]).

Der konkrete Teil einer WSDL-Datei enthält Informationen darüber, wie, also über welches Protokoll (`wsdl:binding`) mit dem Webservice kommuniziert werden kann und wo, also unter welcher Adresse er erreichbar ist (`wsdl:port`). Natürlich können für eine Schnittstelle auch mehrere Bindings und Ports existieren, wenn der Dienst verschiedene Kommunikationsprotokolle anbietet oder unter verschiedenen Adressen erreichbar ist.

Codebeispiel 2.2 zeigt eine WSDL-Datei, die alle oben genannten Bereiche enthält. Sie beschreibt einen Service, der die im vorherigen Abschnitt abgebildeten SOAP-Beispielcodes (Listing 2.1) zur Kommunikation verwendet. Zum genaueren Verständnis der WSDL sei auf die vom W3C unter [CCMW01] veröffentlichten Spezifikationen verwiesen.

Listing 2.2 Beispiel einer Servicebeschreibung in WSDL

```
<definitions name="studentInformation"
         targetNamespace="http://example.com/studentInformation.wsdl"
         xmlns:tns="http://example.com/studentInformation.wsdl"
         xmlns:xsd1="http://example.com/studentInformation.xsd"
         xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
         xmlns="http://schemas.xmlsoap.org/wsdl/"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <types>
       <xsd:schema targetNamespace="http://example.com/studentInformation.xsd">
                      <xsd:simpleType name="MatrNoType">
                         <xsd:restriction base="xsd:integer">
                               <xsd:length value="7"/>
                         </xsd:restriction>
                      </xsd:simpleType>
                      <xsd:element name="GetStudentName">
                        <xsd:complexType>
                               <xsd:sequence>
                                     <xsd:element name="MatrNo" type="xsd1:MatrNoType"/>
                               </xsd:sequence>
                        </xsd:complexType>
                      </xsd:element>
       </xsd:schema>
    </types>
    <message name="GetStudentNameInput">
        <part name="body" element="xsd1:GetStudentName"/>
    </message>
    <message name="GetStudentNameOutput">
        <part name="body" element="xsd:string"/>
    </message>
    <portType name="studentInformationPortType">
        <operation name="GetStudentName">
           <input message="tns:GetStudentNameInput"/>
           <output message="tns:GetStudentNameOutput"/>
        </operation>
    </portType>
    <binding name="studentInformationSoapBinding" type="tns:studentInformationPortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetStudentName">
           <soap:operation soapAction="http://example.com/GetStudentName"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
    <service name="studentInformationService">
        <documentation>Information about Students</documentation>
        <port name="studentInformationPort" binding="tns:studentInformationSoapBinding">
           <soap:address location="http://example.com/studentInformation"/>
        </port>
    </service>
 </definitions>
```

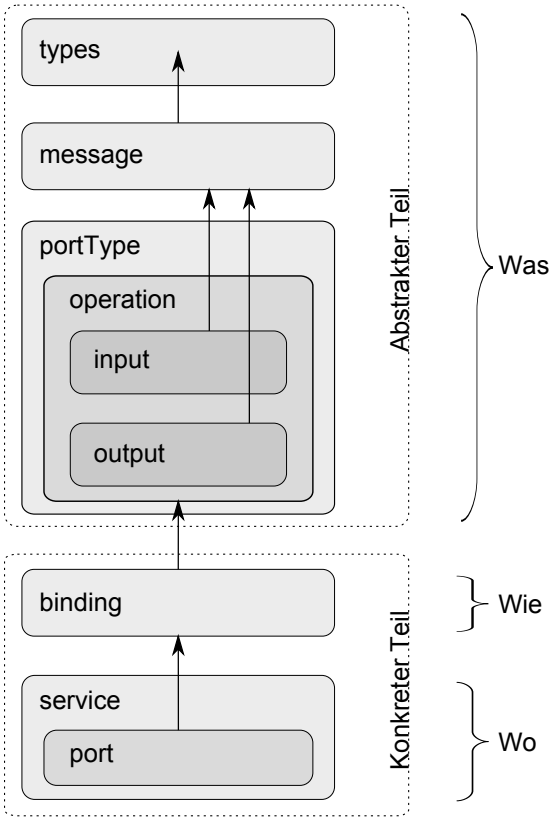

Abbildung 2.5: Aufbau einer WSDL-Datei (vgl. [Cri07])

2.2 Workflow-Technologie

Auch die Workflow-Technologie stammt ursprünglich aus dem Bereich der Geschäftsanwendungen. Der Workflow-Begriff ist dabei eng mit dem Begriff des *Prozesses* bzw. des *Geschäftsprozesses* verbunden. Ein Prozess stellt einen Ablauf von Bearbeitungsschritten dar. Hierbei ist aber – z.B. im Gegensatz zu einem Projekt – die prinzipielle Wiederholbarkeit des Ablaufes charakteristisch [FG08]. Das *Prozessmodell* beschreibt die Struktur eines Prozesses und legt fest, wie der Prozess abläuft, d.h. es enthält alle Aktivitäten eines Prozesses und definiert die möglichen Ausführungspfade zwischen diesen Aktivitäten [LR00]. Wird der Prozess dann tatsächlich ausgeführt, dient das Prozessmodell als Vorlage bzw. als Ablaufplan. Ein konkreter Durchlauf durch das Modell wird, vergleichbar mit der aus der objektorientierten Programmierung bekannten Terminologie, *Prozessinstanz* genannt. So kann ein Prozessmodell z.B. beschreiben, welche Schritte notwendig sind, um einen Studenten an einer Hochschule zu immatrikulieren. Bewirbt sich nun der Studienanwärter Max Mustermann auf einen Studienplatz im Studiengang Maschienenbau, stellt der konkrete Vorgang der Immatrikulation dieses Studenten eine Prozessinstanz dar.

Zu beachten ist, dass alle Instanzen eines Prozessmodells zwar dem selben Muster folgen, jedoch nicht exakt identisch sein müssen. Zum Beispiel wird die Immatrikulation eines ausländischen

Studienanwärters oder eines Anwärters auf ein Zweitstudium einem anderen Pfad durch das Prozessmodell folgen, da hier besondere Bearbeitungsschritte notwendig sind.

Prozesse müssen nicht zwingend von Computern ausgeführt werden. Prinzipiell kann jeder wiederholbare Ablauf, der einem bestimmtem Muster folgt, in einem Prozessmodell beschrieben werden, sei es die Heuernte in der Landwirtschaft, die Wahl eines Studierendenvertreters oder auch das Backen einer Pizza. Wenn Teile des Prozesses allerdings mit Unterstützung von Computern (teil-)automatisiert ausgeführt werden sollen, müssen diese in einer für den Computer lesbaren Form modelliert werden: Der Computer benötigt Informationen darüber, wie die Prozessschritte in der Praxis ausgeführt werden, wer für die Ausführung zuständig ist und welche Ressourcen dazu benötigt werden. Das Ergebnis dieser Modellierung wird *Workflowmodell* genannt [LR00]. Analog zu Prozessinstanzen bezeichnen *Workflowinstanzen* den konkreten Ablauf eines Workflows. Die einzelnen Aktivitäten, die in einem Workflow miteinander verknüpft werden, können als *Workflowschritte* bezeichnet werden.

Die genaue Abgrenzung der Begriffe „Prozess“ bzw. „Prozessmodell“ auf der einen und „Workflow“ bzw. „Workflowmodell“ auf der anderen Seite wird allerdings uneinheitlich gehandhabt und in der Praxis oft vernachlässigt. Auch in dieser Arbeit werden die Begriffe nicht strikt unterschieden, wenn sich die genaue Bedeutung des Begriffes aus dem Kontext ergibt.

Im Sprachgebrauch wird dagegen oft zwischen *Business Workflows* und *Scientific Workflows* unterschieden. Erstere modellieren Geschäftsabläufe, letztere werden zur (teil-)automatisierten Ausführung wissenschaftlicher Aufgaben verwendet. Das Workflow-Konzept unterscheidet jedoch nicht generell zwischen den verschiedenen Bereichen und auch die im nächsten Kapitel vorgestellten Werkzeuge und Sprachen können oft auf generische Weise in verschiedenen Anwendungsbereich eingesetzt werden. Abschnitt 2.3.2 geht näher auf die Besonderheiten von Simulationsworkflows ein.

Grundsätzlich muss nicht jeder Schritt eines Workflows automatisierbar sein. Allerdings legt die formalisierte Beschreibung des Ablaufes, wie sie von Workflowsprachen und -Systemen ermöglicht wird, nahe, dass der konkrete Ablauf einer Workflowinstanz von Computersystemen unterstützt wird und einzelne Schritte soweit wie möglich automatisiert werden. Aufgaben, die nicht automatisiert werden können aber für den Ablauf des Workflows notwendig sind können mit *Human Task*-Systemen an den Workflow angebunden werden (vgl. Abschnitt 4.4.3).

2.2.1 Workflow-Management-Systeme

Als *Workflow-Management-Systeme (WfMS)* werden allgemein Computeranwendungen bezeichnet, die dazu dienen, Workflows zu modellieren, zu verwalten, auszuführen und die Ausführung zu analysieren. Eine präzise Abgrenzung dieser Systeme ist schwierig, da Workflow-Management-Systeme in vielfältiger Weise implementiert werden können, um den unterschiedlichen Anforderungen verschiedenster Anwender gerecht zu werden. Allerdings zeigen alle solche Systeme gemeinsame Charakteristika und einen ähnlichen Grundaufbau. Um die Interoperabilität unter-

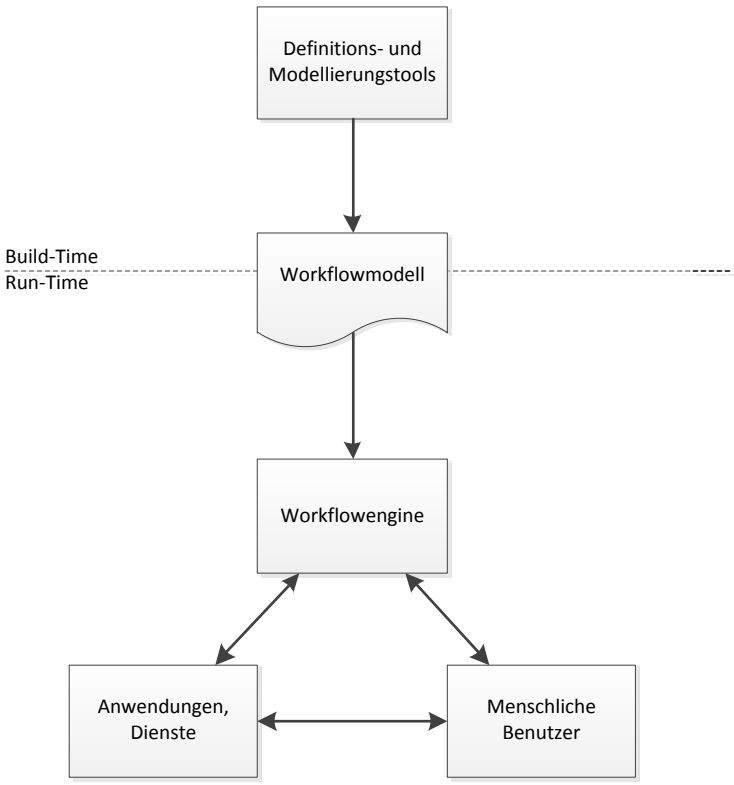

Abbildung 2.6: Teile eines Workflow-Management-Systems (vgl. [Hol95])

schiedlicher Systeme zu ermöglichen, definiert die *Workflow Management Coalition (WfMC)*[7] in [Hol95] ein Referenzmodell, welches ein Workflow-Management-System konzeptionell beschreibt, ohne auf Details wie die verwendete Kommunikationsinfrastruktur einzugehen.

Für den Entwurf dieses Referenzmodells wurden die drei Hauptaufgaben von WfMSen wie folgt identifiziert:

- Unterstützung beim Entwurf eines Workflows durch ein *Modelliertool*. Am Ende der Modellierung steht ein maschienenlesbares Abbild des Prozesses in einer Workflowsprache.
- Die Erzeugung und Ablaufkontrolle von Workflowinstanzen aus dem Workflowmodell. Der Teil des Systems, der diese Aufgabe übernimmt, wird auch *Workflowengine* genannt.
- Die Interaktion mit den einzelnen Computeranwendungen oder Personen, die für die Bearbeitung der einzelnen Prozessschritte zuständig sind.

Die Teile des Systems, die diese Aufgaben übernehmen, werden entsprechend ihrem Ausführungszeitpunkt in Buildtime- und Runtime-Funktionen unterteilt (siehe Abbildung 2.6). Das in

[7] Die Workflow Management Coalition (`http://www.wfmc.org/`) ist ein Verbund von Nutzern, Entwicklern, Beratern und Wissenschaftlern im Bereich des Workflow-Managements. Die WfMC entwickelt und verwaltet verschiedene Standards um die Interoperabilität von Workflow-Management-System zu fördern

einer maschienenlesbaren Workflowsprache (man spricht hier auch von einer *Ausführungssprache*) geschriebene Workflowmodell stellt das Bindeglied zwischen diesen Bereichen dar. Auf die dabei zur Verfügung stehenden Sprachen wird im folgenden Abschnitt näher eingegangen.

In realen Anwendungsszenarien sind die einzelnen Anwendungen, die zur Bearbeitung von Workflowschritten aufgerufen werden, oft nicht auf einem Rechner konzentriert, sondern über eine heterogene IT-Landschaft verstreut. Dies gilt auch für Scientific Workflows, da Teile von wissenschaftlichen Simulationen oft auf hochspezialisierten Hardware- und Softwareplattformen ausgeführt werden. Um die Kommunikation der Workflowengine mit den einzelnen Anwendungen zu gewährleisten, bietet sich hier eine inhärent netzwerkfähige SOA als Runtime-Umgebung für Workflow-Management-Systeme an.

2.2.2 Workflow-Sprachen

Für die Abbildung von Workflows in einem Workflowmodell existieren eine ganze Reihe von Sprachen. Teilweise setzen diese unterschiedliche Akzente in ihrem syntaktischem und semantischen Aufbau, andere Sprachen zielen aber auch in eine ähnliche Richtung und wurden von konkurrierenden Herstellern oder Herstellerverbänden parallel entwickelt. Der Bereich der Workflow-Sprachen ist allerdings einer gewissen Konsolidierung unterworfen. In letzter Zeit hat sich hier besonders die *Business Process Execution Language (BPEL)* herauskristallisiert. Daneben existiert mit der *Business Process Model and Notation (BPMN)* eine verbreitete grafische Spezifikationssprache, um Geschäftsprozesse und Workflows darzustellen. Da BPMN in Kapitel 3 für den konzeptionellen Entwurf eines Workflows zur Modellreduktion verwendet wird und BPEL bei der in Kapitel 4 beschriebenen Implementierung zum Einsatz kommt, wird in den folgen Abschnitten detaillierter auf diese Sprachen eingegangen.

BPMN

BPMN ist bis zur Version 1.2 eine rein grafische Sprache zur Darstellung von Workflows. Sie wurde mit dem Ziel entwickelt, für alle an der Definition, Implementierung und Analyse von Business-Workflows beteiligten Personen, also auch für Betriebswirtschaftler, einfach verständlich zu sein. Die Sprache will so die Lücke zwischen dem Design und der technischen Umsetzung von Workflows schließen [OMG11].

BPMN kann als eine standardisierte Sprache für Flowcharts bzw. Flussdiagramme verstanden werden. Um den Anforderungen nach einer möglichst einfach zu verwendenden Workflowsprache zu genügen, ist der Sprachumfang relativ gering. Das Workflowdiagramm 2.7 zeigt die im Folgenden vorgestellten Elemente im Kontext eines anschaulichen Beispiels, dem Bestellen einer Pizza bei einem Pizzalieferdienst.

- Ein *Event* markiert irgendein Ereignis, auf das im Workflow reagiert wird. Events treten in Form von *Start-* (1) und *Endevents* (2) zu Beginn und am Ende des Workflows auf, können aber auch zwischen verschiedenen Workflowschritten ausgelöst werden. In letzterem Fall spricht man von *Intermediate Events* (3). Events können zum Beispiel durch Eingang einer Nachricht oder durch Ablauf einer Zeitspanne ausgelöst werden.

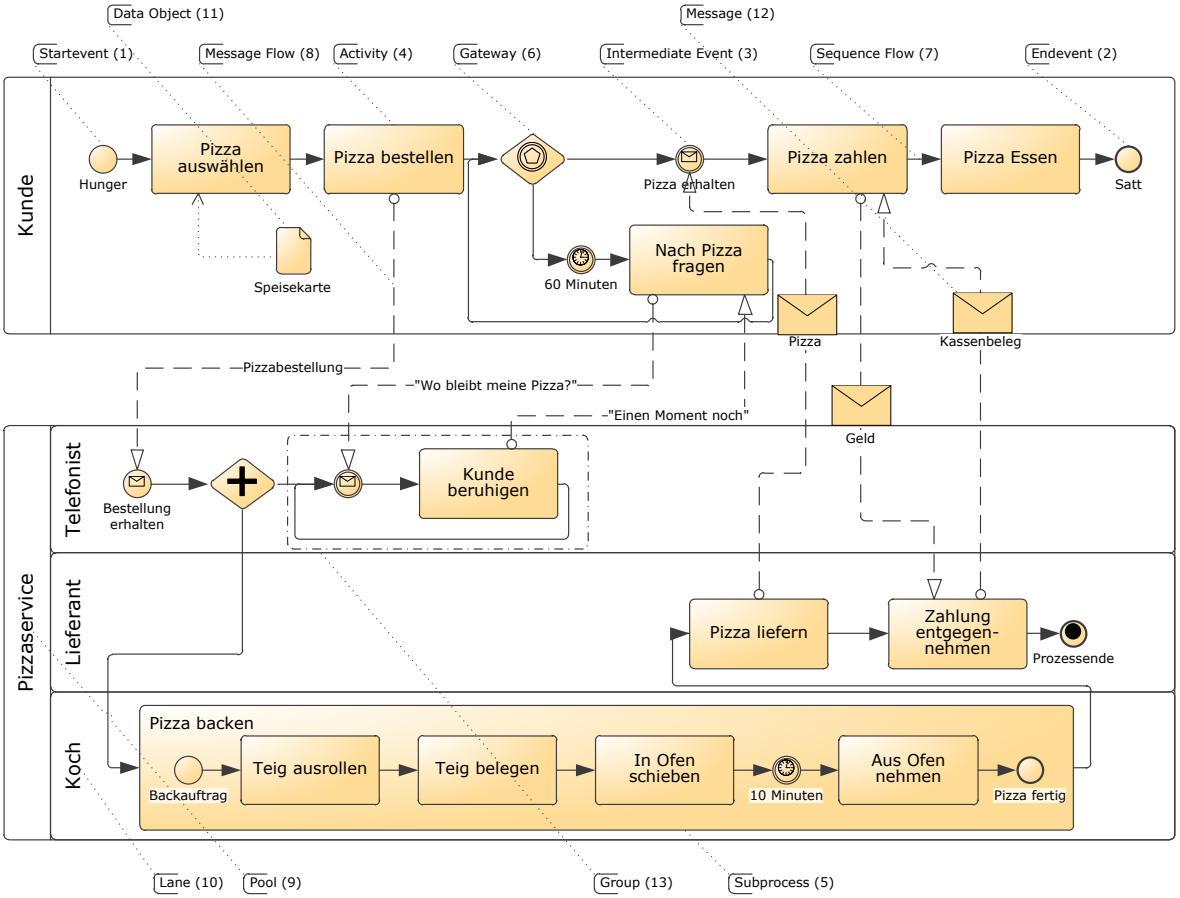

Abbildung 2.7: Beispiel eines Workflows zur Bestellung von Pizza (vgl. [OMG10])

- Eine *Activity* (4) ist eine Aufgabe, die zur Abarbeitung des Workflows zu erledigen ist. Activities können in Form von atomaren Aufgaben oder auch in Form von weiter unterteilbaren *Subprocesses* (5) auftreten.
- *Gateways* (6) stellen die Kontrollstrukturen im Ablauf des Workflows dar. Gateways können je nach Ausprägung dafür genutzt werden, Abläufe zu teilen, zusammenzuführen oder in Abhängigkeit einer Bedingung auf verschiedene Pfade zu führen.
- Der Ablauf des Workflows folgt dem *Sequenceflow* (7). Die Sequenzflusspfeile geben an, welche Aktivitäten in welcher Reihenfolge ausgeführt werden.
- Der *Messageflow* (8) wird verwendet, um zu zeigen, dass Informationen zwischen verschiedenen Workflowteilnehmern ausgetauscht werden.
- Ein *Pool* (9) repräsentiert einen Workflowteilnehmer oder eine Gruppe von Teilnehmern. Workflowteilnehmer können zum Beispiel Sachbearbeiter sein, die bestimmte Aktivitäten bearbeiten. Pools können weitere Symbole enthalten und dienen in diesem Fall als Container, um die verschiedenen Aufgaben oder Prozesse zusammenzufassen, die einem Teilnehmer zugeordnet sind. Leere Pools werden als *Black Boxes* verstanden. Sie

repräsentieren Workflowteilnehmer, die zwar gewisse Aufgaben übernehmen, es wird allerdings nicht weiter spezifiziert, wie sie diese Aufgaben erledigen. Stellt ein Pool eine Gruppe von Workflowteilnehmern dar, kann er weiter in *Lanes* (10) unterteilt werden, welche die individuellen Teilprozesse der einzelnen Workflowteilnehmer enthalten.

- *Data Objects* (11) kennzeichnen Daten, die in einer Aktivität verwendet werden können
- *Messages* (12) markieren Informationen, die zwischen verschiedenen Workflowteilnehmern ausgetauscht werden. Ihr Fluss wird durch die Messageflow-Pfeile gekennzeichnet.
- *Groups* (13) dienen zur visuellen Zusammenfassung unterschiedlicher Workflowelemente. Sie haben keinen Einfluss auf den Ablauf des Workflows, können aber beispielsweise dazu verwendet werden, logisch zusammengehörende Aktivitäten zu gruppieren.

Bis zur Version 1.2 spezifiziert die BPMN nur das Aussehen und die Bedeutung der Symbole, mit denen Workflows grafisch modelliert werden können. Sie eignet sich daher sehr gut zur Beschreibung eines Geschäftsprozesses auf fachlicher Ebene, auch durch technisch weniger geschulte Betriebswirtschaftsspezialisten. Sie ist allerdings nicht maschienenlesbar und kann deshalb nicht als von einer Workflowengine interpretierbare Ausführungssprache verwendet werden. Es sind zwar Werkzeuge verfügbar, um einen Workflow von der BPMN in eine Ausführungssprache wie BPEL zu übersetzen, doch erfordert der übersetzte Code in der Praxis eigentlich immer manuelle Nachbearbeitung und ist nicht ohne technisches Wissen anwendbar. Der Grund dafür liegt darin, dass die Ausführungssemantik vieler Kontrukte in BPMN 1.2 nicht genau genug spezifiziert ist, um den so tranformierten Workflow ohne Anpassungen ausführen zu können (vgl. [OADH06]).

Die im März 2011 verabschiedete Version 2.0 der BPMN-Spezifikation standardisiert ein XML-basiertes Format, in dem BPMN-Diagramme gespeichert werden können. Außerdem wurden viele für die Ausführung relevanten Sprachelemente wesentlich genauer definiert. Damit ist der gegenseitige Datenaustausch zwischen verschiedenen Modellierungtools sowie die Ausführung durch Workflowengines prinzipiell möglich (vgl. [OMG11]). Für diese Arbeit spielt die XML-Representation der BPMN aber keine Rolle, für weitere Informationen sei auf die Spezifikation [OMG11] und das Dokument „BPMN by Example“ [OMG10] verwiesen.

BPEL

Die *Business Process Execution Language (BPEL)* ist eine Ausführungssprache für Workflows. Die Sprache ging als Nachfolger aus der blockorientierten Sprache XLANG[8] von Microsoft und aus der, auf gerichteten Graphen basierenden, *Web Service Flow Language (WSFL)*[9] von IBM hervor [Hav05]. BPEL vereint die Ansätze dieser beiden Workflowsprachen. Seit 2003 ist *WS-BPEL* (bis 2004 BPEL4WS) ein von OASIS unter [JE07] spezifizierter, herstellerunabhängiger Standard, der sich wachsender Beliebtheit erfreut und mittlerweile von einer Vielzahl von Workflow-Management-Systemen auf unterschiedlichen Plattformen unterstützt wird (vgl.

[8] http://msdn.microsoft.com/de-de/library/aa577463(v=bts.70).aspx
[9] http://xml.coverpages.org/wsfl.html

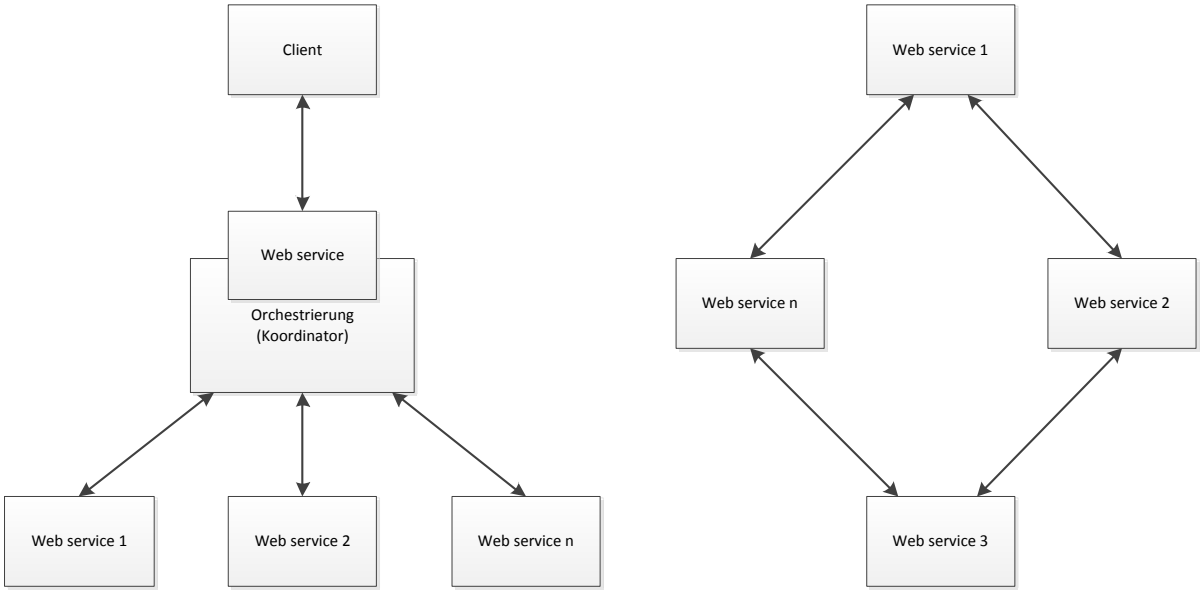

Abbildung 2.8: „Orchestrierung“ (links) und „Choreographie“ (rechts) (vgl. [JMS06])

[Gha08]). Die Informationen in diesem Kapitel basieren, wenn nicht anders angegeben, auf dieser Spezifikation. BPEL ist die gebräuchliche Kurzform für die Sprache.

Das „WS-“ im offiziellen Namen des Standards ordnet BPEL den Standards des Webservice-Stacks zu und assoziiert die Sprache mit dem Webservice- und SOA-Umfeld. Auch aus technischer Sicht ist BPEL stark mit den Webservice-Technologien und -Protokollen verknüpft. So werden die einzelnen Schritte eines in BPEL modellierten Workflows in der Regel durch Webservices bearbeitet, gleichzeitig stellt der Workflow selbst einen Webservice bereit, über den er gesteuert werden kann.

Orchestrierung von Webservices

Es gibt zwei prinzipiell verschiedene Methoden, um Webservices zu einem umfangreicheren Prozess zusammenzusetzen (siehe Abbildung 2.8) . Bei der *Orchestrierung* steuert ein zentraler Koordinator die Ausführung der einzelnen Webservices. Die Dienste selbst brauchen in einem solchen Szenario kein Wissen über den Prozessablauf besitzen, sie müssen nicht einmal wissen, dass sie für die Bearbeitung eines übergeordneten Prozesses verwendet werden. Diese Methode eignet sich also, um der Forderung nach einer losen Kopplung der Dienste in einer Service-orientierten Architektur gerecht zu werden. Darüber hinaus ermöglicht diese Methode die einfache Verwaltung und Steuerung der Workflowinstanzen, sowie die flexible Behandlung von Fehlersituationen, da die komplette Prozesslogik von einer zentralen Komponente kontrolliert wird.

Die *Choreographie* bezeichnet dagegen eine verteilte Methode der Prozesssteuerung. Die choreographierten Dienste kommunizieren, einem Peer-to-Peer-Ansatz folgend, direkt miteinander.

Das Wissen über die Prozesslogik liegt nicht zentral vor, sondern verteilt sich über die Dienste. Folglich benötigt jeder Dienst Informationen über seine Rolle im Gesamtprozess und muss die im Prozessfluss benachbarten Webservices kennen.

In BPEL wird außerdem zwischen ausführbaren und abstrakten Workflows unterschieden. Ausführbare Workflows enthalten alle, zum Steuern des Workflows notwendigen, operativen Instruktionen. Sie können direkt von einer Workflowengine interpretiert werden, die dann als Koordinator fungiert und die verschiedenen, orchestrierten Webservices aufruft.

Die seltener verwendeten, abstrakten Workflows sind nicht ausführbar, da sie nicht alle zur Ausführung benötigen Informationen enthalten. Sie können aber als Vorlage für die Definition ausführbarer Prozesse dienen oder dazu verwendet werden, das von außen zu beobachtende Verhalten des Prozesses formal zu beschreiben. Details, die für den jeweiligen Anwendungszweck eines abstrakten Prozesses keine Rolle spielen, werden bei der Modellierung weggelassen. Für diese Arbeit spielen abstrakte Workflows keine Rolle, für weitere Informationen zu diesem Thema sei auf Kapitel 13 der BPEL-Spezifikation verwiesen.

Die Webservices, mit denen der Workflowkoordinator bei der Ausführung des Workflows kommuniziert, werden in BPEL-Workflows über sogenannte *PartnerLinks* definiert. Dabei werden keine konkreten Dienste hinterlegt, sondern über PartnerLinkTypes die Rollen der Kommunikationsteilnehmer sowie die von ihnen bereitgestellten Schnittstellen beschrieben. Dafür wird der abstrakte Teil der WSDL-Beschreibungen mit den wsdl:portTypes verwendet (siehe Abschnitt 2.1.2). Erst bei der Instanziierung eines Workflows werden die PartnerLinkTypes dann mit konkreten Service Providern verknüpft. Durch diese späte Bindung müssen die tatsächlich verwendeten Webservices bei der Erstellung eines Workflows nicht bekannt sein und können später einfach ausgetauscht werden.

Spracheigenschaften

BPEL ist bis zu einem gewissen Grad mit herkömmlichen Programmiersprachen vergleichbar. So sieht die Sprache Variablen, Zuweisungen und Kontrollstrukturen wie Schleifen und Verzweigungen vor. Ein umfangreiches Konzept zum Behandeln von Fehlern und Ausnahmesituationen ist ebenfalls vorhanden. Die wichtigsten Sprachfunktionen von BPEL beziehen sich allerdings auf den Aufruf von Webservices, mit denen synchron oder asynchron kommuniziert werden kann. BPEL erlaubt es nach [JMS06]:

- die Logik von Geschäftsprozessen durch eine Kombination von Webservices zu beschreiben,
- weniger komplexe Prozesse und Dienste zu umfangreiche Geschäftsprozesse zu kombinieren,
- lang laufende Dienste synchron oder asynchron aufzurufen und auf ihre Antworten zu reagieren,
- mehrere Dienste parallel oder sequentiell aufzurufen,
- eingehende Nachrichten von der Workflowengine an den entsprechenden Prozess weiterleiten zu lassen,

- verschiedene Aktivitäten durch die parallel durchzuführen und die Prozessflüsse kontrolliert zusammenzuführen,
- komplexe Geschäftsprozesse in verschachtelte Geltungsbereiche, sogenannte *Scopes* aufzuteilen,
- auf zeit- oder nachrichtenbezogene Ereignisse und Ausnahmesituationen durch *Handler* zu reagieren.

Aktivitäten

Der Zugriff auf Daten und die Ansteuerung von Webservices wird in BPEL-Workflows von sogenannten *Aktivitäten* ausgeführt. BPEL unterscheidet zwischen atomaren und strukturierten Aktivitäten. Atomare Aktivitäten führen einzelne Aktionen aus, während strukturierte Aktivitäten weitere Aktivitätsblöcke enthalten und den Kontrollfluss zwischen diesen Aktivitäten definieren.

Zu den strukturieren Aktivitäten zählen aus klassischen Programmiersprachen bekannte Konstrukte wie `if`, `while`, `forEach` oder `repeatUntil`. Daneben existieren mit `sequence` und `flow` zwei Elemente zur Steuerung der Ausführungsreihenfolge der untergeordneten Aktivitäten. Die in einer `sequence` enthaltenen Aktivitäten werden nacheinander ausgeführt, während das `flow`-Element die Steuerung des Kontrollflusses durch gerichtete Graphen erlaubt. Es kann auch dazu verwendet werden, Aktivitäten parallel auszuführen.

Zu den atomaren Aktivitäten zählen neben den `invoke`, `receive` und `reply`-Aktivitäten zur synchronen oder asynchronen Kommunikation mit externen Webservices u.A. auch die `assign`-Aktivität zum Zuweisen von Daten an eine Variable. Zur Typisierung von Variablen werden die aus XML-Schema bekannten XML-Datentypen verwendet. Der Zugriff auf strukturierte Daten erfolgt mithilfe der vom W3C standardisierten Abfragesprache *XPath*[10].

Scopes

Der Geltungbereich von Variablen, PartnerLinks und Handlern wird in BPEL durch eine Hierarchie von `scope`-Elementen gesteuert. Der äußerste Scope ist der Prozess selbst. Untergeordnete Scopes können wie Aktivitäten in den Prozess eingefügt werden und begrenzen den Geltungsbereich der darin definierten Variablen, PartnerLinks und Handler.

[10] `http://www.w3.org/TR/xpath/`

Handler

Eine zentrale Eigenschaft von BPEL ist die Möglichkeit, sogenannte Handler zu definieren, um mit beliebigen Aktivitäten auf verschiedene Ereignisse reagieren zu können. BPEL unterscheidet dabei zwischen *Event Handlern*, *Fault Handlern*, *Compensation Handlern* und *Termination Handlern*.

Event Handler können entweder durch eingehende Nachrichten oder durch zeitbasierte Ereignisse, also dem Erreichen bestimmter Zeitpunkte oder dem Ablaufen vordefinierter Zeitspannen ausgelöst werden. Die Ausführung von Event Handlern erfolgt parallel zum restlichen Prozess.

Fault und Compensation Handler sind die Grundbausteine des Fehlerbehandlungssystems von BPEL und dienen dazu, auf Ausnahmesituationen zu reagieren, die bei der Ausführung des Workflows auftreten. Ausnahmesitationen können in einem Workflow entweder durch eine Fehlerbenachrichtigung eines aufgerufenen Webservices, durch Ausnahmen bei der Workflowausführung (z.B. Division durch 0) oder manuell durch die `throw`-Aktivität ausgelöst werden. Im Fall eines Fehlers wird der Ablauf des im jeweiligen Scope enthaltenen Teilworkflows unterbrochen und der Fault Handler des Scopes aufgerufen. Ist im entsprechenden Scope kein Fault-Handler definiert oder löst der Fault-Handler selbst wieder eine Ausnahmesituation aus, wird der Fehler an den übergeordneten Scope weitergereicht. Das Fehlerbehandlungssystem ist also in gewisser Weise mit dem try-catch-Mechanismus verbreiteter Programmiersprachen vergleichbar.

Manchmal müssen im Fall eines Fehlers bereits vollständig ausgeführte Teilworkflows wieder rückgängig gemacht werden, um den Workflows in einen definierten Zustand zu versetzen. In BPEL können dazu die Compensation Handler der jeweiligen Scopes verwendet werden. Diese werden im Fehlerfall üblicherweise aus einem Fault Handler heraus durch die `compensate`-Aktivität gestartet und dienen zur Kompensierung der bereits erledigten Aufgaben eines Scopes oder einer ganzen Hierarchie von Scopes.

Termination Handlers schließlich werden automatisch beim unerwarteten Beenden der Ausführung eines Scopes ausgeführt. Im Gegensatz zu Fault-Handlern können sie selbst keinen weiteren Fehler auslösen und den Fehler nicht an übergeordnete Scopes weitergeben, sondern dienen lediglich zur Durchführung abschließender Maßnahmen im jeweiligen Scope.

Correlation

Eine Workflowengine ist grundsätzlich in der Lage, mehrere parallel laufende Instanzen eines Workflows zu verwalten. Da BPEL-Workflows während ihrer Laufzeit auf verschiedene eingehende Nachrichten reagieren können, muss die Workflowengine einen Mechanismus implementieren, um diese Nachrichten der richtigen Workflowinstanz zuzuordnen. BPEL sieht zu diesem Zweck *Correlation Sets* vor. Correlation Sets bestehen aus einer oder mehreren Nachrichteneigenschaften, die die Nachricht eindeutig einer Konversation und damit einer Workflowinstanz zuordnen. Beispiele für solche Eigenschaften sind mitgeschickte Bestell- oder Rechnungsnummern. BPEL Aktivitäten, die den Empfang von Nachrichten erlauben, können mit einem oder mehreren Correlation Sets verknüpft werden. Zur Run-Time untersucht die

Workflowengine dann eingehende Nachrichten auf die mit diesen Correlation Sets verknüpften Eigenschaften und leitet die Nachricht an die entsprechende Instanz weiter.

Beispiel

Listing 2.3 zeigt einen einfachen Workflow, der eine Zeichenkette vom aufrufenden Client entgegennimmt, dieser das Wort „Hallo“ voranstellt und das Ergebnis an den Client zurücksendet. Hingewiesen sie auf die Definition des PartnerLinks in Zeile 15ff und die XPath-Anweisung `concat` zum Zusammensetzen der Zeichenketten auf Zeile 40. Die WSDL-Datei mit der exakten Definition der für den Client sichtbaren Schnittstelle ist aus Platzgründen nicht abgedruckt, sie wird mit der `import`-Anweisung in Zeile 10ff eingebunden.

Auffällig ist, dass die Definition dieser einfachen Aufgabe, für die keine Orchestrierung externer Webservices notwendig ist, trotzdem eine vergleichsweise große Menge an Code benötigt. Die vollständig manuelle Implementierung realer BPEL-Workflows wäre daher eine relativ komplexe, zeitintensive Aufgabe. Aus diesem Grund ist die Verwendung von grafischen Modelliertools für die Entwicklung umfangreicher BPEL-Workflows fast unerlässlich (vgl. [Gha08]). In dieser Arbeit wurde für die Entwicklung von BPEL-Workflows der Eclipse-BPEL-Designer[11] verwendet.

2.2.3 Web Service Interface für Simulationsanswendungen

Innerhalb einer, auf Webservices aufbauenden, Serviceorientierten Architektur können Prozesse direkt als BPEL-Workflows modelliert und ausgeführt werden. In der Praxis werden in einen Prozess aber oft auch Anwendungen eingebunden, die keine geeignete Webservice-Schnittstelle zur Verfügung stellen. Gerade in Scientific Workflows wird in der Praxis oft auf sogenannte *Legacy-Anwendungen* zurückgegriffen, also auf Software, die nicht explizit für die Verwendung in einer SOA entwickelt wurde. Ein für diese Arbeit besonders relevantes Beispiel einer Legacy-Anwendung ist die Simulations- und Numerikumgebung Matlab (siehe Abschnitt 2.5.1), die für die Ausführung des Modellreduktionsworkflows benötigt wird. Um diese Anwendungen in einen, mit BPEL modellierten Workflow einzubinden, muss ihre Funktionalität zunächst als Webservice bereitgestellt werden.

In [Rut09] wird das *Generic Web Service Interface (WSI)* vorgestellt, das dazu dient, Webservices für Legacy-Simulationsanwendungen zu erstellen. Da sich die Schnittstellen unterschiedlicher Simulationsanwendungen stark unterscheiden, ist die Kernfunktionalität des WSI generisch gehalten, die konkrete Anpassung an einzelne Simulationsanwendungen erfolgt über Plugins. Zu den Kernfunktionen zählt unter anderem die Verwaltung von Simulationsinstanzen. Das WSI ermöglicht die mehrfache, parallele Ausführung einer Simulationsanwendung und sorgt dabei für die strikte Trennung der Arbeitsdaten, die den jeweiligen Instanzen zugeordnet sind. Dazu legt das WSI für jede Simulationsinstanz ein Verzeichnis im Dateisystem an und führt Simulationsanwendungen immer im Kontext dieses Verzeichnisses aus.

[11] http://www.eclipse.org/bpel/

Listing 2.3 Beispiel eines einfachen BPEL-Workflows

```
<process name="HelloWorld2"
    targetNamespace="http://ode/bpel/unit-test"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:tns="http://ode/bpel/unit-test"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:test="http://ode/bpel/unit-test.wsdl"
    queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath2.0"
    expressionLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath2.0">

  <import location="HelloWorld2.wsdl"
     namespace="http://ode/bpel/unit-test.wsdl"
     importType="http://schemas.xmlsoap.org/wsdl/" />

   <partnerLinks>
      <partnerLink name="helloPartnerLink"
         partnerLinkType="test:HelloPartnerLinkType"
         myRole="me" />
   </partnerLinks>

   <variables>
     <variable name="myVar" messageType="test:HelloMessage"/>
     <variable name="tmpVar" type="xsd:string"/>
   </variables>

   <sequence>
       <receive
          name="start"
          partnerLink="helloPartnerLink"
          portType="test:HelloPortType"
          operation="hello"
          variable="myVar"
          createInstance="yes"/>

      <assign name="assign1">
          <copy>
              <from variable="myVar" part="TestPart"/>
              <to variable="tmpVar"/>
          </copy>
          <copy>
              <from>concat('Hallo ',$tmpVar)</from>
              <to variable="myVar" part="TestPart"/>
          </copy>
      </assign>
       <reply name="end"
             partnerLink="helloPartnerLink"
             portType="test:HelloPortType"
             operation="hello"
             variable="myVar"/>
   </sequence>
</process>
```

Das WSI unterstützt sowohl die synchrone als auch die asynchrone Ausführung von Simulationsanwendungen. Bei der synchronen Ausführung wird die Verbindung zum Service Client, also zum Beispiel zur Workflowengine während der Ausführung der Simulationsanwendung aufrecht erhalten. Bei der asynchronen Ausführung wird der Transportkanal zum Client geschlossen, nachdem die Anwendung gestartet wurde. Nach dem Abschluss der Simulationsaufgabe schickt das Web Service Interface dann über eine neue Verbindung eine Antwort an den Client. Diese Vorgehensweise eignet sich besonders für lang laufende Simulationsaufgaben, da so Zeitüberschreitungen in den zugrunde liegenden Übertragungsprotokollen (insbesondere HTTP) vermieden werden können.

Außerdem unterscheidet das WSI zwischen statischen und interaktiven Simulationen. Kennzeichnend für die Klassifizierung ist die Art und Weise, wie Einfluss auf den Simulationsablauf genommen wird. Statische Simulationen werden vom Web Service Interface lediglich gestartet und laufen dann ohne weitere Interaktion. Der Simulationsablauf kann lediglich durch die vorherige Konfiguration beeinflusst werden. Interaktive Simulationen lassen sich dagegen auch während des Simulationsablaufs durch Webservice-Aufrufe beeinflussen. Zum Beispiel können im Fall der interaktiven Verwendung von Matlab verschiedene Rechenaufträge sukzessive an eine einmal gestartete Matlab-Instanz geschickt werden.

Interoperabilität mit proprietären Legacy-Anwendungen

Wenn der Quellcode einer Simulationsanwendung zur Verfügung steht und die Lizenz Änderungen daran zulässt, kann die Webservice-Fähigkeit im Nachhinein ergänzt werden. In vielen realen Fällen ist dies aber nicht der Fall. Dies gilt unter anderem auch für Matlab, das unter einer proprietären Lizenz steht.

Das WSI unterstützt für solche Fälle die Verwendung komplett unveränderter Anwendungen. Für die Kommunikation mit der Legacy-Anwendung stehen dabei abhängig von der verwendeten Applikation nach [Rut09] unterschiedliche Kanäle zur Verfügung:

Dateiaustausch: Das Web Service Interface stellt die Dateien bereit, mit denen die Anwendung arbeiten soll. Danach wird die Simulationsanwendung gestartet und schreibt die Ausgabedaten wiederum in Dateien. Nach Abschluss der Bearbeitung kann das Web Service Interface die erzeugten Dateien auswerten.

Standard-Ein-/Ausgabe: Das Web Service Interface startet die Simulationsanwendung und nutzt dann die Standard-Ein-/Ausgabe des Betriebssystems zur weiteren Kommunikation mit der Anwendung.

Mechanismen zur Interprozesskommunikation: Abhängig vom verwendeten Betriebssystem existieren eine Reihe von Frameworks und Werkzeugen für die *Interprozesskommunikation (IPC)*. Als Beispiel seien das *Component Object Model (COM)* von Microsoft oder das plattformunabhängige *D-Bus* genannt. Wenn die Simulationsanwendung die Kommunikation über ein solches Framework unterstützt, kann das Web Service Interface über ein geeignetes Plugin auf diesem Weg mit der Anwendung kommunizieren.

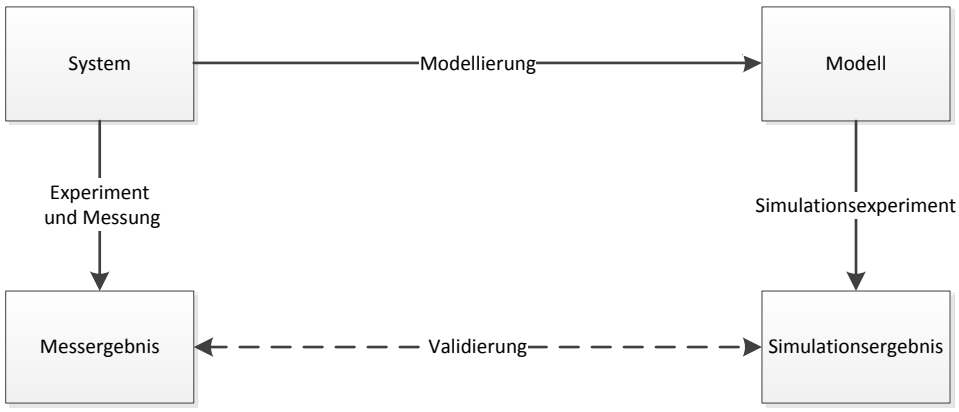

Abbildung 2.9: Prinzip der Modellierung und Simulation von Systemen

2.3 Simulationstechnik

Der Begriff der *Simulation* wird in VDI[12]-Richtlinie 3633 wie folgt definiert:

> „Simulation ist das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen die auf die Wirklichkeit übertragbar sind."

Genauer gesagt wird unter einer Simulation in der Regel die Nachbildung des Verhaltens eines beliebigen Systems verstanden. Das Vorhandensein eines geeigneten, experimentierfähigen Modells des Systems ist die Voraussetzung für die Durchführung einer Simulation. Das Erzeugen dieses Modells wird als *Modellierung* bezeichnet, das Nachstellen des interessierenden Verhaltens am Modell als *Simulationsexperiment* (siehe Abbildung 2.9).

Der Begriff *System* ist hier im Sinne der Systemtheorie zu verstehen. Systeme sind demnach strukturelle oder funktionale Gruppen von Elementen, die in einer Weise aufeinander bezogen sind, dass sie als ein sich von ihrer Umgebung abgrenzendes Ganzes betrachtet werden können. In der Simulationstechnik interessiert besonders die Klasse der dynamischen Systemen. Diese zeichnen sich durch eine zeitliche Änderung ihrer wichtigsten Kenngrößen aus. Dabei wird zwischen Eingangsgrößen, die auf das System einwirken und die zeitlichen Veränderungen innerhalb des Systems verursachen, und Ausgangsgrößen unterschieden, die das Verhalten des Systems als Reaktion auf die Eingangsgrößen beschreiben [Lun08]. Abhängig vom jeweiligen Wissenschaftsbereich können die unterschiedlichsten technischen oder natürlichen Einheiten wie Geräte, Anlagen, Organismen, Wirtschaftsräume oder auch soziale Gruppen als Systeme aufgefasst werden.

Mit *Modellen* sind hier Abbildung von Systemen gemeint. Diese dürfen nicht mit den weiter oben beschriebenen Workflowmodellen verwechselt werden. Zur klaren Unterscheidung werden Workflowmodelle im Rahmen dieser Arbeit immer auch als solche bezeichnet.

[12]Verein Deutscher Ingenieure (http://www.vdi.de/)

Ganz allgemein kann jede Nachbildung eines Systems als Modell für eine Simulation verwendet werden, welche unter den interessierenden Rand- und Anfangsbedingungen ein dem System vergleichbares Verhalten zeigt. Dies schließt zum Beispiel auch Crashtest-Dummies oder Miniaturmodelle von Flugzeugen für Windkanalsimulationen mit ein. Für diese Arbeit werden jedoch lediglich mathematisch-logische Abbildungen als Modell betrachtet, welche verwendet werden können, um das Verhalten des Systems am Rechner nachzubilden. Bei der rechnergestützten Simulation auf Basis dieser Modelle spricht man auch von *Computersimulationen*.

Die stetige Weiterentwicklung im Bereich der Informationstechnik hat die Einsatzbereiche der Computersimulationen immer weiter vergrößert und so praktisch alle Bereiche der empirischen Wissenschaften nachhaltig verändert. Die Beispiele für den Einsatz von Simulationstechnik in der Wissenschaft, Forschung und Entwicklung sind dementsprechend breit gefächert. Für diese Arbeit besonders relevant ist die Simulation der in einem mechanischen Bauteil auftretenden Kräfte und Spannungen mittels der Finite Elemente Methode (FEM), auf die in Abschnitt 2.4.2 näher eingegangen wird. Simulationen werden aber zum Beispiel auch dafür eingesetzt, das Flugverhalten von Flugzeugen, die Temperatur- und Druckverhältnisse in einem Verbrennungsmotor oder die biochemischen Vorgänge in einem lebenden Organismus genauer zu untersuchen. Auch Märkte, Wirtschaftssysteme und sogar soziale Zusammenhänge werden durch den Einsatz von Computersimulationen erforscht.

Die Gründe für die Durchführung von Simulationsexperimenten anstelle realer Experimente sind vielfältig. Simulationen kommen beispielsweise zum Einsatz (vgl. [BZBP09]),

- wenn eine Untersuchung am realen System technisch nicht oder nur mit unvertretbarem Aufwand realisierbar wäre. Die Baustatik eines Hochhauses oder die Evakuierung eines vollbesetzten Fußballstadions können nicht mit vertretbarem Aufwand an einem echten Objekt getestet werden.
- wenn die Untersuchung am realen System gefährlich oder ethisch nicht vertretbar wäre. Man denke hier an Simulationen der Ausbreitung von Krankheitserregern oder Untersuchungen des Flugverhaltens von Passagierflugzeugen. Auch Kernwaffensimulationen, die reale Kernwaffentests seit der Verfügbarkeit entsprechender Computersysteme weitgehend abgelöst haben, fallen in diese Kategorie.
- wenn das reale System noch nicht existiert. Technische Entwicklungen, deren Verhalten eine nicht-triviale Dynamik aufweisen, werden erst am Computer entworfen und ihr Verhalten anschließend simuliert, bevor ein realer Prototyp entworfen wird. Auf diese Weise wird der Entwicklungsprozess beschleunigt und die Entwicklungskosten reduziert. Diese Vorgehensweise ist zum Beispiel in der KFZ-Entwicklung verbreitet.
- wenn sich interessante Größen am realen System nicht oder nur schwer messen lassen. Beispiele hierfür sind mechanische Spannungen im Inneren von Bauteilen, die Verhältnisse im Inneren von Himmelskörpern oder auch das Verhalten von Teilchen auf subatomarer Ebene.
- wenn das interessierende Verhalten in der Realität zu schnell oder zu langsam abläuft. Die Entstehung von Galaxien lässt sich mit Teleskopen zwar in Momentaufnahmen beobachten, die zeitlichen Vorgänge können aber nur mit Simulationen untersucht werden.

Die Abläufe in elektrischen Schaltkreisen sind dagegen teilweise zu schnell für exakte Messungen. Ein weiteres populäres Beispiel für die Bedeutung der Simulationszeit ist die Wettervorhersage. Die Möglichkeit, Wettervorgänge schneller als in Echtzeit zu simulieren, erlaubt die rechtzeitige Vorhersage des Wetters.

- wenn die exakte Reproduzierbarkeit eines Experiments notwendig ist, das nur schwer unter den gleichen Bedingungen wiederholt werden kann. Unter Einsatz von Simulationstechnik lassen sich mehrere „virtuelle Experimente“ am selben Modell durchführen und dabei ganz gezielt nur bestimmte Parameter verändern, um deren Einfluss auf das Systemverhalten zu untersuchen.

Modellierung

Voraussetzung für ein brauchbares Simulationsergebnis ist ein Modell, das unter den interessierenden Bedingungen ein dem realen System vergleichbares Verhalten aufweist. Gibt das Modell die Wirklichkeit nur unzureichend wieder, lassen sich durch die Simulation nur ungenaue oder schlichtweg falsche Ergebnisse erzielen. Es gilt das Prinzip: Man kann nicht genauer simulieren, als man modellieren kann.

Die Modellierung selbst – in der Regel ein komplexer Vorgang – ist Gegenstand der jeweiligen Fachdisziplinen und soll hier nicht näher erörtert werden. Bei den hier betrachteten Computersimulationen steht am Ende der Modellierung aber immer ein mathematisches Modell, das in der Lage ist, die dynamischen Vorgänge des betrachteten Systems quantitativ zu beschreiben. Mathematik und Informatik stellen hierzu eine ganze Reihe an Beschreibungsmitteln bzw. Instrumentarien bereit [BZBP09]:

- algebraische Gleichungen oder Ungleichungen wie geometrische Zwangsbedingungen und bekannte oder vermutete Zusammenhänge ($E = mc^2$).
- Systeme gewöhnlicher Differentialgleichungen, also Differentialgleichungen mit nur einer unabhängigen Variablen, z.B. der Zeit.
- Systeme partieller Differentialgleichungen, also Differentialgleichungen mit mehreren, voneinander unabhängigen Variablen, wie dem Ort und der Zeit.
- Automaten und Zustandsübergangsdiagramme oder Graphen zur Modellierung ereignisdiskreter Systeme.
- Wahrscheinlichkeitsverteilungen, um zufällige Größen wie Ankunftsereignisse in einer Warteschlange zu beschreiben.
- regelbasierte Systeme oder Fuzzy-Systeme.
- neuronale Netze zur Modellierung von Lernprozessen
- ...

Es sei darauf hingewiesen, dass kein Modell das reale System exakt nachbilden kann. Für eine Bewertung der Aussagekraft eines Simulationsergebnisses ist daher eine *Validierung* des Modells unerlässlich. Dazu wird das Simulationsergebnis mit Beobachtungen am realen System verglichen (vgl. Abbildung 2.9) und bei zu großen Abweichungen das verwendete Modell angepasst. Doch auch hier ist Vorsicht geboten: Das Übereinstimmen des Modellverhaltens mit dem Verhaltens des realen Systems in einer beobachteten Situation lässt nicht auf die generelle Korrektheit des Modells schließen. Die Richtigkeit eines Simulationsergebnisses lässt sich also nicht beweisen.

Simulationspipeline

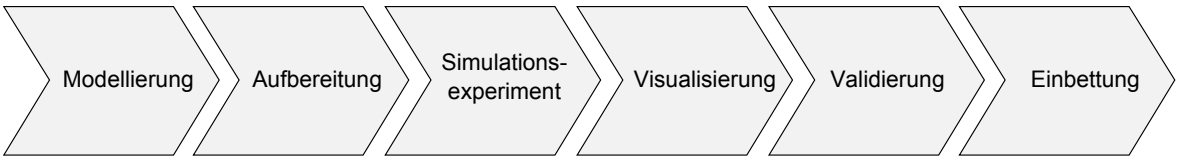

Abbildung 2.10: Simulationspipeline nach [BZBP09]

Im weiteren Sinne ist der Begriff der Simulation nicht nur mit dem eigentlichen Simulationsexperiment verbunden, sondern bezeichnet den ganzen Prozess von der Modellierung bis zur Weiterverarbeitung der Simulationsergebnisse. In [BZBP09] wird hierfür das Bild der in Abbildung 2.10 dargestellten *Simulationspipeline* verwendet. Die folgenden Schritte werden bei einer Simulation typischerweise in mehreren Feedbackschleifen durchlaufen:

Modellierung Die im vorigen Abschnitt beschriebene Herleitung des Modells aus dem realen System.

Aufbereitung Das mathematische Modell des Systems muss in der Regel entsprechend aufbereitet werden, bevor ein Simulationsexperiment darauf ausgeführt werden kann. Dies beinhaltet z.B. eine Diskretisierung kontinuierlicher Modelle und kontinuierlich beschriebener Eingangsdaten oder die Reduktion zu komplexer Modelle.

Simulationsexperiment Das Beobachten des Modellverhaltens unter Einfluss der interessierenden Eingangsdaten.

Visualisierung Die Ergebnisdaten der Simulation müssen entsprechend dargestellt werden, um ihre Interpretation zu ermöglichen.

Validierung Wie oben erwähnt ist eine Validierung der Simulationsergebnisse durch Vergleich mit Messergebnissen zwingend notwendig, um die Aussagekraft und Verlässlichkeit der Simulationsprozesse abschätzen und bewerten zu können.

Einbettung Eine Simulation ist in der Praxis kein losgelöster Prozess, sondern in übergeordnete Abläufe eingebettet. Die Ergebnisse der Festigkeitsanalyse eines Bauteils werden verwendet, um die Konstruktion zu optimieren. Die Ergebnisse einer Wettersimulation fließen in die Wettervorhersage mit ein. Oft müssen Simulationsergebnisse für die Verwendung in solchen Prozessen aufbereitet werden.

Die einzelnen Schritte der Simulationspipeline dürfen allerdings nicht als voneinander losgelöst betrachtet werden. Die Übergänge sind oft fließend und jeder Schritt muss auf den ganzen Simulationsprozess abgestimmt werden.

2.3.1 Simulationssoftware

Als Simulationssoftware werden Computerprogramme bezeichnet, die für die Computersimulation der mathematischen Modelle verwendet werden. Viele Simulationsprogramme übernehmen dabei nicht nur die für das eigentliche Simulationsexperiment notwendigen Berechnungen, sondern unterstützen den Anwender bei anderen Schritten der Simulationspipeline.

Da die Vorgehensweise bei einer Simulation stark von den Eigenschaften des verwendeten Modells und der Form seiner Diskretisierung abhängt, muss auch die verwendete Simulationssoftware auf diese Charakteristika abgestimmt sein. Daher unterscheidet sich die für die Simulation verwendete Software je nach Wissenschaftsdisziplin stark. Bei den in dieser Arbeit betrachteten Workflows aus der technischen Mechanik wird häufig die in Abschnitt 2.4.2 vorgestellte Finite-Elemente-Methode und darauf aufbauende Simulationssoftware eingesetzt. Dagegen eignet sich für Simulationen in der Verfahrenstechnik und Strömungsmechanik oft das Finite-Volumen-Verfahren, was andere Simulationssoftware notwendig macht. Eine ganz andere Herangehensweise und andere Software wird bei Simulationsanwendungen gewählt, die auf diskreten Ereignissen basieren, beispielsweise in der Verkehrsplanung oder Logistik. So umfangreich wie die Fülle an Einsatzszenarien für die Simulationstechnik ist, so breit gefächert ist auch die Auswahl an Software, die für Simulationen verwendet wird.

Dazu kommt, dass Simulationen häufig nicht auf handelsüblichen Hard- und Softwareplattformen ausgeführt werden. Oft sind die untersuchten Modelle zu komplex und die zu bearbeitenden Datenmengen zu groß, um die Simulation auf Arbeitsplatzrechnern in akzeptabler Zeit durchführen zu können. Stattdessen werden hier große, verteilte Rechnersysteme (Stichwort *Grid-Computing*) oder spezielle Hardware wie Parallel- oder Vektorrechner eingesetzt. Die Simulationssoftware muss für solche Plattformen in der Regel speziell angepasst oder entwickelt werden. Aufgrund der erwähnten Vielfalt an Simulationssoftware und den teilweise sehr schmalen Einsatzbereichen wäre eine komplette Auflistung häufig genutzter Programme an dieser Stelle nicht hilfreich. Stattdessen folgt eine rein exemplarische Auflistung an Software, um einen Überblick über die Bandbreite verfügbarer Simulationssoftware zu geben:

DUNE Die *Distributed and Unified Numerics Environment* ist eine frei lizenzierte C++-Bibliothek zur Lösung partieller Differentialgleichungen mit gitterbasierten Methoden. Sie unterstützt sowohl die Implementierung der Finite-Elemente-Methode als auch der Finite-Volumen- und Finite-Differenzen-Verfahren[13].

Arena Ein kommerzielles Programmpaket zur Simulation ereignisdiskreter Prozesse. Die Modellierung erfolgt dabei mit grafischen Tools[14].

[13] http://www.dune-project.org/
[14] http://www.arenasimulation.com/

AnyLogic Ein kommerzielles Java-Programm zur Erstellung und Simulation kontinuierlicher, ereignisdiskreter und agentenbasierter Modelle. Die Modelle können grafisch und mittels Java-Code erstellt werden[15].

Matlab/Simulink Die Numerikumgebung Matlab und das darauf aufbauende Simulationsprogramm Simulink werden in Abschnitt 2.5.1 näher vorgestellt.

2.3.2 Simulationsworkflows

In Abschnitt 2.3 wurde die Simulationspipeline vorgestellt, die den Simulationsprozess als Abfolge von mehreren, logisch aufeinander aufbauenden Schritten darstellt. Häufig wird dieser komplette Simulationsvorgang in unterschiedlichen Ausprägungen wiederholt ausgeführt, etwa um das Verhalten eines Systems unter verschiedenen Randbedingungen zu untersuchen oder um in der Produktentwicklung verschiedene Produktentwürfe einer Belastungssimulation zu unterziehen. In evolutionären Optimierungsverfahren wird die Simulation teilweise sogar in automatisierte Iterationszyklen eingebunden [Wei02]. Es liegt also nahe, den kompletten Vorgang oder Teile davon als wiederholbaren, aus Einzelschritten bestehenden Prozess zu betrachten und als automatisierbaren Workflow zu modellieren.

Diese Vorgehensweise bietet besonders im Hinblick auf die aktuellen Veränderungen in der wissenschaftlichen Methodik große Vorteile. Durch die zunehmende elektronische Vernetzung wird die nahtlose, weltweite Zusammenarbeit unterschiedlicher Forschungsinstitutionen ermöglicht. Die Forschung basiert dabei in zunehmendem Maße auf Simulationen komplexer Systeme und der Untersuchung großer Datenmengen, die auf ebenso weltweit verteilten Rechnernetzen ausgeführt werden. Diese, als *e-Science* zusammengefassten Trends führen zwangsläufig zu einer Nachfrage nach standardisierten Protokollen und Schnittstellen zur automatisierten elektronischen Kommunikation in verteilten, heterogenen IT-Landschaften. Aus vergleichbaren Anforderungen heraus entstanden in der Geschäfts-IT die Technologien und Konzepte rund um SOA, Webservices, Workflows und BPEL.

Doch obwohl in der Wirtschaftsinformatik schon seit vielen Jahren in diesem Bereich geforscht wird und mittlerweile eine große Zahl erprobter Anwendungen zur Orchestrierung von Webservices zu Workflows verfügbar sind, wird diese Technologie im wissenschaftlichen Bereich noch nicht auf breiter Basis eingesetzt. In [Tay07] werden einige Unterschiede zwischen Business Workflows und Scientific Workflows identifiziert, die diesen Umstand erklären können.

Zum einen sind die Anforderungen an die Flexibilität wissenschaftliche Prozesse in der Praxis oft höher als an Geschäftsprozesse. Während Vorgänge aus der Wirtschaft, wie das Eröffnen eines Bankkontos, einer klar definierten Vorgabe folgen, kann ein Wissenschaftler meist spontan entscheiden, ob er z.B. Messdaten zuerst filtert, bevor er sie zur Erzeugung eines Simulationsmodells nutzt. Workflow-Management-Systeme müssen diese flexible Vorgehensweise unterstützen, um in der Wissenschaft eingesetzt zu werden.Ein weiterer Unterschied liegt in den personellen Strukturen und den üblichen Arbeitsabläufen: In der Wirtschaft wird für die Implementierung

[15] http://www.xjtek.com/

ausführbarer Workflows meist auf die Kenntnisse von IT-Spezialisten gesetzt. Die Wirtschaftsinformatik schließt hier die Lücke zwischen wirtschaftlichem und technischem Expertenwissen. Wissenschaftliche Workflows werden dagegen in der Regel von den Wissenschaftlern selbst erstellt und verwaltet, die nicht zwingend Experten im Bereich der Datenverabeitung und Anwendungsintegration sind. Workflow-Management-Systeme müssen für den Einsatz in wissenschaftlichen Bereichen also auch ohne tief gehendes, informationstechnisches Expertenwissen nutzbar sein.

Verschiedene Projekte haben sich deshalb die Entwicklung von Workflow-Management-Systemen zum Ziel gesetzt, die speziell auf die Anforderungen von Scientific Workflows ausgelegt sind. Beispiele dafür sind Taverna[16] und Kepler[17]. Diese Systeme können bei bestimmten wissenschaftlichen Anwendungsfällen einige Vorteile gegenüber BPEL-basierten Systemen aufweisen. Besonders für Taverna spricht die Verfügbarkeit einer integrierten, quelloffen vertriebenen Umgebung zur Entwicklung und Ausführung wissenschaftlicher Workflows. Allerdings sind diese Systeme im Vergleich zu BPEL-basierten Systemen weniger gut erprobt und rund um diese Systeme existieren wesentlich weniger Tools und Softwareprodukte als in der BPEL-Welt. Außerdem sind diese Systeme meist auf bestimmte wissenschaftliche Bereiche ausgerichtet und verfügen im Vergleich zu BPEL über ein schmaleres Spektrum an Einsatzmöglichkeiten. Der Vergleich verschiedener wissenschaftlicher Workflow-Systeme ist allerdings nicht Teil dieser Arbeit, hierzu sei auf Arbeiten wie [CG08] und [TMF+10] verwiesen.

2.4 Mechanische Grundlagen

In dieser Arbeit werden die bereits vorgestellten Workflow-Technologien zur Bearbeitung einer Aufgabenstellung aus der technischen Mechanik verwendet. Genauer gesagt wird ein Workflow zur Durchführung einer Modellreduktion vorgestellt. Dabei wird das Modell eines elastischen Mehrkörpersystems reduziert, diese Klasse von Systemen wird in Abschnitt 2.4.1 kurz vorgestellt. Abschnitt 2.4.2 gibt einen Einblick in die Finite-Elemente-Methode, die zur anschließenden Simulation des Systemverhaltens verwendet wird. Schließlich stellt Abschnitt 2.4.3 die Grundlagen der Modellreduktion vor.

2.4.1 Elastische Mehrkörpersysteme

Unter einem Mehrkörpersystem wird in der Mechanik ein System von Einzelkörpern verstanden, die durch Bindungselemente verbunden sind und unter dem Einfluss von Kräften und Momenten stehen. Anschauliche Beispiele für Mehrkörpersysteme sind Pendel, Kurbeltriebe, Radaufhängungen oder Greifarme von Robotern. In einem klassischen Mehrkörpersystem werden die einzelnen Körper als starr, also als unveränderlich in ihrer Form betrachtet. Durch die Bewegungsgleichungen der einzelnen Körper und einer mathematischen Formulierung der

[16] http://www.taverna.org.uk/
[17] https://kepler-project.org/

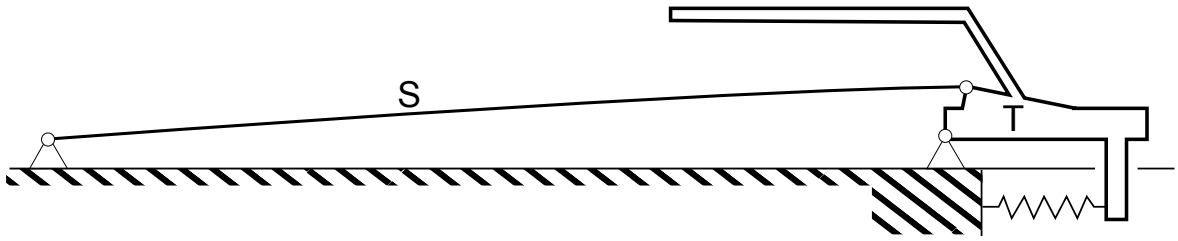

Abbildung 2.11: Schematische Darstellung des Tremolos einer elektrischen Gitarre

geometrischen Zwangsbedingungen (z.B. mechanische Verbindungen verschiedener Körper) lassen sich für diese Systeme gewöhnliche Differentialgleichungssysteme meist geringer Dimension aufstellen. Die Bewegungen der Einzelkörper und die im System auftretenden Kräfte werden dann durch die numerische oder (in einfachen Fällen) analytische Lösung dieser Gleichungssysteme errechnet. In vielen realen Fällen können die elastischen Verformungen der Körper aber nicht einfach vernachlässigt werden. Man spricht dann von *elastischen Mehrkörpersystemen (EMKS)*.

Abbildung 2.11 zeigt als Beispiel die schemenhafte Darstellung der Tremolovorrichtung einer elektrischen Gitarre, mit der während des Spielens die Tonhöhe variiert werden kann. Das System besteht im Wesentlichen aus zwei Körpern, der Saite (S) und dem eigentlichen Tremolo (T). Während das Tremolo für die meisten Untersuchungen als starr betrachtet werden kann, ist dies bei der Saite nicht möglich: Ohne elastische Verformungen würde beim Spielen kein Ton entstehen.

Die Bewegung jedes Punktes eines herkömmlichen Mehrkörpersystems kann durch Bewegungsgleichungen für jeden translatorischen und rotatorischen Freiheitsgrad der verschiedenen Einzelkörper beschrieben werden. Das mathematische Modell des Systems besteht also aus abzählbar vielen gewöhnlichen Differentialgleichungen. Bei elastischen Mehrkörpersystemen ist dies nicht mehr möglich: Die Bewegung jedes Punktes hängt neben der Bewegung auch von der Verformung der Körper ab, die an jedem Punkt unterschiedliche Auswirkungen zeigt. Jeder Punkt auf der oben erwähnten Gitarrensaite schwingt mit unterschiedlicher Amplitude. Zur Beschreibung dieser Bewegungen sind partielle Differentialgleichungen notwendig, die das Verhalten des Systems abhängig vom Ort und der Zeit beschreiben.

Das Lösen partieller Differentialgleichungen ist im allgemeinen wesentlich komplizierter als das Lösen gewöhnlicher Differentialgleichungen, in der Regel ist hier der Einsatz numerischer Verfahren notwendig [Ran08].

2.4.2 Finite Elemente Methode

Die Finite Elemente Methode (FEM) ist ein numerisches Verfahren zur approximativen Lösung von partiellen Differentialgleichungen, wie sie bei der Modellierung elastischer Mehrkörpersysteme auftreten. Die Grundidee der Methode besteht darin, das zu untersuchende Gebiet, also beispielsweise das Volumen eines elastischen Körpers, in eine große Zahl geometrisch einfacher Elemente wie Balken, Platten oder Tetraeder zu unterteilen, deren Verhalten dann mit

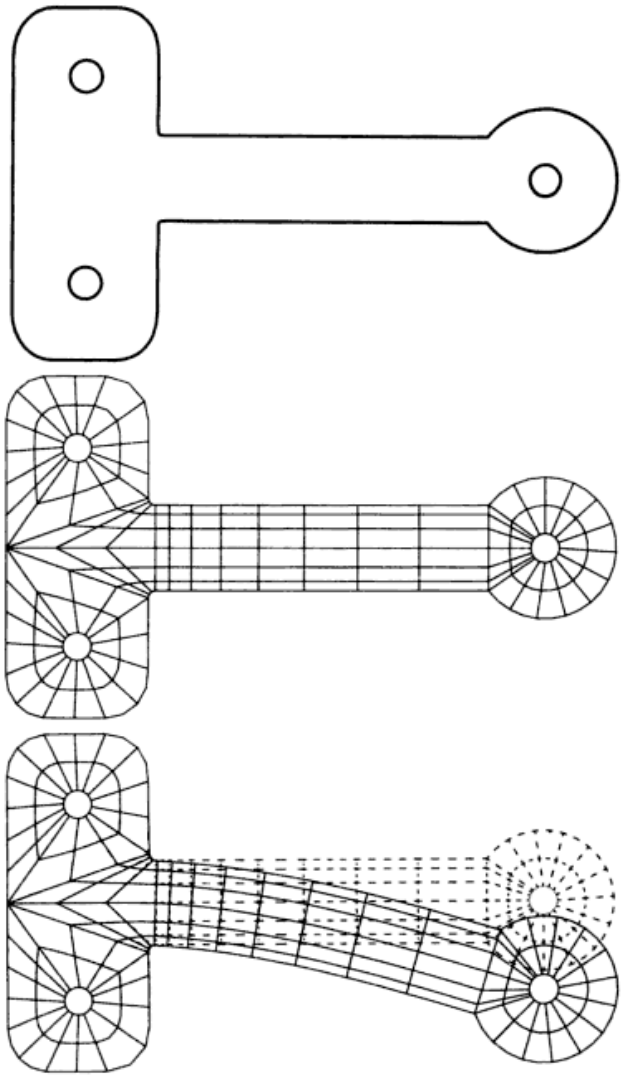

Abbildung 2.12: Querschnitt eines Trägers (oben), FE-Diskretisierung (mitte) und Ergebnis einer Belastungssimulation (unten) [BZ01]

einfacheren Gleichungen beschrieben werden kann. Im Gegensatz zu den Infinitesimalelementen, die bei der analytischen Integration verwendet werden, zeichnen sich diese Elemente durch eine endliche (finite) Größe aus, woraus sich der Name der Methode herleitet.

Diese *Diskretisierung* ist dabei keinesfalls trivial und hat großen Einfluss auf das Ergebnis der Untersuchung. Wird das Gitter an den entscheidenden Stellen zu grob gewählt, werden unter Umständen entscheidende Effekte nicht berücksichtigt. Abbildung 2.12 stellt die Diskretisierung eines Trägers und das Ergebnis einer anschließenden Simulation der Verformung unter Einwirkung einer äußeren Kraft dar.

Für die Elemente des entstehenden Gitters werden Differentialgleichungen aufgestellt, die das Verhalten des jeweiligen Elements in Abhängigkeit seiner Nachbarn beschreiben. Wenn das Referenz-Koordinatensystem mit dem elastischen Körper mitbewegt wird, werden die Relativbewegungen der Elemente nur durch die elastischen Verformungen bestimmt, welche als klein und damit linear angenommen werden [SW99]. Die partiellen Differentialgleichungen werden so in ein System gewöhnlicher Differentialgleichungen umgewandelt, welches numerisch integriert werden kann. Die Systemdimension errechnet sich aus der Anzahl der Elemente multipliziert mit der Anzahl der Freiheitsgrade der einzelnen Elemente, sie wird in der Praxis also sehr groß. Allerdings sind die Systemmatrizen in der Regel dünn besetzt, da ein Element nur mit seinen Nachbarn in direkter Wechselwirkung steht.

Bei der Anwendung der Finite-Elemente-Methode zur Simulation elastischer Körper ergibt sich für das System der Dimension n nach [Oh10] die folgende Bewegungsgleichung:

$$\begin{aligned} \boldsymbol{M}_e \cdot \ddot{\boldsymbol{q}}(t) + \boldsymbol{D}_e \cdot \dot{\boldsymbol{q}}(t) + \boldsymbol{K}_e \cdot \boldsymbol{q}(t) &= \boldsymbol{B}_e \cdot \boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}_e \cdot \boldsymbol{q}(t) \end{aligned} \tag{2.1}$$

Dabei sind sind in $\boldsymbol{q}(t) \in \mathbb{R}^n$ die Knotenkoordinaten des FE-Gitters enthalten, $\boldsymbol{u}(t) \in \mathbb{R}^p$ enthält die Systemeingänge wie externe Kräfte und $\boldsymbol{y}(t) \in \mathbb{R}^q$ dient als Systemausgang, enhält also Bewegungen der interessierenden Knoten. $\boldsymbol{M}_e \in \mathbb{R}^{n \times n}$ wird als Massenmatrix, $\boldsymbol{D}_e \in \mathbb{R}^{n \times n}$ als Dämpfungsmatrix und $\boldsymbol{K}_e \in \mathbb{R}^{n \times n}$ als Steifigkeitsmatrix bezeichnet. Die Eingangsmatrix $\boldsymbol{B}_e \in \mathbb{R}^{n \times p}$ definiert, auf welche Knoten und in welcher Richtung die externen Kräfte wirken; die Ausgangsmatrix $\boldsymbol{C}_e \in \mathbb{R}^{q \times n}$ dient zur Bestimmung der interessierenden Knoten.

Die mathematischen Hintergründe der Finite-Elemente-Methode gehen über den Umfang dieser Arbeit hinaus. Zu weiteren Informationen wird deshalb auf Fachliteratur wie [BZ01] verwiesen.

2.4.3 Modellreduktion

Die Verwendung von Computerprogrammen zur Erstellung und Diskretisierung von Simulationsmodellen erlaubt die Berücksichtigung vieler Freiheitsgrade und feiner Gitter, was schnell zu sehr großen Systemdimensionen führt. Die so erstellten Modelle weisen dann zwar ein dem realen Modell ähnliches Verhalten auf, allerdings sind die darauf ausgeführten Simulationsrechnungen sehr komplex und zeitintensiv, so dass die Arbeit mit den Modellen oft nicht mehr praktikabel ist. Mit Modellreduktionsverfahren versucht man deshalb, den Umfang des Modells zu reduzieren, ohne dabei das charakteristische Verhalten des Systems zu verlieren.

Bei den für diese Arbeit relevanten Reduktionsverfahren wird dazu das Originalsystem der Dimension n auf einen geeigneten Unterraum ν der Dimension $m \ll n$ projeziert ([Oh10]):

$$\boldsymbol{q}(t) \approx \boldsymbol{V} \cdot \bar{\boldsymbol{q}}(t) \tag{2.2}$$

Hier enthält $\bar{\boldsymbol{q}}(t) \in \mathbb{R}^m$ die verallgemeinerten Koordinaten des reduzierten Systems, $\boldsymbol{V} \in \mathbb{R}^{n \times m}$ ist die Projektionsmatrix, die auf verschiedene Arten aufgestellt werden kann. Neben den, im folgenden Abschnitt vorgestellten, modalen Reduktionsverfahren existiert dafür unter anderem noch die Methode der Krylov-Unterräume und das Craig-Bampton-Verfahren.

Modale Reduktion

Modale Reduktionsverfahren sind die bekanntesten Verfahren zur Modellreduktion und werden meistens zur Reduktion der Modelle elastischer Mehrkörpersysteme verwendet. Der Unterraum ν, auf den das Originalsystem projeziert wird, wird hier durch ausgewählte Eigenvektoren des Systems aufgespannt. Zunächst wird dazu aus dem System 2.1 durch Einsetzen des Ansatzes $\boldsymbol{q} = e^{\lambda_i t}\boldsymbol{\phi}_i$ mit $\lambda_i \in \mathbb{C}$ und $\boldsymbol{\phi}_i \in \mathbb{C}^n$ ein quadratisches Eigenwertproblem aufgestellt (vgl. [FE11]):

$$(\lambda_i^2 \boldsymbol{M}_e + \lambda_i \boldsymbol{D}_e + \boldsymbol{K}_e) \cdot \boldsymbol{\phi}_i = 0 \tag{2.3}$$

Nun lassen sich die Eigenvektoren $\boldsymbol{\phi}_i$, auch Eigenmoden genannt, berechnen und eine Auswahl davon in der Modalmatrix $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1 \dots \boldsymbol{\phi}_m]$ zusammenstellen, die dann als Projektionsmatrix $\boldsymbol{V} = \boldsymbol{\Phi}$ verwendet wird. Die Auswahl der bei der Reduktion berücksichtigten Eigenmoden bestimmt das Verhalten des reduzierten Systems. In der Regel sind besonders die Eigenmoden interessant, deren zugehörige Eigenwerte einen kleinen Betrag aufweisen, da durch sie das niederfrequente Verhalten des Systems charakterisiert wird, welches meistens die größten Amplituden aufweist (vgl. [GK89]).

2.5 Verwendete Software

In diesem Abschnitt wird verschiedene Software vorgestellt, die für die Implementierung und Ausführung des in Kapitel 4 vorgestellten Workflows verwendet wurde.

2.5.1 MATLAB

Matlab ist ein Programmpaket des Herstellers *The Mathworks*[18] und dient der Durchführung verschiedenster numerischer Berechnungen sowie der grafischen Darstellung der Ergebnisse. Kernstück von Matlab ist seine proprietäre Programmiersprache, die speziell für die Formulierung mathematischer Aufgaben entworfen wurde. Das Programmpaket Matlab dient dabei sowohl als Interpreter als auch als Integrierte Entwicklungsumgebung (IDE) für diese Sprache. Ein weiterer integraler Bestandteil von Matlab ist ein Werkzeug zur interaktiven, zwei- oder dreidimensionalen Visualisierung von Datenreihen. In dieser Arbeit wird Matlab in der Version R2010b genutzt.

[18] http://www.mathworks.de/index.html

+	Addition von Skalaren oder Matrizen
-	Subtraktion von Skalaren oder Matrizen
*	Multiplikation und Matrixmultiplikation
.*	Elementweise Multiplikation zweier Matrizen
/, \	Division und rechts- bzw. linksseitige Matrixdivision
./, .\	Elementweise Division zweier Matrizen
^	Exponentiation und Matrixexponentiation
.^	Elementweise Exponentiation zweier Matrizen
'	Komplexe Konjugation einer Matrix (unärer Operator)
.'	Nicht-komplexe Konjugation einer Matrix (unärer Operator)

Tabelle 2.2: Operatoren in Matlab, vgl. [Sch99]

Matlab-Programmanweisungen können in einer Shell interaktiv ausgeführt werden. Umfangreichere Scripts oder Funktionen werden dagegen in Dateien mit der Endung .m hinterlegt. Für die Arbeit mit diesen *m-Files* bietet Matlab einen Editor sowie Debugger, Profiler und weitere Entwicklungswerkzeuge.

Sämtliche Arbeitsdaten einer laufenden Matlab-Sitzung werden im sogenannten *Workspace* hinterlegt. Dieser lässt sich während einer interaktiven Sitzung mit einem grafischen Browser durchsuchen. Die Inhalte des Workspaces sind im Arbeitsspeicher des Rechners hinterlegt und verschwinden beim Beenden von Matlab. Sie lassen sich allerdings jederzeit vollständig in einer Datei mit der Endung .mat in einem Matlab-spezifischen Format persistent abspeichern und später wieder importieren. Dadurch lässt sich die Arbeit in Matlab jederzeit unterbrechen und an genau dieser Stelle wieder aufnehmen.

Sprachfunktionen

Der Name MATLAB ist eigentlich eine Abkürzung für MATrix LABoratory. Die Fokussierung auf die Arbeit mit Matrizen manifestiert sich besonders im Umfang der Matlab-eigenen Programmiersprache. Sie ist schwach typisiert und auf die Arbeit mit mehrdimensionalen Datentypen wie Matrizen und Vektoren ausgelegt. Standarddatentyp ist in Matlab die Matrix mit Fließkommazahlen doppelter Genauigkeit (vgl. [Sch99]). Die Anweisung `a=3` weist der Variablen a also keinen skalaren, ganzzahligen Wert, sondern eine mit dem Wert 3.0 besetzte 1×1-Matrix zu. Wie in Tabelle 2.2 dargestellt können sämtliche Operatoren in Matlab auch auf Matrizen bzw. Vektoren ausgeführt werden. Die meisten vordefinierten Funktionen akzeptieren ebenfalls mehrdimensionale Daten als Argumente und führen ihre Operationen, falls diese nur für Skalare definiert sind, elementweise aus. Die tief gehende Unterstützung mehrdimensionaler Datentypen erlaubt die weitreichende Arbeit mit Matrizen und Vektoren, ohne auf Schleifen zurückgreifen zu müssen. Diese Eigenschaft von Matlab erlaubt nicht nur die einfache Parallelisierbarkeit der Ausführung, sondern sorgt auch für eine sehr kompakte Darstellung des Codes, die an die in der Mathematik übliche Schreibweise erinnert und deshalb vielen Wissenschaftlern ein gewohntes Bild liefert.

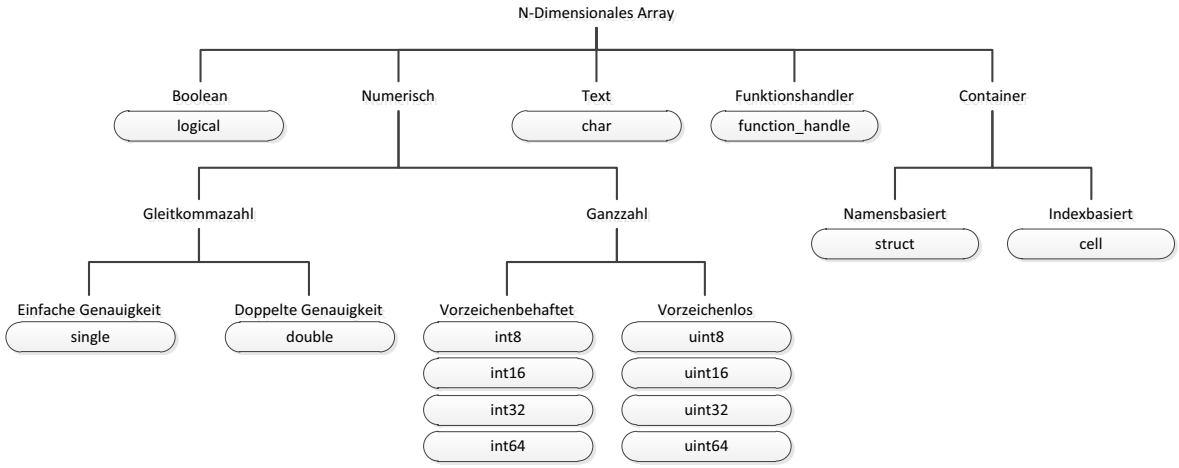

Abbildung 2.13: Datentypen in Matlab (Version R2010b)[Mat]

Neben Fließkommazahlen doppelter Genauigkeit unterstützt Matlab noch eine Reihe weiterer Datentypen, die ebenfalls nicht als Skalare, sondern als mehrdimensionale Datentypen behandelt werden. Abbildung 2.13 gibt eine Übersicht über die in Matlab verfügbaren Datentypen. Zu beachten ist insbesondere, dass Funktionshandler als normale Variablen behandelt werden und als solche an andere Funktionen übergeben oder für die Speicherung in mat-Files serialisiert werden können. Neben der normalen Matrixdarstellung kennt Matlab auch ein spezielles Speicherformat für dünn besetzte Matrizen. Dabei werden nur die von 0 verschiedenen Elemente einer Matrix explizit gespeichert.

Seit Version 5 ist in Matlab auch die objektorientierte Programmierung möglich. Dabei wird die Definition von Klassen und Paketen, die Überladung von Funktionen und Operatoren und die (Mehrfach-)Vererbung von Klassen unterstützt.

Toolboxes

Mit dem Basispaket von Matlab können bereits vielfältige numerische Aufgaben ausgeführt werden. Für fachspezifische Anwendungen kann außerdem auf eine Reihe von *Toolboxes* zurückgegriffen werden, die den Umgang von Matlab erweitern und zusätzliche Funktionen definieren. Diese müssen allerdings separat erworben werden. Unter anderem sind folgende Toolboxes verfügbar:

Symbolic Math Toolbox Erlaubt das computergestützte symbolische Rechnen, ähnlich wie in *Maple*[19] oder *Mathematica*[20].

Control System Toolbox Bietet Unterstützung bei der Untersuchung dynamischer Systeme und dem Entwurf von Reglern.

[19] http://www.maplesoft.com/
[20] http://www.wolfram.com/mathematica/

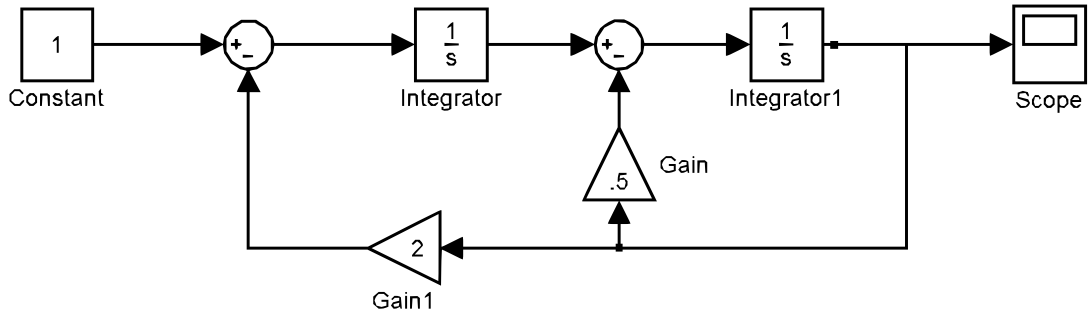

Abbildung 2.14: Einfaches System in Simulink

System Identification Toolbox Unterstützt bei der Modellierung eines dynamischen Systems anhand von Messdaten.

Image Processing Toolbox Ermöglicht die Bildverarbeitung in Matlab.

Simulink

Simulink ist ein auf Matlab basierendes Computerprogramm zur Modellierung und Simulation dynamischer Systeme. Der Anwender kann mathematische Modelle in Simulink in Form eines grafischen Blockschaltbilds[21] erstellen (siehe Abbildung 2.14). Simulink stellt ihm dafür verschiedene Blöcke zur Verwendung zeitkontinuierlicher oder -diskreter mathematischer Funktionen zur Verfügung. Wie Matlab lässt sich Simulink um zusätzliche Blöcke für Aufgabenstellungen aus spezifischen Fachbereichen ergänzen. Beispielsweise existieren Erweiterungen zur Modellierung von elektrischen Netzwerken, von Antriebssträngen in der Fahrzeugtechnik oder von ereignisdiskreten Systemen.

Zur Simulation selbst sind in Simulink verschiedene numerische Solver mit fester oder variabler Schrittweite integriert. Die Simulationsergebnisse können anschließend grafisch ausgegeben oder in Matlab weiterverarbeitet werden. Über Erweiterungen ist auch die direkte Weitergabe an verschiedene Geräte möglich, was die Integration in reale Experimente erlaubt.

Simulink ist in der Anwendung eng mit Matlab verzahnt. Datenreihen können vom Matlab-Workspace eingelesen und in den Workspace zurückgeschrieben werden. Darüber hinaus ist es möglich, beliebige Matlab-Funktionen als Simulink-Block zu verwenden oder umgekehrt die Simulation von Simulink-Modellen aus einem m-File heraus zu starten.

Für die in dieser Arbeit behandelte Aufgabenstellung spielt Simulink keine weitere Rolle. Durch die enge Integration in Matlab können allerdings die in Abschnitt 4.2 vorgestellten Möglichkeiten zur Einbindung von Matlab in Simulationsworkflows auch zur Einbindung von Simulink genutzt werden, was die Relevanz dieses Abschnitts für die Praxis erhöht.

[21] Blockschaltbilder sind eine, insbesondere in der Regelungstechnik verbreitete Form der Darstellung eines dynamischen Systems. In den Schaubildern stehen die Blöcke für mathematische Funktionen und Filter, während die Pfeile den rückkopplungsfreien Signalfluss zwischen den Blöcken darstellen.

Schnittstellen

Version R2010b von Matlab läuft unter Windows, Linux oder Mac OS X und wird üblicherweise über eine grafische Oberfläche bedient. Für die Verwendung von Matlab in einem Workflow sind allerdings die Schnittstellen entscheidend, über die Matlab von anderen Anwendungen gesteuert werden kann. Die Schnittstellen, die Matlab zu diesem Zweck zur Verfügung stellt werden in diesem Abschnitt vorgestellt, während Abschnitt 4.2 dann auf die konkrete Workflowintegration von Matlab eingeht.

Viele Funktionen von Matlab lassen sich auch über die Kommandozeile des jeweiligen Betriebssystems nutzen und können so von externen Programmen wie dem in Abschnitt 2.2.3 vorgestellten Generic Web Service Interface aufgerufen werden. Im einfachsten Fall wird Matlab dazu mit dem Kommandozeilenparameter `-r` ein m-file übergeben, welches die nötigen Berechnungen durchführt und die Ergebnisse dann in einem geeigneten Format in einer Datei speichert. Daten aus Matlab können dabei in einer ganzen Reihe von Formaten wie XML, CSV oder Excel-Spreadsheets exportiert werden, eine vollständige Auflistung der unterstützen Formate findet sich in der Matlab-Dokumention[22].

Ist der verlustfreie Export des Matlab-Workspaces gewünscht, sollten die Daten im proprietären Matlab-Format in mat-Files gespeichert werden. Für den Zugriff auf mat-Files aus externen, in C/C++ oder Fortran geschriebenen Applikationen steht eine Programmbibliothek zur Verfügung.

Natürlich kann Matlab per Kommandozeile auch interaktiv genutzt werden. Unter Unix-artigen Betriebssystemen ist es möglich, die Dateneingabe über Pipes[23] von anderen Prozessen zu beziehen und die Datenausgabe auf dem gleichen Weg an andere Prozesse weiterzuleiten. Matlab agiert in diesem Fall als Filterprogramm. Die Windows-Version von Matlab unterstützt dagegen die *Component Object Model (COM)*-Technologie von Microsoft. Matlab kann über diese IPC-Plattform als Server betrieben und dann von einer anderen Software ferngesteuert werden. Der Umfang dieser Schnittstelle übersteigt das bloße Ausführen von Matlab-Code und erlaubt auch den Austausch von Variablen zwischen Client und Matlab-Server.

Unter dem Namen *Matlab Engine* wird eine Programmbibliothek mitgeliefert, die die Steuerung durch externe Software erleichtert. Wird die Bibliothek in C/C++- oder Fortran-Anwendungen eingebunden, ist der Zugriff auf Matlab durch direkte Funktionsaufrufe im Quellcode der Anwendung möglich. Die Matlab Engine verwendet zur eigentlichen Kommunikation mit Matlab je nach Betriebssystem Pipes oder die COM-Plattform.

Nur am Rande sei erwähnt, dass Matlab auch sehr umfangreiche Möglichkeiten bietet, um auf externe Programme und Dienste zuzugreifen. Unter anderem wird die Verwendung von

[22] http://www.mathworks.de/help/techdoc/import_export/bst9qgh.html

[23] Als Pipe werden vom Betriebssystem bereitgestellte Werkzeuge zur Interprozesskommunikation (IPC) bezeichnet. Pipes können von zwei Prozessen geöffnet und mit Daten gefüllt, beziehungsweise ausgelesen werden. Die Kommunikation erfolgt dabei nach dem FIFO-Prinzip (First in, First out): Die Daten können von einem Prozess nur in der Reihenfolge ausgelesen werden, in der sie vom anderen geschrieben wurden. Es wird zwischen anonymen und benannten Pipes unterschieden. Letztere werden unter Unix ähnlich wie Dateien behandelt.

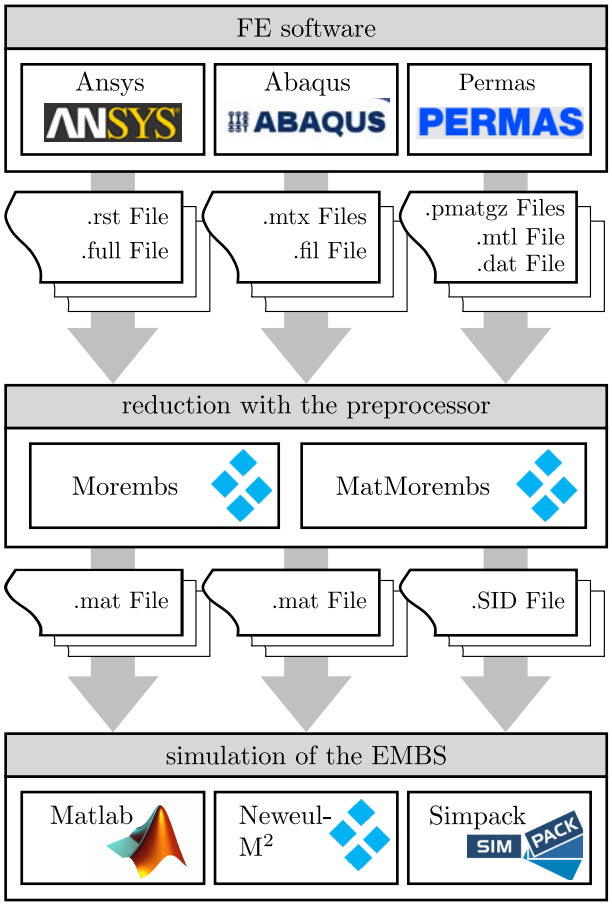

Abbildung 2.15: Datenfluss in Morembs ([NFE11])

C/C++-, Java- oder .NET-Bibliotheken und der Aufruf von Webservices unterstützt. Für den Zugriff auf externe Geräte über die Serielle Schnittstelle steht ebenfalls eine API zur Verfügung.

2.5.2 MatMorembs

MOREMBS[24] steht für *Model Order Reduction in Elastic Multi-Body Systems* und ist ein vom Institut für Technische und Numerische Mechanik (ITM) der Universität Stuttgart entwickeltes Werkzeug zur Reduktion der Freiheitsgrade des Modells eines elastischen Mehrkörpersystems. Das ursprüngliche Morembs-Programmpaket ist in C++ geschrieben. Mit MatMorembs existiert eine, im Umfang vergleichbare Matlab-Implementierung der Software, welche für diese Arbeit eine wichtigere Rolle spielt. Dieser Abschnitt basiert auf den Informationen aus der MatMorembs-Dokumentation [FN].

[24] http://www.itm.uni-stuttgart.de/research/model_reduction/MOREMBS_de.php

Ausgangspunkt der Modellreduktion ist immer ein diskretisiertes Modell eines elastischen Körpers, welches in der Regel mit einem Finite-Elemente-Programm erstellt wird. Morembs bzw. MatMorembs unterstützen den Import der Modelldaten aus den gänigen FE-Programmen ANSYS[25], ABAQUS[26] und PERMAS[27]. Anschließend erfolgt die Durchführung der Modellreduktion. Diese kann unter Verwendung verschiedener erprobter und experimenteller Reduktionsverfahren durchgeführt werden. Neben dem modalen Reduktionsverahren (vgl. Abschnitt 2.4.3) sind unter anderem auch das Craig-Bampton-Verfahren und ein auf Krylov-Unterräumen basierendes Reduktionsverfahren in Morembs bzw. MatMorembs implementiert. Abschließend können die reduzierten Modelldaten für die Verwendung in einem Simulationsprogramm exportiert werden. Auch hier stehen verschiedene Formate zur Auswahl.

2.5.3 Apache ODE

Apache ODE (Orchestration Director Engine)[28] ist eine Workflowengine, die in WS-BPEL (siehe Abschnitt 2.2.2) modellierte Workflows ausführen kann. ODE wird unter dem Dach der *Apache Software Foundation*[29] als quelloffene Software entwickelt und vertrieben.

ODE enthält selbst keine Funktionalität zur Kommunikation mit der Außenwelt. Stattdessen integriert sich die Software dazu über die sogenannten ODE Integration Layers mit fremden Kommunikations-Frameworks wie *Axis2*[30] oder *JBI*[31]-Implementierungen. Im Rahmen dieser Arbeit wird für die Kommunikation zwischen der Workflowengine und den Service-Providern beziehungsweise dem Benutzer das Webservice-Toolkit Axis2 genutzt. Apache ODE und die von Axis2 bereitgestellte Webservice-Anbindung laufen in diesem Fall als Servlet unter dem Servlet-Container *Apache Tomcat*.

Ein explizites Entwicklungsziel von Apache ODE ist die verlässliche Ausführung von langlaufenden Workflows auch auf unverlässlichen Systemen. Dazu muss ODE in der Lage sein, den jeweiligen Zustand der Workflowinstanzen transaktionssicher zu speichern. Auch hier implementiert ODE keine eigene Funktionalität, sondern kann über die *ODE Data Access Objects (DAOs)* mit verschiedenen Datenbank-Management-Systemen verbunden werden [Apa].

[25] http://www.ansys.com/
[26] http://www.simulia.com/
[27] http://www.intes.de/
[28] http://ode.apache.org/
[29] Die Apache Software Foundation(http://www.apache.org/) ist eine ehrenamtliche Organisation zur Förderung der Entwicklung einer Reihe von quelloffenen Softwareprojekten. Diese stehen unter der Apache-Lizenz, einer pragmatischen Open-Source-Lizenz, welche die freizügige Weiterverwendung der damit lizenzierten Software erlaubt.
[30] Apache Axis2(http://axis.apache.org/axis2/java/core/) ist ein Webservice-Toolkit, mit dem sowohl Webservice-Clients als auch Webservice-Server implementiert werden können. Es kann unter anderem dafür verwendet werden, anderen Anwendungen SOAP-basierte Webservice-Funktionalität bereitzustellen. Es stehen Implementierungen in Java und C zur Verfügung.
[31] Java Business Integration(http://jcp.org/aboutJava/communityprocess/final/jsr208/index.html)

Die Implementierung dieser DAOs kann z.B. über einen Persistenz-Manager wie Hibernate[32] oder openJPA[33] erfolgen.

[32] http://www.hibernate.org
[33] http://openjpa.apache.org

3 Konzept eines Workflows zur Modellreduktion

In diesem Kapitel wird ein Workflow zur computergestützten Ausführung einer Modellreduktion vorgestellt. Als zu reduzierendes Modell wird hier beispielhaft das Finite-Elemente-Gitter eines elastischen Mehrkörpersystems verwendet. Zunächst steht dabei die sinnvolle Aufteilung des Prozesses in einzelne Workflow-Schritte und die Definition ihrer Ausführungslogik mittels grafischer Notationen im Vordergrund. Die konkrete Implementierung in BPEL wird in Kapitel 4 behandelt. Da sich diese auf MatMorembs (siehe Abschnitt 2.5.2) stützt, orientieren sich auch das hier vorgestellte Konzept des Workflows und die Aufgaben der einzelnen Schritte am Umfang dieses Programmpakets.

Zunächst wird der grundsätzliche Ablauf des Prozesses vorgestellt und die Aspekte erläutert, die bei seinem Entwurf berücksichtigt wurden (3.1). Die einzelnen Teilprozesse zur Datenvorbereitung (Abschnitt 3.2), der eigentlichen Modellreduktion (Abschnitt 3.3), der Auswertung und Visualisierung der Ergebnisse (Abschnitt 3.4) und dem abschließenden Datenexport (Abschnitt 3.5) werden in den folgenden Abschnitten näher erklärt. Schließlich geht Abschnitt 3.6 darauf ein, wie alternative Workflows aus dem hier ausgearbeiteten Konzept erstellt werden können.

3.1 Ablauf der Modellreduktion

Obwohl die Modellreduktion selbst nur ein Schritt in einem übergeordneten Simulationsvorgang ist, muss sie nicht als atomarer Vorgang behandelt werden, sondern lässt sich in weitere Schritte unterteilen. Dabei lassen sich insbesondere die Vorbereitung der Modellreduktion sowie die Auswertung und Nachbearbeitung des reduzierten Modells von der eigentlichen Modellreduktion trennen. Im Folgenden werden die so getrennten Prozessabschnitte als *Preprocessing*-, *Reduktions*- und *Postprocesssing*-Phase bezeichnet. Diese Bezeichnungen beziehen sich hier nur auf die Phasen der Modellreduktion und sind nicht mit den Pre- und Postprozessoren in FE-Programmen zu verwechseln. Abbildung 3.1 zeigt diese erste Aufteilung des Reduktionsprozesses in einem BPMN-Diagramm. Es ist bereits eine Iterationsschleife vorgesehen, um die weiter unten erläuterte Konfiguration der Modellreduktion anzupassen, falls das Ergebnis nicht den gestellten Anforderungen genügt.

Sowohl die Reduktions- als auch die Pre- und Postprocessing-Phasen können solange in weitere Schritte unterteilt werden, bis am Ende in jedem Schritt nur noch eine einzelne, grundlegende Rechenoperation ausgeführt wird. Dies dient allerdings weder der Übersichtlichkeit, der einfachen Implementierbarkeit noch der Ausführungseffizienz des Workflows. Vor der

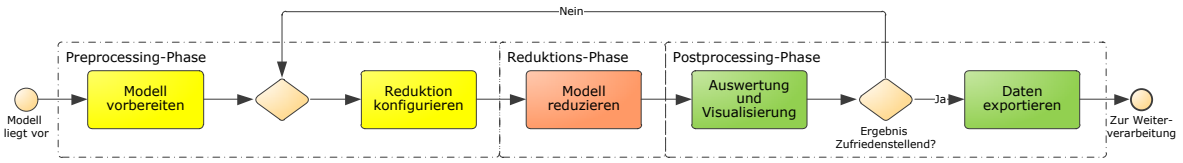

Abbildung 3.1: Grob unterteilter Prozess zur Modellreduktion

tatsächlichen Implementierung des Workflows ist also zu klären, in welche Einzelschritte der Prozess sinnigerweise unterteilt werden sollte. Dabei sind verschiedene Aspekte zu betrachten:

- Die Aufteilung der Aufgaben auf verschiedene Workflowteilnehmer. Dazu zählen neben den unterschiedlichen Personen, die an der Bearbeitung des Workflows beteiligt sind, auch auf verschiedenen Rechnern laufende Webservices oder Legacy-Anwendungen. In einem orchestrierten Workflow werden die Aktivitäten von einzelnen Teilnehmern ausgeführt, Kommunikation findet in der Regel zwischen der Workflowengine und den einzelnen Teilnehmern statt. Aktivitäten, die von verschiedenen Workflowteilnehmern ausgeführt werden, sollten also voneinander getrennt werden.
- Die Steuerung des Kontrollflusses. Die Stellen im Prozess, an denen der Kontrollfluss unterteilt, zusammengeführt oder abhängig von Bedingungen oder Ereignissen auf unterschiedliche Stränge geleitet wird, müssen von den restlichen Aktivitäten getrennt werden, um der Engine die gezielte Ablaufsteuerung zu ermöglichen.
- Mögliche spätere Veränderungen der Workflowlogik oder einzelner Teile des Workflows. Zum Beispiel sollte der Import von Eingangsdaten in eine separate Aktivität ausgelagert werden, wenn sich das Datenformat oder die Datenquelle später ändern könnten
- Eine mögliche spätere Umverteilung der Aufgaben auf die Workflowteilnehmer oder Änderungen der verwendeten Ressourcen. Zum Beispiel ist die Isolierung von rechenintensiven Aufgaben sinnvoll, wenn diese später auf spezialisierte Hardware ausgelagert oder von effizienteren Programmen bearbeitet werden könnten.
- Die Wiederverwendbarkeit einzelner Dienste. Wurden für die Bearbeitung bestimmter Teilaufgaben bereits Dienste implementiert oder lassen sich Teile des Prozesses für andere Aufgaben wiederverwenden, sollten diese als eigenständige Workflow-Schritte behandelt werden.

Abbildung 3.2 zeigt eine mögliche, praxisnahe Zerlegung des Workflows in Einzelschritte. Die Bearbeitung der einzelnen Schritte wird zwischen Matlab und dem Wissenschaftler, der den Workflow startet und kontrolliert, aufgeteilt. Letzterer wird im folgenden als *Benutzer* bezeichnet. Der Pool, der den durch Matlab zu bearbeitenden Prozess darstellt, ist in zwei Teile, dargestellt durch Lanes, aufgeteilt. Ein Teil dient dem Import und Export der Daten, während der andere die eigentliche Modellreduktion in der Reduktionsphase übernimmt. Diese Aufteilung ist notwendig, um die Reduktionsphase innerhalb eines Ablaufs des kompletten Prozesses wiederholt instanziieren zu können, bis die Qualität des Ergebnisses ausreicht. Würde ein solcher Iterationsvorgang in einem einzigen Teilworkflow implementiert, wäre eine sehr komplexe Schleife notwendig, da die Abbruchbedingung von einer Benutzerentscheidung, also

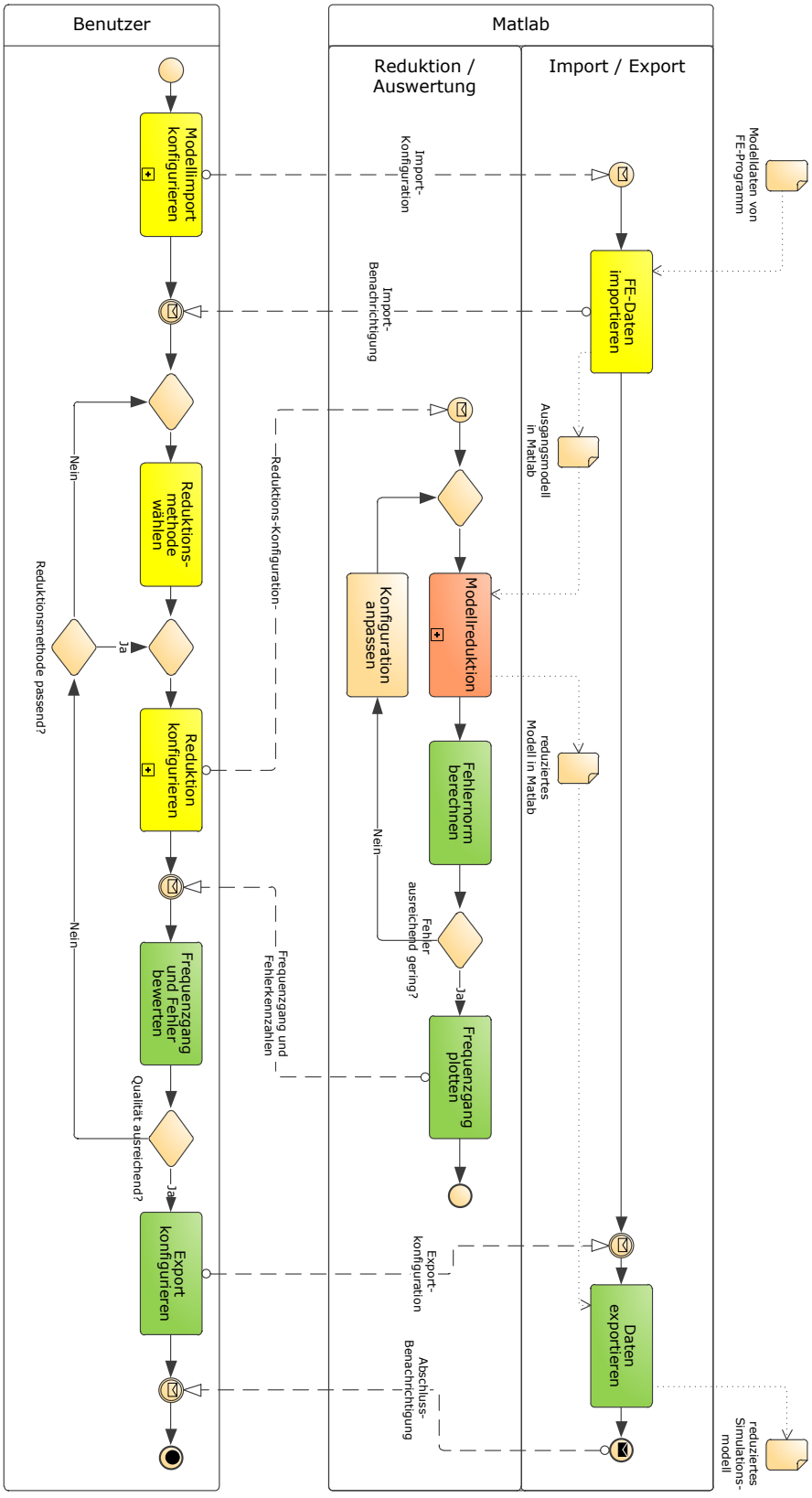

Abbildung 3.2: Ausführlicher Prozess zur Modellreduktion

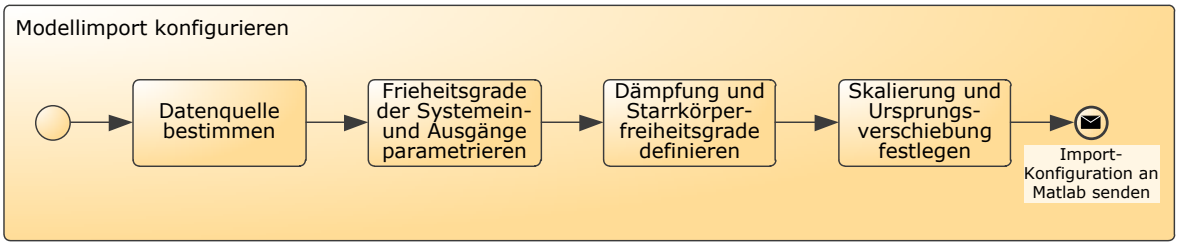

Abbildung 3.3: Teilworkflow zur Konfiguration des Modellimports („Modellimport konfigurieren“)

von einem externen Ereignis abhängig wäre. Die Details des Iterationsvorgangs werden im Abschnitt 3.4 näher erläutert.

3.2 Import und Vorbereitung der Daten

Um das Ausgangsmodell mit Matlab reduzieren zu können, müssen die Modelldaten in geeigneter Form im Matlab-Workspace vorhanden sein. Zur Diskretisierung des Modells eines elastischen Mehrkörpersystems wird in der Regel allerdings spezialisierte Finite-Elemente-Software verwendet, welche die erzeugten Gitter in unterschiedlichen Formaten speichert. Diese müssen von Matlab eingelesen und dann unter Umständen noch in die, für die Durchführung der Modellreduktion notwendige Form gebracht beziehungsweise um fehlende Informationen ergänzt werden.

Vor dem Datenimport muss der Benutzer die Datenquelle festlegen sowie einige Parameter des Import-Vorgangs konfigurieren. Dazu zählen die Ein- und Ausgänge des zu simulierenden Systems. Diese entsprechen in der Regel den Freiheitsgraden der Knoten im FE-Gitter, welche mit anderen Körpern im elastischen Mehrkörpersystem verbunden sind, man spricht hier deshalb auch von *Koppelknoten*. Außerdem werden für die spätere Reduktion des Modells Informationen zur Systemdämpfung und zu möglichen Starrkörperbewegungen sowie die Parameter einer gegebenenfalls durchzuführenden Skalierung und Verschiebung des Ursprungs benötigt. Abbildung 3.3 stellt den für diesen Schritt notwendigen Teilworkflow grafisch dar.

Auf den internen Aufbau der Import-Aktivität(„FE-Daten importieren“) soll hier nicht näher eingegangen werden, da er vom jeweiligen verwendeten Ausgangsformat abhängt und für die eigentliche Ausführung der Modellreduktion nebensächlich ist. Es sei jedoch darauf hingewiesen, dass der Modell-Import in jedem Fall als eigenständiger Workflow-Schritt behandelt werden sollte, um flexibel auf spätere Änderungen des Ausgangsformats reagieren zu können.

Nachdem das zu reduzierende Modell bereitgestellt wurde, folgt die Vorbereitung der eigentlichen Modellreduktion. Die hierzu notwendigen Aktivitäten hängen vom verwendeten Reduktionsverfahren ab, welches also zuerst festgelegt werden muss. Die Wahl zwischen modaler Reduktion, Krylov-Verfahren oder einer anderen Reduktionsmethode kann entweder zur Build-Time oder zur Run-Time des Workflowmodells erfolgen. Bei der Wahl zur Build-Time wird das verwendete Modellreduktionsverfahren fest im Workflowmodell hinterlegt. Dies hat den Vorteil

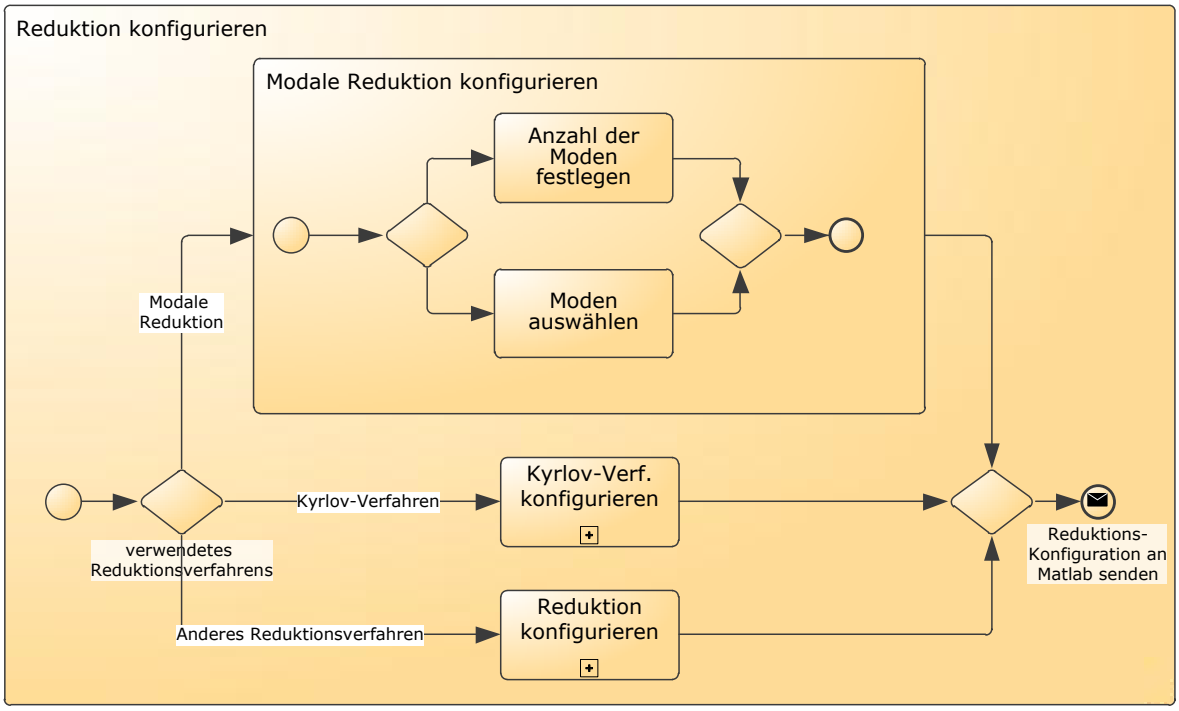

Abbildung 3.4: Teilworkflow zur Konfiguration des Modellreduktion („Reduktion konfigurieren“)

eines einfacheren Workflows, da keine Nutzerinteraktion zur Wahl des Reduktionsverfahrens notwendig ist. Allerdings muss das Workflow-Modell in diesem Fall bei jedem Wechsel des Verfahrens geändert und der Workflowengine erneut bereitgestellt (*deployed*) werden, was mit zusätzlichem Aufwand verbunden ist und entsprechende Kenntnisse und Berechtigungen des Benutzers voraussetzt. Für die Wahl des Reduktionsverfahrens zur Run-Time spricht deshalb die höhere Flexibilität. Der Nutzer kann für jede Workflowinstanz erneut entscheiden, welches Reduktionsverfahren verwendet werden soll. Die Entscheidung kann sogar in einen iterativen Prozess eingebunden werden, der die Änderung des Reduktionsverfahrens und die anschließende, erneute Durchführung der Reduktions-Phase bei Unzufriedenheit mit dem Ergebnis erlaubt. In dem, in Abbildung 3.2 dargestellten Workflow wird der Benutzer deshalb zur Run-Time nach dem zu verwendenden Reduktionsverfahren befragt. Würde das Reduktionsverfahren dagegen zur Build-Time festgelegt, entfiele der Workflowschritt zur Wahl des Verfahrens und damit auch die Möglichkeit, das Reduktionsverfahren in einer Schleife anzupassen.

Nach der Festlegung des Reduktionsverfahrens folgt die weitere Konfiguration und Parametrierung der Modellreduktion durch den Benutzer. Von diesen hängt unter anderem die Dimension ab, auf die das Ausgangssystem reduziert wird. Die genauen Parameter der Modellreduktion unterscheiden sich je nach verwendetem Reduktionsverfahren. Abbildung 3.4 zeigt den für die Parametrierung zu implementierenden Teilworkflow, wobei hier nur die Konfiguration des in Abschnitt 2.4.3 vorgestellten modalen Reduktionsverfahrens im Detail dargestellt wird. Die MatMorembs-Implementierung dieses Verfahrens erwartet wahlweise die gezielte Auswahl der

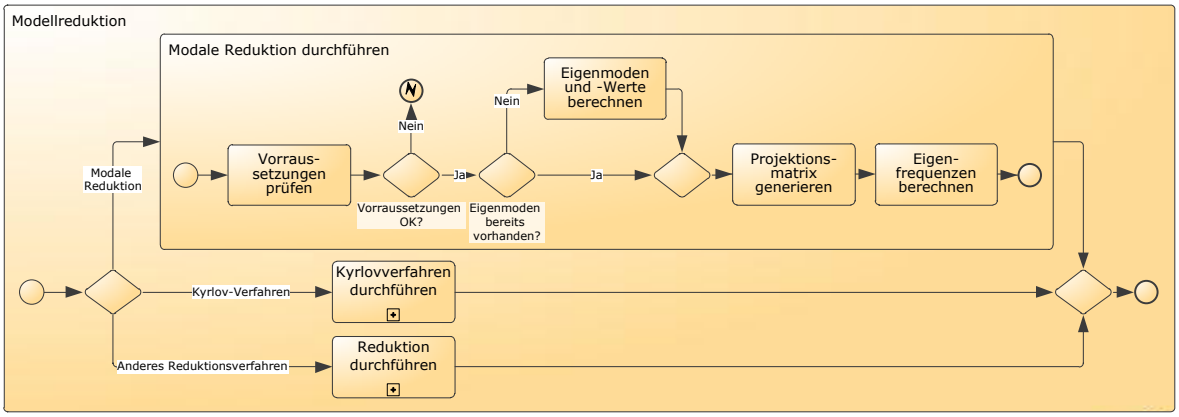

Abbildung 3.5: Vereinfachter Teilworkflow zur Durchführung der Modellreduktion („Modellreduktion“)

Moden, die bei der Modellreduktion behalten werden sollen, oder einfach die Angabe der Zahl m der zu berücksichtigenden Moden. Im zweiten Fall werden bei der Reduktion die m Moden der niedrigsten Frequenz behalten, da das durch diese Moden charakterisierte Verhalten meistens besonders relevant ist.

3.3 Durchführung der Modellreduktion

Nach der Bereitstellung aller notwendigen Daten und Parameter in der Preprocessing-Phase folgt die eigentliche Modellreduktion. Diese erfordert keine weitere Nutzerinteraktion, kann aber, bei einer entsprechend hohen Dimension des zu reduzierenden Systems, einige Zeit in Anspruch nehmen.

Beim Entwurf des Workflows muss geklärt werden, ob eine weitere Unterteilung dieses Reduktionsschritts sinnvoll ist. Das entscheidende Kriterium hierbei ist die Aufteilbarkeit der Rechenoperationen auf verschiedene Workflowteilnehmer. Teiloperationen, für die die Verwendung spezialisierter Hardware oder Software in Frage kommt, sollten als eigene Aktivitäten behandelt werden. Das gleiche gilt für Teiloperationen, die auf verschiedenen Strängen des vom Workflowkoordinator gesteuerten Kontrollflusses liegen und deshalb getrennt aufgerufen werden können sollten.

Abbildung 3.5 zeigt, stark vereinfacht, den inneren Aufbau des Teilworkflows zur Durchführung der Modellreduktion, der vom verwendeten Reduktionsverfahren abhängt. Hier wird nur der Ablauf der modalen Reduktion ausführlich dargestellt, während der Ablauf der alternativen Verfahren in geschlossenen Teilworkflows verpackt und hier nicht weiter erörtert wird.

Bei der Durchführung der modalen Reduktion muss zuerst geprüft werden, ob das zu reduzierende Modell und die Konfiguration des Reduktionsverfahrens die Voraussetzungen erfüllt. Leere Systemmatrizen oder die fehlende Angabe der zu berücksichtigenden Moden führen an dieser Stelle zum Auslösen einer Fehlers und zum Abbruch des Prozesses. Dieser Fehler sollte

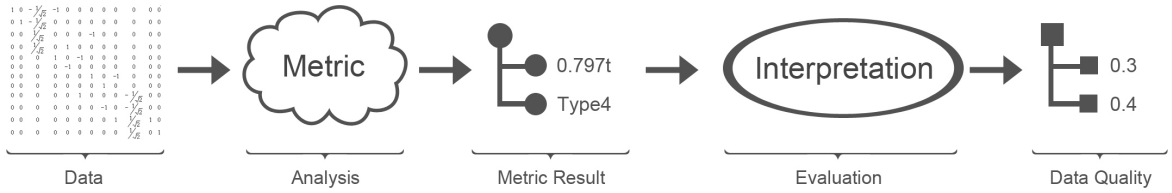

Abbildung 3.6: Berechnung der Datenqualität ([Bre11])

natürlich vom übergeordneten Workflow abgefangen und entsprechend behandelt werden, z.B. indem der Benutzer über den Fehler informiert und zum korrigieren der Eingabedaten oder der Konfiguration aufgefordert wird. Die Implementierung einer solchen Fehlerbehandlung ist nicht Teil dieser Arbeit, wird allerdings in Abschnitt 4.4.4 als Erweiterung vorgeschlagen.

Anschließend wird überprüft, ob die vorhandenen Modelldaten bereits die Eigenmoden des Systems enthalten. Ist dies nicht der Fall, müssen diese in einer vergleichsweise rechenaufwändigen Operation bestimmt werden. Für die Durchführung dieser Rechenoperationen existieren parallelisierbare Algorithmen, die auf spezieller Hardware wie Parallelrechnern, verteilten Rechnersystemen oder CUDA[1]-fähigen Grafikprozessoren (vgl. [Sim08]) ausgeführt werden könnten.

Die Eigenmoden des Systems werden verwendet, um die Projektionsmatrix des Systems wie vom Benutzer konfiguriert zusammenzustellen. Abschließend werden dann aus den Eigenwerten die Eigenfrequenzen des reduzierten Systems berechnet.

3.4 Auswertung und Visualisierung

Bei der Reduktion der Dimension eines mathematischen Modells geht immer ein Teil des charakteristischen Modellverhaltens verloren. Bei den hier betrachteten Modellen elastischer Mehrkörpersysteme äußert sich das dadurch, dass bestimmte dynamische Verformungen des realen Systems in der Simulation auf dem reduzierten Modell nicht mehr beobachtet werden können. Es ist im Allgemeinen nicht möglich, vor der Reduktion vorherzusagen, ob das reduzierte Modell das interessierende Verhalten des realen Systems mit ausreichender Genauigkeit nachbilden kann und ob die Qualität des reduzierten Modells den Anforderungen der anschließenden Simulation genügt. Auf eine anschließende Auswertung und Beurteilung des reduzierten Modells kann also nicht verzichtet werden.

In [Bre11] wird eine allgemeine Methode vorgestellt, die Qualität von Daten zu untersuchen (vgl. Abbildung 3.6). Zuerst werden die Daten demnach durch entsprechende Metriken wertungsneutral auf bestimmte Eigenschaften und Charakteristika untersucht. Am Ende dieser Analyse stehen Metrikergebnisse, die zwar als Indikator dafür dienen können, ob die Daten für einen bestimmten Zweck geeignet sind, jedoch in keinem spezifischen Kontext stehen und

[1]Compute Unified Device Architecture (http://developer.nvidia.com/category/zone/cuda-zone)

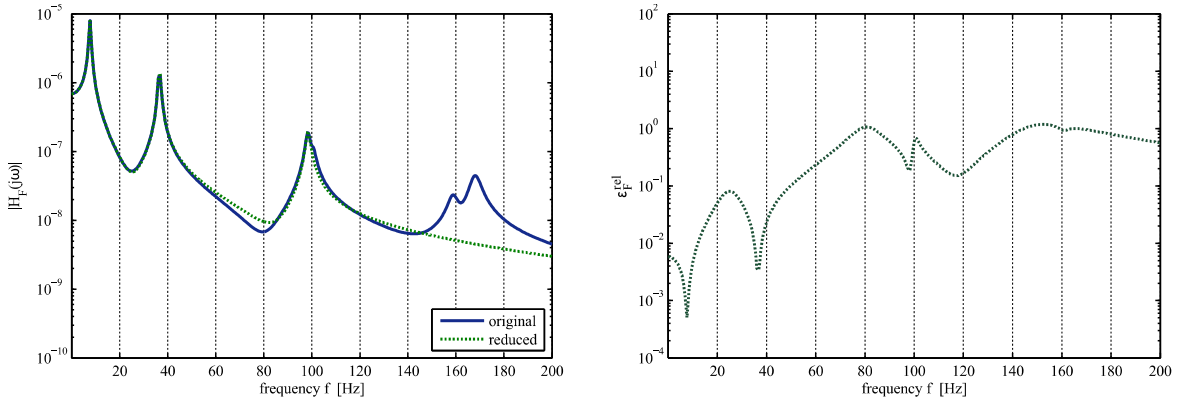

Abbildung 3.7: Frequenzgang des Systemverhaltens (links) und relativer Fehler (rechts)

keine Wertung der Datenqualität darstellen. Beispiele hierfür sind die verschiedenen Normen einer Matrix, Angaben über die Vollständigkeit eines Datensatzes oder die Standardabweichung einer Wahrscheinlichkeitsverteilung. Anschließend erfolgt die Evaluation und Bewertung dieser Metrikergebnisse im Kontext der Anforderungen an die untersuchten Daten. Dabei wird ein vergleichbarer Wert ermittelt, der Auskunft darüber erteilt, wie gut die untersuchten Daten für die jeweilige Anwendung geeignet sind. Sowohl die Datenanalyse als auch die kontextabhängige Evaluation der Ergebnisse können prinzipiell objektiv durch idempotente Algorithmen oder subjektiv durch menschliche Entscheidungen erfolgen.

Metriken zur Analyse der reduzierten Modelldaten

Methoden zur zuerst erfolgenden, wertungsneutralen Analyse der Modellreduktion werden in [Feh11] beschrieben. Zunächst werden das Originalsystem $\boldsymbol{H}(s)$ (Gleichungen 2.1) und das entsprechende reduzierte System $\bar{\boldsymbol{H}}(s)$ im Frequenzraum betrachtet:

$$\boldsymbol{Y}(s) = \boldsymbol{H}(s) \cdot \boldsymbol{U}(s) \tag{3.1}$$

$$\bar{\boldsymbol{Y}}(s) = \bar{\boldsymbol{H}}(s) \cdot \boldsymbol{U}(s) \tag{3.2}$$

Zur grafischen Darstellung des Frequenzverhaltens der beiden Systeme im interessierenden Frequenzbereich können Matrixnormen wie die Frobeniusnorm oder die Spektralnorm verwendet werden. Im linken Teil der Abbildung 3.7 werden die frobeniusnormierten Frequenzverläufe eines Beispielsystems in seiner ursprünglichen und in seiner reduzierten Form dargestellt.

Zur weiteren Untersuchung des durch die Modellreduktion entstandenen Fehlers wird das Fehlersystem betrachtet, das sich aus der Differenz des ursprünglichen und des reduzierten Systems ergibt:

$$\boldsymbol{H}_E(s) = \boldsymbol{H}(s) - \bar{\boldsymbol{H}}(s) \tag{3.3}$$

Mit diesem Fehlersystem kann der relative Fehler der Modellreduktion unter Verwendung der Frobenius-Norm (Gleichung 3.4) oder der Spektralnorm (Gleichung 3.5) errechnet werden:

$$\epsilon_F^{rel}(f) = \frac{\|\boldsymbol{H}_E(i2\pi f)\|_F}{\|\boldsymbol{H}(i2\pi f)\|_F} = \frac{\|\boldsymbol{H}(i2\pi f) - \bar{\boldsymbol{H}}(i2\pi f)\|_F}{\|\boldsymbol{H}(i2\pi f)\|_F}, \qquad f \in [f_{min}, f_{max}] \tag{3.4}$$

$$\epsilon_2^{rel}(f) = \frac{\|\boldsymbol{H}_E(i2\pi f)\|_2}{\|\boldsymbol{H}(i2\pi f)\|_2} = \frac{\|\boldsymbol{H}(i2\pi f) - \bar{\boldsymbol{H}}(i2\pi f)\|_2}{\|\boldsymbol{H}(i2\pi f)\|_2}, \qquad f \in [f_{min}, f_{max}] \tag{3.5}$$

Dieser relative Fehler kann nun ebenfalls geplottet werden. Im rechten Teil der Abbildung 3.7 wird der Frequenzverlauf des ϵ_F^{rel}-Fehlers des reduzierten Beispielsystems dargestellt.

Zudem existieren verschiedene Verfahren zum quantitativen Vergleich des reduzierten System mit dem Originalsystem anhand einer skalaren Größe, also einer einzelnen, frequenzunabhängigen Kennzahl. Eine Möglichkeit ist die Verwendung der frequenzgewichteten $\mathcal{H}_\infty^i$-Norm des Fehlersystems, die im wesentlichen den größten entstehenden Fehler im interessierenden Frequenzbereich angibt:

$$\|\boldsymbol{H}_E\|_{\mathcal{H}_\infty^i} = \sup_{f \in [f_{min}, f_{max}]} \|\boldsymbol{H}_E(f)\|_2 \tag{3.6}$$

Außerdem kann die gewichtete $\mathcal{H}_2^i$-Norm verwendet werden, die als Integral der Frobeniusnorm des Fehlers über den interessierenden Frequenzbereich bestimmt wird:

$$\|\boldsymbol{H}_E\|_{\mathcal{H}_2^i} = \left(\int_{f_{min}}^{f_{max}} \|\boldsymbol{H}_E(i2\pi f)\|_F^2 df \right)^{\frac{1}{2}} \tag{3.7}$$

Evaluation der Metrikergebnisse

Die Visualisierungen des Frequenzgangs des Systems und des relativen Fehlers (Abbildung 3.7) eignen sich besonders zur Interpretation durch einen Menschen. Die $\mathcal{H}_\infty^i$ und $\mathcal{H}_2^i$-Normen sind dagegen als skalare Metriken für eine automatisierte Evaluation durch ein Computerprogramm prädestiniert. In dem hier vorgestellten Workflow (Abbildung 3.2) werden daher beide Möglichkeiten kombiniert und zum Steuern des weiteren Prozessflusses verwendet.

Nach der Berechnung der skalaren Metriken (Gleichungen 3.6 und 3.7) durch Matlab folgt zunächst deren automatisierte Evaluation. Dabei wird aus den skalaren Metrikergebnissen ein Datenqualitätskennwert Q berechnet:

$$Q = Q(\|\boldsymbol{H}_E\|_{\mathcal{H}_2^i}, \|\boldsymbol{H}_E\|_{\mathcal{H}_\infty^i}), \qquad Q \in [0..1] \tag{3.8}$$

Die Berechnungsvorschrift für Q ist nicht fest vorgegeben. Die Gewichtung der einzelnen Fehlernormen kann an die Anforderungen an das reduzierte Modell angepasst werden. Allerdings sollte $Q(\|\boldsymbol{H}_E\|_{\mathcal{H}_2^i}, \|\boldsymbol{H}_E\|_{\mathcal{H}_\infty^i})$ monoton fallen, d.h. die Datenqualität sollte bei größer werdenden

Fehlernormen abnehmen. Bei der in Kapitel 4 dieser Arbeit vorgestellten Implementierung wird der Datenqualitätskennwert mit dem von 1 subtrahierten geometrischen Mittel der Fehlernormen berechnet:

$$Q = 1 - \sqrt{\|\boldsymbol{H}_E\|_{\mathcal{H}_2^i} \cdot \|\boldsymbol{H}_E\|_{\mathcal{H}_\infty^i}} \tag{3.9}$$

Anschließend wird die Datenqualität mit der minimalen geforderten Datenqualität Q_{min} verglichen. Wenn die Datenqualität nicht den Anforderungen entspricht ($Q < Q_{min}$), muss die Modellreduktion mit angepasster Konfiguration wiederholt werden. Zum Beispiel könnte die Zahl der bei der Reduktion zu berücksichtigenden Moden m in Abhängigkeit der Differenz zwischen der geforderten und der tatsächlichen Datenqualität erhöht werden:

$$m_{j+1} = m_j + f(Q_{min} - Q_j) \tag{3.10}$$

Hier steht $j \in \mathbb{N}$ für den jeweiligen Iterationsschritt und $f : \mathbb{R} \mapsto \mathbb{N}$ für eine beliebige Funktion, die die Anzahl der zusätzlich zu berücksichtigenden Moden berechnet.

Wird die Datenqualitätsanforderung in diesem Schritt erfüllt, erfolgt im nächsten Schritt die Bewertung der Ergebnisse der Modellreduktion durch den Benutzer. Dazu werden die oben beschriebenen Frequenzgänge des Originalsystems, des reduzierten Systems und des relativen Fehlers durch Matlab generiert und anschließend dem Benutzer zur Interpretation und Beurteilung präsentiert. Abhängig vom Ergebnis dieser Beurteilung kann der Benutzer die Modellreduktion mit anderen Einstellungen wiederholen oder das Ergebnis der Reduktion bestätigen und mit dem anschließenden Datenexport fortfahren.

Falls der Benutzer die Qualität des reduzierten Modells nicht als ausreichend erachtet, muss er zunächst entscheiden, ob das Reduktionsverfahren gewechselt oder lediglich die Konfiguration des aktuellen Verfahrens angepasst werden soll. Nach der entsprechenden Anpassung der Konfiguration wird der komplette, iterative Reduktionsvorgang und die Analyse des reduzierten Systems erneut ausgeführt. Da dieser Teilprozess als eigenständiger Workflow realisiert ist, kann er dazu einfach neu instanziiert werden.

3.5 Datenexport

Nach der Modellreduktion stehen im Matlab-Workspace die Standarddaten[2] des reduzierten Modells zur Verfügung. Diese können zur weiteren Untersuchung und Simulation des Systems in Matlab verwendet werden. Häufig soll die Simulation und die anschließende Auswertung des reduzierten Systems aber in einem externen Simulations- oder FE-Programm durchgeführt werden. Der dazu notwendige Export der erzeugten Daten muss zuerst vom Benutzer konfiguriert werden. Abhängig vom gewählten Exportformat müssen hierzu neben dem Speicherort der zu erzeugenden Dateien noch weitere Parameter wie eventuell benötigte Skalierungs- und Verschiebungsfaktoren gesetzt werden.

[2]Die Standarddaten eines elastischen Körpers sind die Daten, welche notwendig sind um den Körper im mitbewegten Koordinatensystem zu simulieren. Der Umfang dieser Daten wird in [Wal94] definiert.

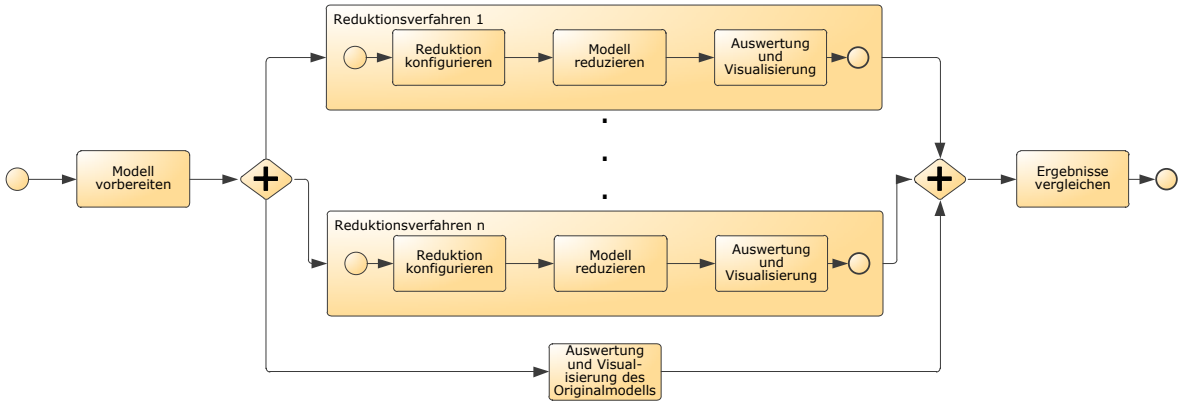

Abbildung 3.8: Workflow zum Vergleich verschiedener Reduktionsverfahren

3.6 Alternative Workflows

Die SOA- und Webservicetechnologie erlaubt eine einfache Orchestrierung neuer Workflows aus bereits vorhandenen Diensten. So lassen sich alternative Prozesse durch Zusammensetzen der einmal erstellten Dienste mit relativ geringem Aufwand als Workflow implementieren.

Zum Beispiel lässt sich das Konzept des bisher vorgestellten Workflows so verändern, dass neben den FE-Gittern auch andere Simulationsmodelle reduziert werden können. Dazu müssen im wesentlichen die Aktivitäten für den Import und den Export der Modelle und die Konfiguration dieser Schritte so angepasst werden, dass die entsprechenden Modelle verarbeitet werden können.

Eine andere mögliche Aufgabenstellung aus dem Bereich der Modellreduktion ist der Vergleich verschiedener Reduktionsverfahren anhand von Beispielmodellen. Dazu muss das verwendete Beispielmodell importiert und dann mit verschiedene Verfahren reduziert werden. Im Anschluss erfolgt der Vergleich der verschiedenen Ergebnisse durch manuelle oder computergestützte Auswertungsverfahren.

Abbildung 3.8 zeigt einen entsprechenden Workflow, der im Wesentlichen aus den bereits vorgestellten Aktivitäten zusammengestellt werden kann. Die Konfigurations- Reduktions- und Auswertungsschritte werden dabei für die verschiedenen Reduktionsverfahren parallel ausgeführt.

Möglich ist auch die Durchführung einer Simulation unter beispielhaften Anfangs- und Randbedingungen auf dem, durch verschiedene Verfahren reduzierten Modell und auf dem nicht reduzierten Originalmodell und der anschließende Vergleich der Simulationsergebnisse.

4 Implementierung des Workflows

In diesem Kapitel wird die Implementierung eines Workflows zur Reduktion der elastischen Freiheitsgrade eines Simulationsmodells vorgestellt. Ausgangspunkt dafür ist das Matlab-basierte Modellreduktionswerkzeug MatMorembs (siehe Abschnitt 2.5.2) welches bereits alle nötigen Rechenoperationen zum Import der FE-Daten des Ausgangsmodells, zur Durchführung und Auswertung der Modellreduktion und zum Export des reduzierten Modells in verschiedene Formate beherrscht.

Der Workflow wird in WS-BPEL (siehe Abschnitt 2.2.2) implementiert und dann auf Basis von Apache ODE (siehe Abschnitt 2.5.3) ausgeführt. Vor der eigentlichen Implementierung müssen einige Fragen geklärt werden, insbesondere muss ein Konzept für die Verwaltung der Arbeitsdaten erstellt und eine Möglichkeit zur Workflowanbindung von Matlab gefunden werden. Dieses Kapitel geht in den Abschnitten 4.1 respektive 4.2 auf diese Fragen ein. Anschließend werden in Abschnitt 4.3 technische Details zur BPEL-Implementierung erläutert und in Abschnitt 4.4 mögliche Erweiterungen dieser Implementierung vorgeschlagen.

4.1 Datenmanagement

Die verschiedenen Schritte der Modellreduktion werden in MatMorembs üblicherweise in einer zusammenhängenden Matlab-Sitzung ausgeführt. Sämtliche Arbeitsdaten, also die Modelldaten sowie die Ergebnisse bereits ausgeführter Teiloperationen, befinden sich dabei im Matlab-Workspace und können von den MatMorembs-Funktionen genutzt werden.

Bei der Implementierung als BPEL-Workflow werden die MatMorembs-Operationen entsprechend dem in Kapitel 3 vorgestellten Konzept in Workflowschritte aufgeteilt. Dabei kann nicht gewährleistet werden, dass die verschiedenen Aktivitäten einer Workflowinstanz in einer zusammenhängenden Matlab-Sitzung ausgeführt werden. Es muss also eine geeignete Methode gefunden werden, um den einzelnen in Matlab ausgeführten Workflowschritten die jeweils benötigten Arbeitsdaten zur Verfügung zu stellen, wobei die Arbeitsdaten parallel laufender Modellreduktionsinstanzen strikt voneinander zu trennen sind.

Dieses Problem wurde mit den Matlab-Funktionen `save` und `load` gelöst, mit denen sämtliche Daten des Matlab-Workspace als Datei exportiert und in einer anderen Matlab-Sitzung wieder eingelesen werden können. Diese Funktionen können dazu genutzt werden, die Arbeitsdaten von einer in Matlab ausgeführten Workflow-Aktivität zur nächsten zu übertragen. Für die Datenübertragung zwischen den in Matlab ausgeführten Operationen bietet sich der Export im proprietären .mat-Format an, da damit sämtliche Matlab-Datentypen verlustfrei serialisiert werden können. Werden die Daten hingegen in einem anderen Programm weiterverarbeitet, ist

der Export der entsprechenden Daten in CSV-Dateien in der Regel zweckmäßiger, da diese in den meisten Programmiersprachen ohne großen Aufwand eingelesen werden können.

Die Trennung der Arbeitsdaten parallel laufender Modellreduktionsinstanzen kann durch das Generic Web Service Interface (siehe Abschnitt 2.2.3) gewährleistet werden. Die aus Matlab exportierten Daten werden dabei im Dateisystem in separaten Verzeichnissen abgelegt, die eindeutig mit einer Instanz-ID verknüpft sind. Vom WSI ausgeführte Operationen verlangen stets die Angabe einer Instanz-ID und greifen nur auf Dateien zu, die sich innerhalb des mit dieser ID verknüpften Verzeichnisses befinden. Somit werden Konflikte vermieden, die durch die Mischung der Arbeitsdaten parallel laufender Modellreduktionsinstanzen entstehen können.

Es ist zu beachten, dass die so verwalteten Daten zunächst nur auf dem Computer zur Verfügung stehen, auf dem das Web Service Interface läuft. Wenn verschiedene Workflowteilnehmer oder die Workflowengine auf die Arbeitsdaten zugreifen sollen, müssen die Daten über ein geeignetes Netzwerkprotokoll übertragen werden. Alternativ könnte hier eine Datenbank in Verbindung mit dem in [RRS+11] vorgestellten SIMPL-Framework eingesetzt werden. Im Rahmen dieser Arbeit wurde der Workflow allerdings so implementiert, dass alle Matlab-Operationen auf dem selben Rechner ausgeführt werden, so dass auf die Übertragung großer Datenmengen verzichtet werden kann.

4.2 Bereitstellung von Matlab-Funktionen als Webservice

Für die Implementierung eines BPEL-Workflows zur Modellreduktion auf Basis von MatMorembs müssen die entsprechenden Matlab-Funktionen als Webservice zur Verfügung stehen. Da Matlab in der genutzten Version keine direkte Möglichkeit zur Bereitstellung von Webservices bietet, müssen die in Kapitel 2.5.1 beschriebenen externen Schnittstellen dazu genutzt werden, die Matlab-Funktionen als Webservice bereitzustellen. Dazu wird wiederum auf das Web Service Interface zurückgegriffen, welches verschiedene Möglichkeiten zur Anbindung solcher Legacy-Anwendungen unterstützt (vgl. Kapitel 2.2.3). Im Folgenden werden einige der Möglichkeiten zur Webservice-Anbindung von Matlab vorgestellt und miteinander verglichen. Tabelle 4.1 fasst die Eigenschaften der verschiedenen Möglichkeiten dann am Ende des Abschnitt zusammen.

4.2.1 Kommandozeilenaufruf

Die Nutzung der Kommandozeilenschnittstelle ist eine sehr einfache Möglichkeit zur Steuerung von Matlab durch Drittprogramme. Die auszuführenden Funktionsaufrufe werden Matlab dabei über den Kommandozeilenparameter `-r` übergeben. Die Ein- und Ausgabe der Daten erfolgt über Dateien, die durch die Matlab-Funktionen `load` und `save` eingelesen bzw. geschrieben werden.

Das Web Service Interface unterstützt diese Art der Steuerung von Legacy-Anwendungen über die Operationen `executeProgramSync` und `executeProgramAsync` direkt. Der Unterschied zwischen synchronen und asychronen Programmaufrufen liegt im Antwortverhalten der Operationen und wird in Abschnitt 2.2.3 erklärt.

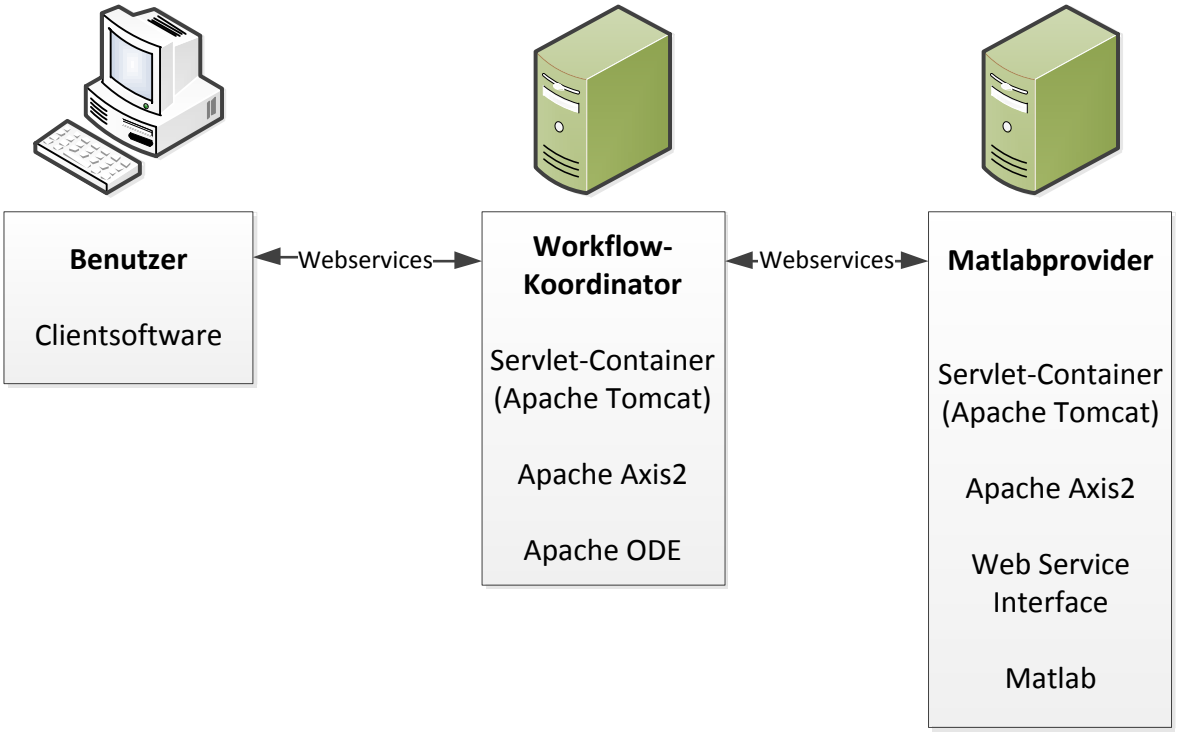

Abbildung 4.1: Rollen und benötigte Software bei der Anbindung von Matlab per Kommandozeile

Zur Umsetzung dieser Methode muss Matlab lokal auf dem Rechner verfügbar gemacht werden, auf dem das Web Service Interface läuft. Die Kommunikation zwischen diesem Rechner, im folgenden Matlab-Provider genannt, und dem Rechner auf dem die Workflow-Engine läuft, im folgenden Workflow-Koordinator genannt, erfolgt dann über Webservices. Analog kann der Benutzer über entsprechende Clientsoftware (siehe Abschnitt 4.3.3) auf die im Workflow-Koordinator laufenden Workflows zugreifen. Alle diese Komponenten können auf verschiedenen Rechnern eines Netzwerkes laufen. Die an der Ausführung des Workflows beteiligten Komponenten und die jeweils benötigte Software werden in Abbildung 4.1 schemahaft dargestellt.

Bei der Umsetzung dieser Methode muss berücksichtigt werden, dass das Web Service Interface keinen direkten Zugriff auf Programme gestattet, die außerhalb des jeweiligen Instanzverzeichnisses liegen. Da eine separate Matlab-Installation für jede Simulationsinstanz nicht praktikabel ist, muss im Instanzverzeichnis eine entsprechende Verknüpfung oder ein Script hinterlegt werden, um eine außerhalb des Instanzverzeichnisses liegende Matlab-Installation aufzurufen. In der hier vorgestellten Implementierung dient dazu ein Unix-Shellscript, welches die daran übergebenen Argumente in einem systemweiten Kontext als Befehl ausführt und somit auch außerhalb des Instanzverzeichnisses liegende Programme aufrufen kann.

Listing 4.1 zeigt beispielhaft eine SOAP-Nachricht, die an die WSI-Operationen `executeProgramSync` oder `executeProgramAsync` geschickt werden kann, um den Modellreduktionsschritt in Matlab auszuführen. Die Nachricht weist das Web Service Interface an,

Listing 4.1 Beispiel eines SOAP-Nachricht an die WSI-Operation executeProgramSync zur Ausführung des Modellreduktionsschritts in Matlab

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:typ="http://wsi.simtech.de/ws/types/">
  <soapenv:Header/>
  <soapenv:Body>
    <typ:ExecuteProgram>
                    <simID>1315230857530</simID>
                    <programPath>/shell.sh</programPath>
                    <workingDirectory/>
                    <arguments>matlab -r n_red=5;reduction;quit;</arguments>
                    <isInteractive>false</isInteractive>
    </typ:ExecuteProgram>
  </soapenv:Body>
</soapenv:Envelope>
```

die Befehlsfolge `n_red=5;reduction;quit;` in Matlab auszuführen. Durch diese wird zuerst die Matlab-Variable `n_red` mit dem Wert 5 belegt. Diese Variable enthält die Anzahl der bei der Modellreduktion zu berücksichtigenden Moden. Anschließend wird das m-File mit dem Namen `reduction.m` aufgerufen und Matlab nach dessen Ausführung beendet. Das Laden der Arbeitsdaten und das Speichern des reduzierten Modells erfolgt durch entsprechende Anweisungen im aufgerufenen m-File.

Der große Vorteil der Verwendung der Kommandozeilenschnittstelle zur Workflow-Anbindung von Matlab ist die einfache Implementierbarkeit. Bei Verwendung des Web Service Interface ist neben einem einfachen Script zum Aufruf von Matlab kein weiterer Code erforderlich. Deshalb wurde diese Lösung zur Implementierung des hier vorgestellten BPEL-Workflows genutzt.

4.2.2 SSH

Bei der Nutzung von Matlab muss dessen proprietäre Lizenz berücksichtigt werden, die dem Einsatz des Programms unter bestimmtem Umständen enge Grenzen setzt. Die im letzten Abschnitt vorgestellte Möglichkeit zur Bereitstellung von Matlab als Webservice setzt voraus, dass Matlab und ein Servlet-Container mit installiertem Axis2 und dem Web Service Interface auf dem selben Computer laufen. Die Einrichtung einer solchen Matlab-Ausführungsumgebung erfordert entweder den Erwerb einer entsprechenden Matlab-Lizenz oder die Installation zusätzlicher Software auf einem Rechner, auf dem bereits eine lizenzierte Matlab-Version verfügbar ist.

Dies kann vermieden werden, wenn über ein geeignetes Netzwerkprotokoll auf eine bereits vorhandene Matlab-Installation zugegriffen werden kann. Geeignet ist hier jedes Protokoll, das die ferngesteuerte Ausführung von Programmen auf dem Server sowie die Übertragung von Dateien zwischen Server und Client ermöglicht. Für diese Arbeit kommt die Verwendung der *Secure Shell (SSH)* in Frage, da SSH-Server im Bereich der Universität Stuttgart auf vielen Servern und Workstations laufen, auf denen auch Matlab installiert ist.

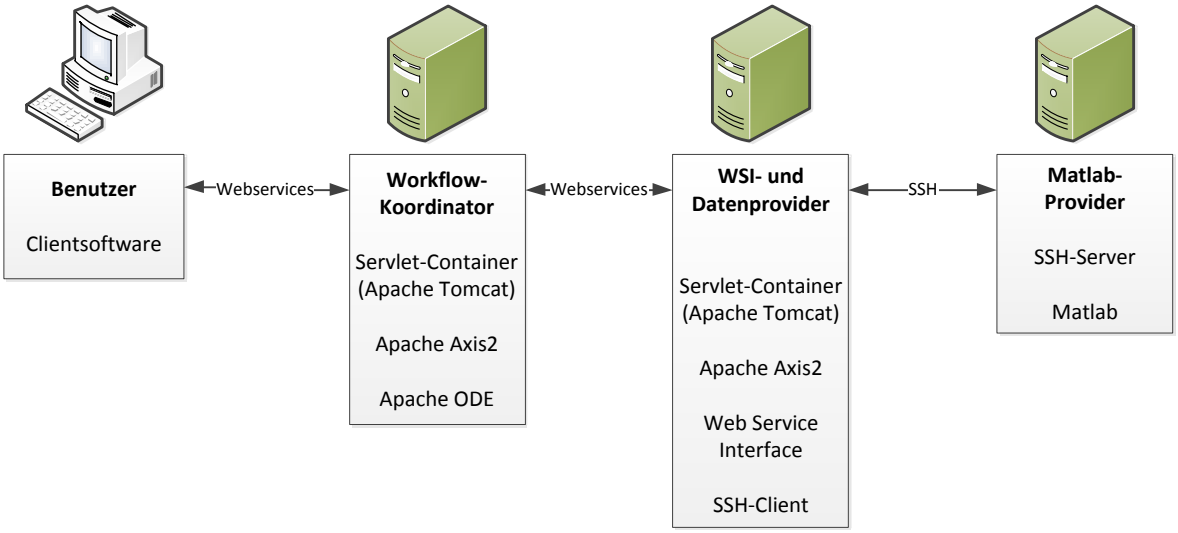

Abbildung 4.2: Rollen und verwendete Software bei der Anbindung von Matlab über SSH

Bei der Verwendung von SSH zur Workflow-Anbindung von Matlab wird das Web Service Interface weiterhin zur Verwaltung der Arbeitsdaten parallel laufender Simulationsinstanzen und zur Bereitstellung der Webservice-Funktionalität benötigt. Allerdings muss Matlab nicht auf dem selben Rechner wie das Web Service Interface installiert sein, sondern läuft auf einem fremden Rechner. Auf diesem muss neben Matlab und einem SSH-Server keine weitere Software installiert werden. In diesem Szenario wird also zwischen dem Matlab-Provider und dem WSI- und Datenprovider unterschieden (siehe Abbildung 4.2).

Bei der Ausführung eines Workflowschritts werden die benötigten Arbeitsdaten per SSH vom WSI- und Datenprovider an den Matlab-Provider übertragen und dort im Dateisystem abgelegt. Die auszuführenden Matlab-Operationen werden dann über SSH gestartet, wobei der eigentliche Aufruf von Matlab dann wie im letzten Abschnitt beschrieben über die Kommandozeilenschnittstelle erfolgt. Anschließend werden die Ergebnisse der Operationen wiederum in das auf dem WSI- und Datenprovider liegende Instanzverzeichnis übertragen. Eine Implementierung dieser Methode, Matlab als Webservice bereitzustellen, wird in [Dor11] beschrieben.

4.2.3 Pipes, COM und Matlab Engine

Der größte Nachteil der bisher vorgestellten Möglichkeiten zur Workflow-Anbindung von Matlab ist deren fehlende Interaktivität: Matlab wird beim Aufruf der entsprechenden Webservice-Operation gestartet und arbeitet dann die übergebenen Befehle ab, ohne dem Client eine Möglichkeit zur weiteren Einflussnahme anzubieten. Das spätere Fernsteuern einmal gestarteter Matlab-Instanzen ist somit nicht möglich. Falls die Matlab-Operationen auf mehrere Workflowschritte aufgeteilt werden, muss für jeden Schritt eine neue Matlab-Instanz gestartet und die Arbeitsdaten erneut importiert und nach der Operation ggf. für nachfolgende Schritte exportiert werden. Bei einer großen Zahl von Workflowschritten verringert dies die Performanz des Workflows erheblich und führt zu langen Ausführungszeiten.

Listing 4.2 Shellscript zur Fernsteuerung von Matlab über Unix-Pipes

```
1  #!/bin/sh
2  #create pipes and start matlab
3  mkfifo matinput
4  mkfifo matcommand
5  matlab <matinput &
6  tail -f matcommand>matinput &
7
8  #send matlab commands
9  echo "n_red=5">matcommand
10 echo "reduction">matcommand
11 echo "error_norms">matcommand
12 echo "plots">matcommand
```

Eine bessere Lösung ist in solchen Fällen die Verwendung der interaktiven Schnittstellen von Matlab. Dabei wird für jede Simulationsinstanz eine Matlab-Instanz gestartet, in welcher dann die Matlab-Operationen sämtlicher Workflowschritte ausgeführt werden. Das erneute Starten von Matlab und das Importieren der Arbeitsdaten zu Beginn jedes Workflowschritts entfallen, da die Arbeitsdaten einer Simulationsinstanz permanent im Matlab-Workspace gehalten werden. Sie müssen lediglich einmal zu Beginn des ersten Matlab-Aufrufs importiert und am Ende des letzten Matlab-Aufrufs sowie bei Bedarf als Zwischenergebnis exportiert werden.

Leider gestaltet sich die Nutzung der interaktiven Matlab-Schnittstellen etwas komplizierter und hängt vom verwendeten Betriebssystem sowie der für die Implementierung des Webservice verwendeten Programmiersprache ab. Außerdem erfordern sie, ähnlich wie die direkte Nutzung der Kommandozeilenschnittstelle, den Betrieb von Matlab weiterer, für die Webservice-Anbindung notwendigen Software auf dem gleichen Computer. Im Folgenden werden die möglichen Schnittstellen sowie die Möglichkeiten ihrer Nutzung deshalb nur kurz erklärt, für den im Rahmen dieser Arbeit implementierten Workflow wurden sie nicht genutzt.

Unix Pipes

Unter Linux und Mac OS X können Pipes genutzt werden, um Befehle an eine laufende Matlab-Instanz zu senden und die entsprechenden Matlab-Ausgaben auszulesen (vgl. Kapitel 2.5.1). Die Matlab-Instanz bleibt dabei geöffnet, solange die schreibende Seite der Befehlspipe und die lesende Seite der Ausgabepipe von einem fremden Programm geöffnet sind. Da sich Pipes unter unix-artigen Betriebssystemen wie normale Dateien nutzen lassen, ist der Zugriff auf diese Schnittstelle aus praktisch jeder Programmiersprache heraus möglich. Als Beispiel zeigt Listing 4.2 ein einfaches Shellscript, das eine Matlab-Instanz öffnet und dann nacheinander verschiedene Schritte des Modellreduktionsworkflows ausführt. Da der Zugriff auf Unix-Pipes auch von Java aus möglich ist, lässt sich diese Schnittstelle in einem WSI-Plugin nutzen. In diesem Fall kann das Web Service Interface wie oben beschrieben zur Verwaltung der Simulationsinstanzen genutzt werden.

COM-Automation Server

Unter Windows-Betriebssystemen unterstützt Matlab die COM-Technologie von Microsoft zur Interprozesskommunikation. Matlab agiert hier als Server, der von jeder Anwendung mit COM-Client-Support aus angesprochen werden kann. Die Schnittstelle ermöglicht dem Client nicht nur das Ausführen beliebiger Matlab-Anweisungen, sondern auch den lesenden und schreibenden Zugriff auf den Matlab-Workspace.

Durch die COM-Schnittstelle bietet die Verwendung des .NET-Frameworks und des *Internet Information Services (IIS)* von Microsoft eine einfache Möglichkeit, Matlab über Webservices interaktiv zu steuern. Der Webservice wird dabei in .NET auf IIS-Basis erstellt. Der Zugriff auf den von Matlab bereitgestellten COM-Automation-Server ist aus .NET heraus problemlos möglich [Gun01].

Die Nutzung der COM-Schnittstelle aus einem Java-Programm heraus, beispielsweise aus einem WSI-Plugin, gestaltet sich etwas komplizierter, da die COM-Technologie von Java nicht nativ unterstützt wird. Zur Verwendung der COM-Schnittstelle aus Java heraus existieren verschiedene freie und kommerzielle Softwareprodukte (JACOB[1], JCOM[2], EZ Jcom[3]), deren Tauglichkeit für den hier beschriebenen Anwendungsfall im Rahmen dieser Arbeit allerdings nicht evaluiert wurde.

Matlab Engine

Bei der Matlab Engine handelt es sich um eine mit Matlab mitgelieferte Programmbibliothek, welche fremder Software die interaktive Kommunikation mit einer Matlab-Instanz ermöglicht. Die Matlab Engine nutzt intern Unix-Pipes bzw. die COM-Technologie (vgl. [Mat]), bietet jedoch auf allen von Matlab unterstützten Betriebssystemen eine einheitliche Schnittstelle, welche sowohl die Ausführung beliebiger Matlab-Anweisungen als auch den Zugriff auf den Matlab-Workspace erlaubt.

Die Matlab Engine existiert in zwei verschiedenen Varianten für die Verwendung in C/C++ und in Fortran. Hier ist die C/C++-Variante der Bibliothek von besonderem Interesse, da sie in Verbindung mit der C-Implementierung von Axis2 zur interaktiven Webservice-Bereitstellung von Matlab verwendet werden kann. Die Verwendung der Bibliothek aus einem WSI-Plugin heraus gestaltet sich allerdings schwieriger, da die Bibliothek keine Schnittstelle zur direkten Verwendung aus Java-Programmen heraus bereitstellt.

[1] http://sourceforge.net/projects/jacob-project/
[2] http://sourceforge.net/projects/jcom/
[3] http://www.ezjcom.com/

	Betriebssystem	Programmier-sprachen	Interaktiv	Netzwerk-fähig[4]	Aufwand
Kommandozeilen-aufruf	Alle	Alle	Nein	Nein	gering
Kommandozeilen-aufruf über SSH	Alle mit SSH-Unterstützung	Alle	Nein	Ja	moderat
Pipes	Unix-artige	Alle	Ja	Nein	moderat
COM	Windows	Alle mit COM-Client-Support	Ja	Nein	hoch
Matlab-Engine	Alle	C/C++, Fortran	Ja	Nein	hoch

Tabelle 4.1: Verschiedene Möglichkeiten zur Bereitstellung von Matlab als Webservice im Vergleich

4.3 Implementierung des BPEL-Workflows

Die für diese Arbeit erstellte BPEL-Implementierung des Workflows zur Modellreduktion orientiert sich in den wesentlichen Punkten an dem in Kapitel 3.1 vorgestellten BPMN-Diagramm (Abbildung 3.2). Die beiden Lanes des Matlab-Pools wurden dabei als eigenständige Teilworkflows realisiert. Dies ermöglicht dem Benutzer, den für die eigentliche Modellreduktion zuständigen Teilworkflow wiederholt zu instanziieren, bis ein zufriedenstellendes Ergebnis erzielt wurde (vgl. Abschnitt 3.2). Beide Teilworkflows verwenden allerdings die selbe WSI-Simulationsinstanz zur Verwaltung der Arbeitsdaten.

Wegen der begrenzten Möglichkeiten des WS-BPEL-Standards zur Beschreibung menschlicher Interaktionen wurde der in der Benutzer-Lane dargestellte Prozess nicht als BPEL-Workflow implementiert, sondern lediglich in einem SoapUI-Testcase abgebildet. Die Verwendung der BPEL-Erweiterung BPEL4People zur Einbindung der Benutzerentscheidungen in einen Workflow wird in Abschnitt 4.4.3 als mögliche Erweiterung dieser Implementierung vorgeschlagen.

In den folgenden Abschnitten werden die Implementierungen und Schnittstellen der Teilworkflows vorgestellt.

[4] Als Netzwerkfähigkeit einer Methode wird hier die Möglichkeit verstanden, Matlab und die für die Webservice-Anbindung notwendige Software auf verschiedenen Computern in einem Netzwerk zu betreiben. Dies bietet Vorteile, wenn beides bspw. aus lizenzrechtlichen Gründen nicht auf dem selben Computer installiert werden kann.

4.3.1 Datenimport und -Export

Dieser BPEL-Workflow umfasst zwei Teile, die zeitlich getrennt ausgeführt werden und größtenteils voneinander unabhängig sind. Der Import-Teil wird vor der eigentlichen Modellreduktion ausgeführt und steuert sämtliche Schritte, die zur Initialisierung der Simulationsinstanz im Web Service Interface, zum Import der Ausgangsdaten und zur Vorbereitung der Modellreduktion notwendig sind. Der Export-Teil wird dann nach Abschluss des im nächsten Abschnitt beschriebenen Reduktionsworkflows ausgeführt und steuert den Export der reduzierten Modelldaten und weitere abschließende Aktionen. Zur Trennung der Variablengeltungsbereiche beider Teile dieses Workflows wurden die in Abschnitt 2.2.2 vorgestellten Scopes verwendet.

Der BPEL-Workflow zum Datenimport und -Export umfasst im wesentlichen die Aktivitäten, welche der Import/Export-Lane des in Abbildung 3.2 dargestellten Reduktionsprozesses zugeordnet sind. Diese wurden allerdings um implementierungsspezifische Operationen wie die Initialisierung der WSI-Instanz ergänzt. Ein weiterer wesentlicher Unterschied zum in Kapitel 3.2 beschriebenen Konzept betrifft die Verwaltung der Arbeitsdaten. Die Workflow-Anbindung von Matlab über die Kommandozeilenschnittstelle (siehe Kapitel 4.2.1) erfordert für jeden Workflowschritt eine neue Matlabinstanz, somit ist es nicht möglich, die Arbeitsdaten über den Matlab-Workspace von einem Schritt zum nächsten zu übertragen (vgl. Kapitel 4.1). Beim hier ausgeführten Datenimport werden die Daten deshalb nicht direkt in den Matlab-Workspace geladen sondern lediglich als .mat-File im WSI-Instanzverzeichnis bereitgestellt, von wo sie dann in den nachfolgenden Workflowschritten von Matlab eingelesen werden können.

Der Aufruf der beiden Teile des Workflows erfolgt über die Operationen `Import` und `Export` des ImportExport-Webservice, die in dieser Reihenfolge aufgerufen werden müssen, da nur durch Aufruf der `Import`-Operation eine neue Workflowinstanz erzeugt wird. Zur Zuordnung des `Export`-Aufrufs an eine bestehende Workflowinstanz wird ein Correlation Set (siehe Abschnitt 2.2.2) verwendet, welches über die ID der mit dem Workflow verknüpften WSI-Simulationsinstanz definiert ist.

Import

Der Import-Teil des Workflows wird vom Client über die Webservice-Operation `Import` gestartet, die folgende Argumente verlangt:

Parameter	Datentyp	Beschreibung
Filename	String	Dateiname des Archivs mit den zu reduzierenden Modelldaten.

Folgende Antwort wird synchron an den Client übertragen:

Parameter	Datentyp	Beschreibung
SimID	Long	ID der Simulationsinstanz im WSI.

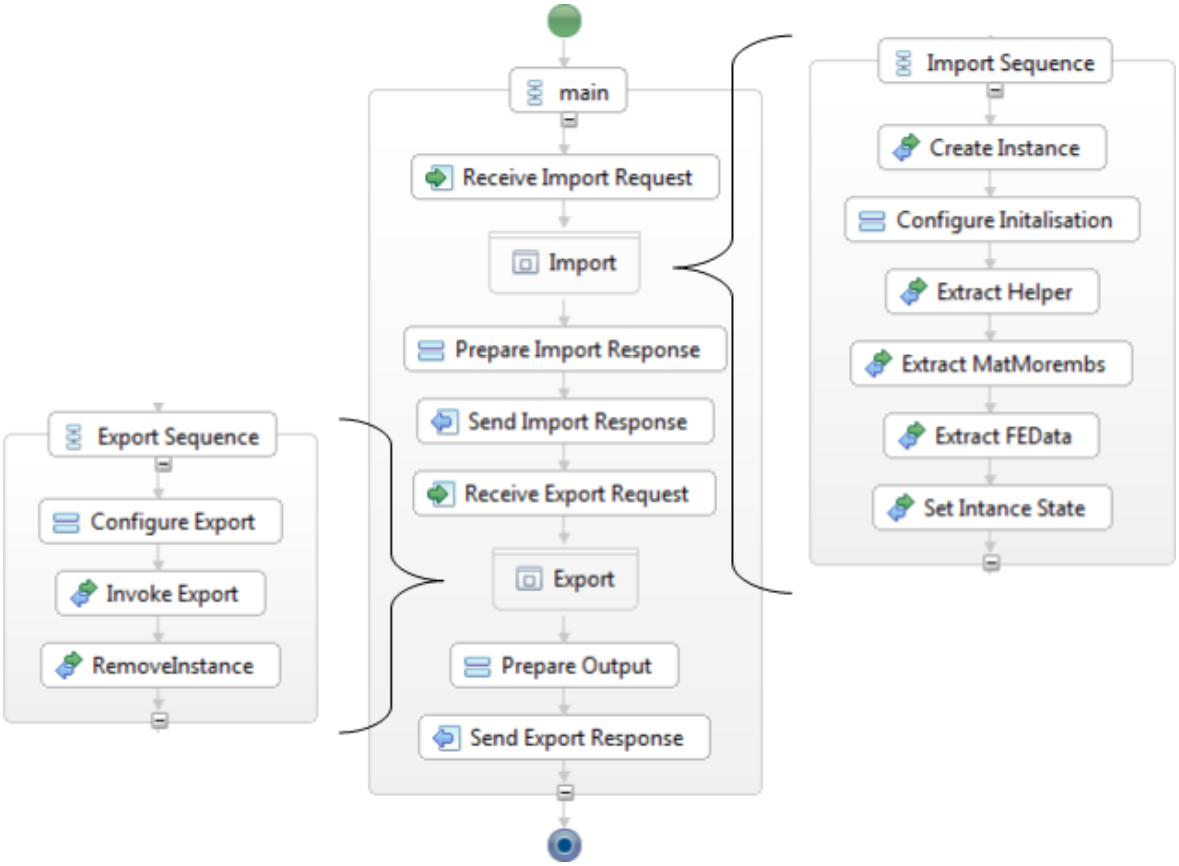

Abbildung 4.3: Workflow zum Datenimport und -Export (Eclipse BPEL-Designer)

Der Import-Teil (rechter Teil von Abbildung 4.3) steuert folgende Aufgaben:

- Erzeugen einer Simulationsinstanz im Web Service Interface. Dabei wird auch das Instanzverzeichnis erstellt („CreateInstance“).
- Kopieren und Entpacken des MatMorembs-Programmpakets („Extract MatMorembs“) und weiterer, für die Modellreduktion benötigter Software („ExtractHelper“) ins Instanzverzeichnis. Dazu wird die WSI-Operation `unpackFile` verwendet.
- Kopieren und Entpacken der Modelldaten ins Instanzverzeichnis, ebenfalls unter Verwendung der WSI-Operation `unpackFile` („Extract FEData“).
- Ggf. Umwandeln der Modelldaten in das für die Modellreduktion benötigte Format. Je nach Ausgangformat können dazu verschiedene MatMorembs-Funktionen aufgerufen werden. Dieser Schritt kann entfallen, wenn die Daten wie im hier gezeigten Beispiel bereits als .mat-File im MatMorembs-Format vorliegen.
- Zustand der WSI-Instanz auf „Runnable“ setzen („Set Instance State“).

Export

Der Export-Teil des Workflows wird vom Client über die Webservice-Operation `Export` gestartet, die folgende Argumente verlangt:

Parameter	Datentyp	Beschreibung
SimID	Long	ID der Simulationsinstanz im WSI.

Folgende Antwort wird synchron an den Client übertragen:

Parameter	Datentyp	Beschreibung
Filename	String	Name des Verzeichnisses, das die reduzierten Modelldaten enthält.

Der Export-Teil (linker Teil von Abbildung 4.3) übernimmt folgende Aufgaben:

- Anlegen eines Export-Verzeichnisses auf dem Matlab-Provider und Kopieren der reduzierten Modelldaten in dieses Verzeichnis. Dazu wird ein einfaches Shellscript aufgerufen, das die Modelldaten aus dem WSI-Instanzverzeichnis ins Exportverzeichnis verschiebt („Invoke Export“).
- Entfernen der WSI-Simulationsinstanz („Remove Instance“).

4.3.2 Modellreduktion und Auswertung

Dieser BPEL-Workflow steuert die Ausführung der Modellreduktion, die Analyse des reduzierten Modells mit den in Abschnitt 3.4 beschriebenen Metriken und die Berechnung eines Kennwerts für die Datenqualität aus den Ergebnissen der Metriken. Unterschreitet die Datenqualität die vom Benutzer gesetzte Minimalanforderung, wird dieser Vorgang in einer `RepeatUntil`-Schleife wiederholt, wobei die Dimension des reduzierten Systems mit jeder Iteration erhöht wird, was zu einem kleineren Fehlersystem und somit zu einer höheren Datenqualität führt. Abschließend werden dann die in Abbildung 3.7 gezeigten Frequenzgang-Visualisierungen erzeugt, um dem Benutzer eine Bewertung des reduzierten Modells zu ermöglichen.

Der BPEL-Workflow setzt eine WSI-Simulationsinstanz im Zustand „Runnable“ vorraus, deren Instanzverzeichnis bereits sämtliche zur Modellreduktion notwendige Software und die Ausgangsdaten im MatMorembs-Format enthält. Er kann daher erst nach Abschluss der `Import`-Operation des im vorigen Abschnitt beschriebenen Workflows für den Datenimport und -Export aufgerufen werden.

Der Workflow wird vom Client über die Webservice-Operation `Reduce` gestartet, welche die folgenden Argumente verlangt:

Parameter	Datentyp	Beschreibung
SimID	Long	ID der Simulationsinstanz im WSI.
ReductionMethod	String	Name der Reduktionsmethode (Modal,Kylov,Craig-Bampton).
InitialReductionDimension	Int	Zieldimension, auf die das Modell in der ersten Iteration reduziert wird.
RequiredDataQuality	float	Geforderte Datenqualität des reduzierten Modells.

Abhängig von der Größe des zu reduzierenden Modells und der Anzahl der benötigten Iterationen kann die Durchführung der Modellreduktion einige Zeit in Anspruch nehmen. Würde der Kommunikationskanal zwischen Workflowengine und Client während der gesamten Bearbeitungsdauer offen gehalten, wären Verbindungsabbrüche aufgrund von Zeitüberschreitungen in den zugrunde liegenden Protokollen (z.B. HTTP) wahrscheinlich. Deshalb wird die Antwort an den Client asynchron, also über einen neuen Transportkanal gesendet, der erst nach Abschluss aller vom Workflow gesteuerten Arbeiten geöffnet wird. Der Benutzer muss dazu den Webservice `ReductionProcessCallback` mit einer Operation `onResult` anbieten, die dann von der Workflowengine mit folgender Nachricht aufgerufen wird:

Parameter	Datentyp	Beschreibung
SimID	Long	ID der Simulationsinstanz im WSI. Aus dieser ergibt sich auch der Speicherort der Frequenzgang-Visualisierungen.
DataQuality	Float	Datenqualitätskennwert des reduzierten Modells.
ReductionDimension	Int	Dimension des reduzierten Modells.

Im Detail übernimmt der in Abbildung 4.4 dargestellte BPEL-Workflow zur Modellreduktion und Auswertung folgende Aufgaben:

- Durchführung der Modellreduktion unter Verwendung des vom Benutzer gewählten Reduktionsverfahrens (ReductionMethod) und mit der vom Benutzer vorgegebenen Zieldimension (InitialReductionDimension). Dafür wird die entsprechende MatMorembs-Funktion aufgerufen („Invoke Reduction“). Im Rahmen dieser Arbeit wurden nur die MatMorembs-Funktionen zur Durchführung des modalen Reduktionsverfahrens an den Workflow angebunden. Der BPEL-Workflow selbst ist jedoch auch für die Verwendung anderer Reduktionsverfahren geeignet.
- Analyse des reduzierten Systems und Berechnung der Fehlernormen wie in Abschnitt 3.4 erklärt („Invoke Analysis“). Dazu werden ebenfalls verschiedene MatMorembs-Funktionen verwendet.

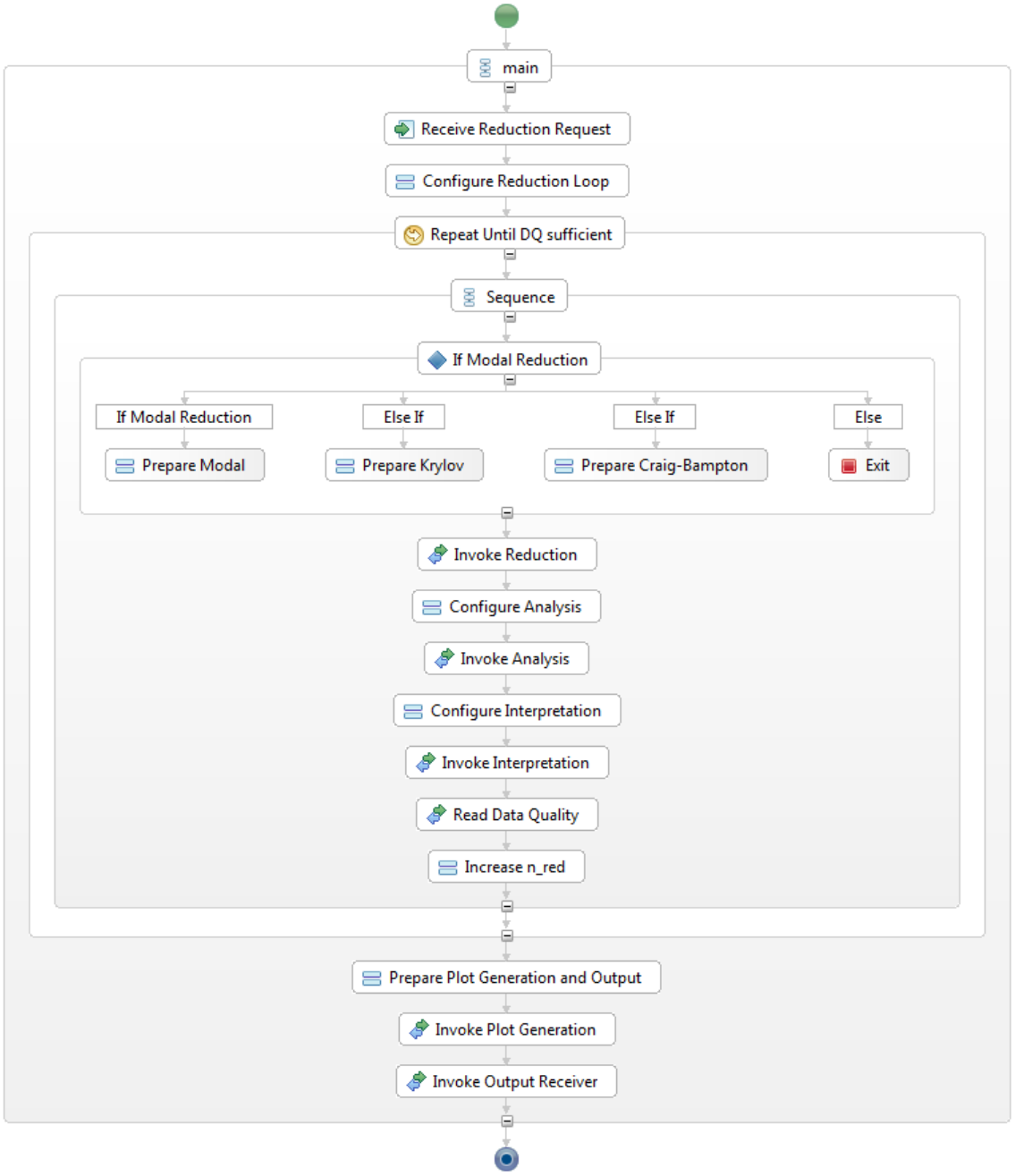

Abbildung 4.4: Workflow zur Modellreduktion und Auswertung des Ergebnisses (Eclipse BPEL-Designer)

- Berechnung eines Kennwerts für die Datenqualität aus den Werten der Fehlernormen unter Verwendung eines Python-Scripts, welches die in Gleichung 3.9 abgebildete Funktion implementiert („Invoke Interpretation“).
- Auslesen und Übertragen des Datenqualitätskennwerts in eine BPEL-Variable („Read Data Quality“).
- Erhöhen der Zieldimension, für eventuell folgende Iterationen („Increase n_red“).
- Vergleich des Datenqualitätskennwerts mit den vom Benutzer gesetzten Anforderungen (RequiredDataQuality) und Wiederholen des kompletten Vorgangs, bis die Datenqualität den Anforderungen entspricht („Repeat until DQ sufficient“).
- Generieren der Frequenzgänge zur manuellen Beurteilung des reduzierten Systems durch den Benutzer („Invoke Plot Generation“).

4.3.3 Benutzerworkflow

Der Ablauf des hier vorgestellten Workflows wird an verschiedenen Stellen durch Entscheidungen eines Benutzers beeinflusst. Die Benutzer-Lane des in Abbildung 3.2 dargestellten BPMN-Diagramms zeigt die Entscheidungen, die der Benutzer, also in der Regel der Wissenschaftler, der die Modellreduktion durchführt, treffen muss. Wie in Kapitel 2.1.2 erläutert, sehen Webservices allerdings keine Benutzerschittstelle, vor sondern dienen ausschließlich der Kommunikation zwischen verschiedenen Computerprogrammen. Da die in den letzten Abschnitten vorgestellten BPEL-Workflows nur über Webservices gesteuert werden können, ist eine Clientsoftware notwendig, welche dem Benutzer die für Entscheidungen notwendigen Informationen präsentiert und die Entscheidungen anschließend per Webservice an die BPEL-Workflows sendet.

Die Entwicklung einer solchen Clientsoftware geht über den Umfang dieser Arbeit hinaus. Um die implementierten BPEL-Workflows zu testen, wird aber auf der Benutzerseite dennoch eine Gegenstelle benötigt, mit der die BPEL-Workflows kommunizieren können und von der sie ihre Eingaben erhalten. Eine solche Gegenstelle wurde für diese Arbeit mit der Webservice-Testumgebung *SoapUI*[5] realisiert.

Der erstellte SoapUI-Testcase übernimmt dabei folgende Aufgaben:

- Aufruf der Import-Operation des ImportExport-Webservice (siehe Abschnitt 4.3.1) und Auswertung der Antwort.
- Aufruf der Reduce-Operation des Reduction-Webservice (siehe Abschnitt 4.3.2).
- Erstellen des ReductionProcessCallback-Webserivce und Empfang der asynchron gesendeten Benachrichtigung des Reduction-Webservice nach erfolgter Modellreduktion.
- Abfragen der Benutzerentscheidung, ob das Ergebnis der ausgeführten Modellreduktion zufriedenstellend ist.

[5] http://www.eviware.com/soapUI/soapui-products-overview.html

- Erneutes Ausführen der Modellreduktion (Aufruf der Reduce-Operation des Reduction-WebService), falls der Benutzer nicht mit dem bisher erzielten Ergebnis zufrieden ist. Er kann zuvor die Parameter der Modellreduktion anpassen.
- Aufruf der Export-Operation des ImportExport-Webservice (siehe Abschnitt 4.3.1).

Die Einbindung von Benutzerentscheidungen in diesen Teilworkflow ist auch Thema des Abschnitts 4.4.3, in dem die Verwendung der BPEL-Erweiterung BPEL4people für diesem Zweck erörtert wird.

4.4 Mögliche Erweiterungen

Der in dieser Arbeit vorgestellte Workflow muss als Prototyp betrachtet werden und bietet noch viel Potential für Erweiterungen und Verbesserungen. Dazu zählt zum einen die Workflow-Anbindung von weiteren MatMorembs-Funktionen, um andere Modellreduktionsverfahren im Workflow nutzen zu können und die Modelldaten aus weiteren Formaten importieren bzw. in weitere Formate exportieren zu können. Diese Funktionen können bei Bedarf auf die gleiche Weise wie die bereits verwendeten MatMorembs-Funktionen in den Workflow integriert werden, der damit verbundene Aufwand hält sich also in Grenzen.

Umfangreichere Erweiterungen und Verbesserungen des Workflows erfordern dagegen in der Regel den Einsatz weiterer Technologien und Konzepte. Ein Beispiel dafür ist die Verbesserung der Performanz bei der Anbindung von Matlab. In der hier vorgestellten Implementierung wird Matlab über die Kommandozeilenschnittstelle an den Workflow angebunden (siehe Abschnitt 4.2). Über diese kann Matlab nicht interaktiv verwendet werden, deshalb muss für jeden Workflowschritt, der zu seiner Ausführung Matlab benötigt, eine neue Matlab-Instanz gestartet werden. Die Verwendung von Unix-Pipes, der COM-Schnittstelle oder der Matlab-Engine zur Matlab-Anbindung könnte wesentliche Geschwindigkeitsvorteile bringen, da diese die Nutzung einer einzigen Matlab-Instanz für mehrere Workflowschritte erlauben. Abschnitt 4.2.3 geht näher auf diese Anbindungsmethoden ein.

In den folgenden Abschnitten werden weitere Erweiterungsmöglichkeiten des Workflows und die dafür notwendigen Technologien und Konzepte vorgestellt.

4.4.1 Data-Quality-Framework

Ein entscheidender Aspekt des Modellreduktions-Workflows ist die Analyse und Bewertung des reduzierten Simulationsmodells. Die dafür notwendigen Teile des Workflows verteilen sich auf verschiedene Workflowteilnehmer und wurden für diese Arbeit unter Verwendung unterschiedlicher Technologien implementiert. So wird das reduzierte Modell mit verschiedenen, in Matlab implementierten Metriken untersucht. Die Bewertung der Metrikergebnisse erfolgt dann in zwei Stufen zuerst automatisiert durch ein Python-Script und dann manuell durch einen Wissenschaftler (siehe Abschnitt 3.4).

Als Alternative dazu könnten alle diese Aktivitäten auf Basis des in [Bre11] vorgestellten *Java Data Quality Frameworks (JDQF)* implementiert werden. Dieses stellt ein Kernsystem zur Verfügung, welches grundlegende Aufgaben der Datenqualitätsberechnung und deren Vorbereitung, der Datenbereitstellung, der Verwaltung von Eingabedaten und Berechnungsergebnissen sowie der Kommunikation mit der Workflowengine und weiteren externen Stellen übernimmt. Durch eine Reihe von Schnittstellen lassen sich die eigentlichen Metrik- und Interpretationsimplementierungen sowie weitere Komponenten als Plugins realisieren, welche die Funktionalität des Kernsystems nutzen können und deren Ausführung von diesem verwaltet wird. Dieses hochgradig modulare Konzept erlaubt die Verwendung des Java Data Quality Frameworks auch für Berechnungsaufgaben mit sehr speziellen Anforderungen, wie sie im Modellreduktionsworkflow auftreten. Von besonderem Interesse ist in diesem Fall die *ExternalTaskAPI* des JDQF, welche die Auslagerung von Berechnungsvorgängen in externe Programme ermöglicht. Dies ist notwendig, wenn die in MatMorembs enthaltenen Metrikimplementierungen zur Datenqualtitätsberechnung herangezogen werden sollen, da diese die Legacy-Software Matlab für ihre Ausführung benötigen. Die gleiche Schnittstelle kann aber auch genutzt werden, um Human Task Systeme anzubinden und auf diese Weise menschliche Entscheidungen in die Analyse- und Bewertungsvorgänge zu integrieren. Einfachere Berechnungen, wie die bisher in einem Python-Script realisierte, automatische Bewertung der Metrikergebnisse, können natürlich auch in Java als reguläres JDQF-Plugin neu implementiert werden.

Der bisher sehr heterogen implementierte Vorgang der Datenqualitätsberechnung im Modellreduktionsworkflow könnte also auf der Basis des Java Data Quality Frameworks konsolidiert werden. Durch die einheitliche Basis ließe sich der Vorgang einfacher und wartungsfreundlicher gestalten. Weitere Vorteile ergäben sich im Hinblick auf die Wiederverwendbarkeit und Austauschbarkeit der Komponenten. So könnten einerseits bereits existierende JDQF-Plugins verwendet werden, um den Berechnungsvorgang zu erweitern. Von besonderem Interesse wären hier Plugins zur Anbindung von Human Task Systemen, um die Bewertung der Metrikergebnisse durch einen Wissenschaftler besser in den Workflow zu integrieren. Auf der anderen Seite könnten die für diesen Workflow entwickelten Metrik- und Interpretationsimplementierungen für andere Anwendungen wiederverwendet werden, die ebenfalls das JDQF zur Datenqualitätsberechnung nutzen.

4.4.2 Visualisierung der Ergebnisse

Im Anschluss an die Modellreduktion werden die Frequenzgänge des ursprünglichen Simulationsmodells, des reduzierten Modells und des bei der Reduktion entstandenen Fehlers visualisiert (siehe Kapitel 3.4). Diese Visualisierungen dienen als Grundlage für die Bewertung des reduzierten Modells durch einen Wissenschaftler. Im hier vorgestellten Workflow werden zur Visualisierung lediglich zweidimensionale, statische Graphen verwendet (vgl. Abbildung 3.7). Diese werden im implementierten Workflow von den entsprechenden MatMorembs-Funktionen in Matlab erstellt und dann als PNG-Grafik in das jeweilige Instanzverzeichnis exportiert, wo sie dann vom Benutzer betrachtet werden können.

Diese Vorgehensweise weist noch verschiedene Probleme und Einschränkungen auf. Zum einen ist sie nicht besonders benutzerfreundlich, da die erzeugten Grafiken von Hand aufgesucht

und geöffnet werden müssen. Zum anderen beschränkt sie die Art der Visualisierungen auf statische Grafiken, die in einer Bilddatei gespeichert werden können. Beide Probleme können mit der in [RHKW11] vorgeschlagenen Methode gelöst werden. Die grafische Ausgabe wird dabei von einer auf dem Client-Rechner installierten Visualisierungskomponente generiert und kann von der Workflowengine über einen Webservice gesteuert werden. Somit kann die Visualisierung automatisch ausgegeben werden, wenn die entsprechenden Daten vorliegen. Die Visualisierungskomponente basiert auf dem Visualisierung-Toolkit VTK, welches neben einfachen zweidimensionalen Grafiken auch 3D-Modelle interaktiv darstellen kann und somit weitaus komplexere Visualisierungsmethoden beherrscht. Allerdings ist der Entwicklungsaufwand dafür auch deutlich größer als bei der bisher implementierten, auf Matlab und MatMorembs basierenden Methode.

4.4.3 Menschliche Interaktion

Die dem Benutzer zugeordnete Lane des in Kapitel 3 beschriebenen und in Abbildung 3.2 dargestellten Workflows wurde in dieser Studienarbeit nicht als ausführbarer BPEL-Prozess modelliert, die dem Benutzer zugeordneten Aktionen werden stattdessen lediglich durch einen SoapUI-Testcase simuliert (siehe Abschnitt 4.3.3). Dies ist in dem Umstand begründet, dass sich durch Menschen zu bearbeitende Aufgaben, im folgenden als *Human Tasks* bezeichnet, nur schlecht in einen BPEL-Workflow einbinden lassen: BPEL-Workflows kommunizieren nach außen hin ausschließlich über Webservices, können also nur durch andere, Webservice-taugliche Software, und nicht durch menschliche Benutzer beeinflusst werden. Zwar könnte eine einfache Client-Software entwickelt werden, welche die Benutzerentscheidungen per Webservice-Aufruf an den Workflow weiterreicht. Diese Client-Software müsste allerdings für jeden zu steuernden Workflow gezielt entwickelt werden, da BPEL ohne Erweiterungen keine Möglichkeit vorsieht, das Benutzerinterface oder weitere Parameter der menschlichen Interaktion wie Benutzerrechte oder Rollenzuweisungen zu konfigurieren.

Um diese Lücke des BPEL-Standards zu schließen, wurden die beiden aufeinander aufbauenden Spezifikationen WS-BPEL4People [Ing09b] (kurz BPEL4People) und WS-HumanTask [Ing09a] entwickelt. Sämtliche Informationen in den folgenden Absätzen stammen aus diesen beiden Spezifikationen.

Die Infrastruktur zur Workflowanbindung von Human Tasks gliedert sich nach der WS-HumanTask-Spezifikation in drei Komponenten (siehe Abbildung 4.5). Die Workflowengine steuert den Ablauf des Workflows und stößt die durch den Benutzer zu erledigende Aufgabe an, sie nimmt in diesem Szenario die Rolle des *Task Parent* ein. Der *Task Processor* erzeugt daraufhin aus der ihm vorliegenden *Task Definition* eine konkrete Task-Instanz, kurz *Task* genannt. Er weist die Aufgabe dann einem Benutzer zu, verwaltet ihre Ausführung, reagiert auf bestimmte Ereignisse wie dem Ablaufen zeitlicher Deadlines und gibt das Ergebnis der Aufgabe letztendlich an das Task Parent zurück, in der Regel über eine asynchrone Verbindung. Die direkte Kommunikation mit dem Benutzer wird allerdings durch eine dritte Komponente, dem *Task Client* übernommen. Dieses Programm läuft auf dem Computer des Benutzers und zeigt diesem alle durch ihn zu erledigenden Aufgaben sowie Formulare für die Eingabe der Ergebnisse, Kommentare usw. an.

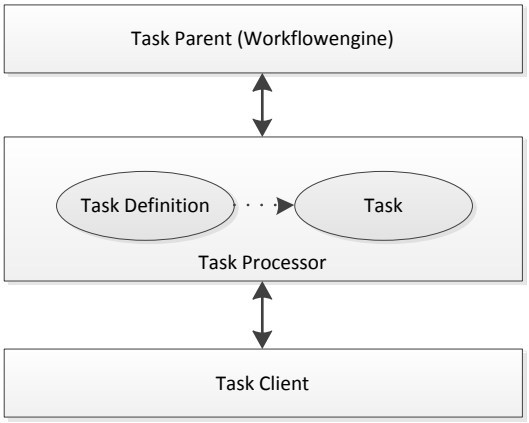

Abbildung 4.5: Architektur der Human Task Infrastruktur (vgl. [Ing09a])

Die Aufteilung der Infrastruktur in drei Komponenten hat eine große Flexibilität des Systems zur Folge. So kann ein Task Processor die Aufgaben verschiedener Workflows oder auch anderer Task Parents, z.B. Issue-Tracking-Systeme[6], gemeinsam verwalten. Ein Task Client kann wiederum alle Aufgaben eines Benutzers gemeinsam darstellen, auch wenn diese aus unterschiedlichen Quellen stammen. Die WS-HumanTask-Spezifikation definiert sämtliche Schnittstellen zwischen den Komponenten sowie eine XML-basierte Sprache zur Definition von Human Tasks. Dadurch wird die Interoperabilität verschiedener Human Task Systeme möglich, Komponenten verschiedener Hersteller können also beim Aufbau einer Infrastruktur kombiniert werden.

BPEL4People stellt dagegen eine Erweiterung von WS-BPEL dar, mit der Human Tasks in einen Workflow eingebunden werden können. Dazu wird BPEL um das Element *people activity* erweitert, welches zum Aufruf des Human Tasks dient. Der Human Task kann dabei entweder im BPEL-Workflow selbst (*inline*) oder in einem externen Human Task System definiert werden. Im zweiten Fall können mehrere Workflows die selbe Aufgabendefinition nutzen, allerdings kann dann bei der Ausführung des Tasks vom Task Processor nicht auf interne BPEL-Variablen zugegriffen werden. Außerdem wird zwischen echten Aufgaben und Benachrichtigungen unterschieden. Im Fall von Benachrichtigungen wird der Benutzer lediglich über einen Umstand informiert, kann allerdings anders als bei einer Aufgabe den Ablauf des Workflows nicht beeinflussen. Ferner sieht die BPEL4People-Spezifikation umfangreiche Möglichkeiten vor, um Aufgaben an bestimmte Personen oder an Gruppen von Personen zuzuweisen. Diese werden hier jedoch nicht näher erörtert, da sie für diese Arbeit keine große Bedeutung haben.

Sämtliche durch den Benutzer auszuführenden Aktivitäten im hier betrachteten Modellreduktionsprozess ließen sich mit WS-HumanTask beschreiben und mit BPEL4People an den Workflow anbinden. Der verwendete Softwarestack müsste zum Aufbau einer Human Task Infrastruktur allerdings deutlich erweitert werden. Insbesondere müsste eine Alternative oder Erweiterung

[6]Ein Issue-Tracking-System oder Ticketing-System verwaltet die Entgegennahme und Bearbeitung von Kundenanfragen.

zur Workflowengine Apache ODE gefunden werden, da diese den BPEL4People-Standard bisher nicht unterstützt. Außerdem müssten alle Human Tasks entsprechend der Spezifikationen beschrieben werden. Da im betrachteten Prozess nur an wenigen Stellen echte Nutzerinteraktion notwendig ist, wäre die Entwicklung einer leichtgewichtigen, zweckgebundenen Client-Software wahrscheinlich die einfachere Lösung. Sollte bei zukünftigen Erweiterungen des Prozesses allerdings umfangreicherer Nutzerinteraktion notwendig werden, wäre die Verwendung einer Human Task-Infrastruktur auf Basis von BPEL4People und WS-HumanTask eine sinnvolle Ergänzung.

4.4.4 Fehlerbehandlung

In zahlreichen Aktivitäten des Modellreduktionsworkflows können Fehler oder Ausnahmesituationen auftreten. Gründe dafür können Fehler in den verarbeiteten Daten (z.B. leere Systemmatrizen im Simulationsmodell), in den Nutzereingaben (z.B. geforderte Datenqualität größer 1) oder auch in der verwendeten Infrastruktur (z.B. fehlende Schreibrechte im Exportverzeichniss, fehlende Hilfsdateien) sein. Im implementierten Workflow werden solche Fehlerquellen nicht gesondert behandelt, es wird grundsätzlich von der Korrektheit aller Eingaben und Daten sowie von einer korrekt eingerichteten Infrastruktur ausgegangen. Tritt ein solcher Fehler auf, führt dies entweder zur Terminierung oder zum Einfrieren der Workflowinstanz, ohne dass der Nutzer direkt über den Fehler und seine Ursache informiert wird.

Die implementierten BPEL-Workflows könnten anhand der in BPEL verfügbaren Fehlerbehandlungsmechanismen so erweitert werden, dass Fehler abgefangen werden und zur Ausführung festgelegter Aktivitäten führen. Die meisten potentiellen Fehler des Prozesses sollten sinnigerweise dadurch behandelt werden, dass der Benutzer über deren Ursache informiert wird und die Möglichkeit hat, die fehlerhaften Eingangsdaten oder Parameter zu korrigieren. Da die BPEL-Workflows allerdings nicht direkt mit dem Benutzer kommunizieren können, müsste die verwendete Client-Software in der Lage sein, diese Fehlermeldungen an den Benutzer weiterzuleiten und ihn zu entsprechenden Maßnahmen auffordern. Da diese Arbeit nicht die Entwicklung einer solchen Client-Software umfasst, wurde auf die Implementierung der Fehlerbehandlung verzichtet.

5 Zusammenfassung und Ausblick

Durch die Verfügbarkeit immer leistungsfähigerer Computer und immer schnellerer Datennetze wird in der Wissenschaft zunehmend auf weltweite Zusammenarbeit zwischen verschiedenen Forschungsinstituten sowie auf Computersimulationen unter Verwendung verteilter, heterogener Rechnersysteme gesetzt (Stichwort e-Science). Dadurch steigen die Anforderungen der Wissenschaft an die verwendete Software, besonders im Hinblick auf Netzwerkfähigkeit, Interoperabilität, Wiederverwendbarkeit und Flexibilität von Softwaresystemen. Ähnliche Anforderungen führten in der wirtschaftsorientierten Informatik zur Entwicklung des Konzepts der Serviceorientierten Architektur und von Workflowsystemen, welche die schnelle Bearbeitung komplexer Aufgabenstellungen durch die lose Kopplung verschiedener Dienste in verteilten Umgebungen erlauben. Die Übertragung dieser Technologien in den wissenschaftlichen Bereich liegt nahe und führte zum Begriff der wissenschaftlichen Workflows, einem aktuellen Forschungsthema am Institut für Architektur von Anwendungssystemen (IAAS) und am Institut für Parallele und Verteilte Systeme (IPVS) an der Universität Stuttgart.

In dieser Studienarbeit wurde untersucht, wie Workflowtechnologien zur automatisierten Ausführung der Modellreduktion eines elastischen Mehrkörpersystems verwendet werden können. Die Modellreduktion ist eine Vorstufe zur Simulation eines mathematischen Modells und dient dazu, die Zahl der Freiheitsgrade des Simulationsmodells zu reduzieren und dabei das Verhalten des Originalsystems möglichst gut zu approximieren. Durch die verminderte Modelldimension kann die Komplexität der nachgelagerten Simulation gesenkt und ihre Ausführungsgeschwindigkeit erhöht werden. Das Institut für Technische und Numerische Mechanik (ITM) der Universität Stuttgart hat unter dem Namen MatMorembs ein Programmpaket entwickelt, um solche Modellreduktionen auf Basis der Simulations- und Numerikumgebung Matlab auszuführen. Dieses Programmpaket dient in dieser Arbeit als Grundlage für die Entwicklung eines Workflows zur Modellreduktion.

Die wissenschaftlichen Disziplinen der Technischen Mechanik und der Informatik haben bisher nur wenige Berührungspunkte und die Verwendung von Workflowtechnologie ist in der Technischen Mechanik noch ein neues Feld. Um Vertreter beider Disziplinen mit den Grundlagentechnologien des jeweils anderen Bereichs vertraut zu machen, wurden in dieser Arbeit zunächst die Konzepte und Technologien der Serviceorientierten Architektur, der Workflowsysteme, der Simulationstechnik sowie die mechanischen Grundlagen der Modellreduktion näher vorgestellt. Anschließend wurde ein Konzept für einen Workflow zur Modellreduktion entwickelt. Dafür wurde der in MatMorembs durchgeführte Modellreduktionsvorgang näher untersucht und in sinnvolle Workflowschritte unterteilt. Aus diesen Schritten wurde ein Prozessmodell entworfen und unter Verwendung von BPMN-Diagrammen dargestellt. Ausgehend von diesem Prozessmodell wurden daraufhin ausführbare BPEL-Workflows zur Durchführung der Modellreduktion entwickelt. Dazu musste zunächst geklärt werden, wie die Arbeitsdaten

mehrerer parallel laufender Modellreduktionsinstanzen verwaltet werden können und wie die Legacy-Software Matlab an einen BPEL-Workflow angebunden werden kann.

Die so entwickelte Implementierung muss als Prototyp für einen Workflow zur automatisierten Ausführung einer Modellreduktion verstanden werden. Sie zeigt, dass die Verwendung von Workflowtechnologie für diesen Anwendungsbereich möglich ist und demonstriert einige Vorteile, die sich dadurch ergeben können. Insbesondere wird aufgezeigt, wie sich ein wissenschaftlicher Workflow mit geringem Aufwand durch die Kombination verschiedener, in einem Netzwerk verfügbarer Dienste orchestrieren lässt. Für einen sinnvollen Einsatz in der Praxis sind allerdings noch verschiedene Verbesserungen und Erweiterungen der hier vorgestellten Implementierung notwendig. Einige Technologien und Konzepte, die dabei zum Einsatz kommen können, wurden in dieser Arbeit ebenfalls vorgestellt.

Danksagung

Meinen besonderen Dank möchte ich an dieser Stelle meinen Betreuern Peter Reimann vom Institut für Parallele und Verteilte Systeme (IPVS), Michael Reiter vom Institut für Architektur von Anwendungssystemen (IAAS) und Jörg Fehr vom Institut für Technische und Numerische Mechanik (ITM) aussprechen, die mich bei dieser Studienarbeit in Kooperation unterstützt haben.

Literaturverzeichnis

[AEJ+10] P. Adams, P. Easton, E. Johnson, R. Merrick, M. Phillips. SOAP over Java Message Service 1.0. W3c working, W3C, 2010. URL http://www.w3.org/TR/2010/WD-soapjms-20101026. (Zitiert auf Seite 16)

[Apa] ODE - Architectural Overview. URL http://ode.apache.org/architectural-overview.html. (Zitiert auf Seite 49)

[Bre11] U. Breitenbücher. *Datenqualität in Simulation-Workflows.* Master's thesis, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, 2011. (Zitiert auf den Seiten 5, 57 und 78)

[BZ01] K. Bathe, P. Zimmermann. *Finite-Elemente-Methoden.* Springer, 2001. (Zitiert auf den Seiten 5, 41 und 42)

[BZBP09] H. Bungartz, S. Zimmer, M. Buchholz, D. Pflüger. *Modellbildung und Simulation: Eine anwendungsorientierte Einführung.* Springer, 2009. (Zitiert auf den Seiten 5, 34, 35 und 36)

[CCMW01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3c note, W3C, 2001. Http://www.w3.org/TR/2001/NOTE-wsdl-20010315. (Zitiert auf Seite 18)

[CG08] V. Curcin, M. Ghanem. Scientific workflow systems - can one size fit all? *2008 Cairo International Biomedical Engineering Conference*, pp. 1–9, 2008. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4786077. (Zitiert auf Seite 39)

[Cri07] Cristcost. Comparison and analogies between WSDL 1.1 and 2.0 structures., 2007. URL http://commons.wikimedia.org/wiki/File:WSDL_11vs20.png. (Zitiert auf den Seiten 5 und 20)

[Dor11] R. Dormien. *Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen.* Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2011. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIP-3127&engl=0. (Zitiert auf Seite 67)

[FE11] J. Fehr, P. Eberhard. Simulation Process of Flexible Multibody Systems with Non-modal Model Order Reduction Techniques. *Multibody System Dynamics*, 25(3):313–334, 2011. (Zitiert auf Seite 43)

[Feh11] J. Fehr. *Automated and Error-controlled Model Reduction in Elastic Multibody Systems*. Dissertation, Schriften aus dem Institut für Technische und Numerische Mechanik der Universität Stuttgart, Band 21. Shaker Verlag, 2011. (Zitiert auf Seite 58)

[FG08] J. Freund, K. Götzer. *Vom Geschäftsprozess zum Workflow: Ein Leitfaden für die Praxis*. Hanser Fachbuchverlag, 2008. (Zitiert auf Seite 20)

[FN] J. Fehr, C. Nowakowski. Matmorembs Documentations. URL http://www.itm.uni-stuttgart.de/itmwiki/index.php/MATMOREMBS. (Zitiert auf Seite 48)

[FZ09] P. Finger, K. Zeppenfeld. *SOA und Webservices*. Informatik Im Fokus. Springer, 2009. (Zitiert auf den Seiten 5, 6, 11, 13, 14, 15 und 17)

[Gha08] I. Ghalimi. Why BPEL Matters, 2008. URL http://itredux.com/2008/09/28/why-bpel/. (Zitiert auf den Seiten 26 und 30)

[GHMJJM07] M. Gudgin, M. Hadley, N. Mendelsohn, A. K. Y. L. Jean-Jacques Moreau, Henrik Frystyk Nielsen. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). W3c recommendation, W3C, 2007. URL http://www.w3.org/TR/2007/REC-soap12-part1-20070427/. (Zitiert auf den Seiten 5 und 16)

[GK89] R. Gasch, K. Knothe. *Strukturdynamik. 2. Kontinua und ihre Diskretisierung*. Strukturdynamik / Robert Gasch; Klaus Knothe. Springer, 1989. (Zitiert auf Seite 43)

[Glo11] S. Glotzer. *International Assessment of Research and Development in Simulation-Based Engineering and Science*. World Scientific Pub Co Inc, 2011. (Zitiert auf Seite 7)

[Gun01] M. Gunderloy. Calling COM Components from .NET Clients, 2001. URL http://msdn.microsoft.com/library/ms973800.aspx. (Zitiert auf Seite 69)

[Hav05] M. Havey. *Essential Business Process Modeling*. O'Reilly Media, Inc., 2005. (Zitiert auf Seite 25)

[Hol95] D. Hollingsworth. Workflow Management Coalition - The Workflow Reference Model. Technical report, Workflow Management Coalition, 1995. URL http://www.wfmc.org/index.php?option=com_docman&task=doc_download&gid=92&Itemid=72. (Zitiert auf den Seiten 5 und 22)

[HW10] G. Hager, G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. Computational Science. Taylor & Francis, 2010. (Zitiert auf Seite 7)

[Ing09a] D. Ings. Web Services – Human Task (WS-HumanTask) Specification Version 1.1. Technical report, OASIS, 2009. URL http://docs.oasis-open.org/bpel4people/ws-humantask-1.1-spec-cd-06.pdf. (Zitiert auf den Seiten 5, 79 und 80)

[Ing09b] D. Ings. WS-BPEL Extension for People (BPEL4People) Specification Version 1.1. Technical report, OASIS, 2009. URL http://docs.oasis-open.org/bpel4people/bpel4people-1.1-spec-cd-06.pdf. (Zitiert auf Seite 79)

[JE07] D. Jordan, J. Evdemon. Web Services Business Process Execution Language Version 2.0. Technical report, OASIS, 2007. URL http://docs.oasis-open.org/wsbpel/2.0/. (Zitiert auf Seite 25)

[JMS06] M. Juric, B. Mathew, P. Sarang. *Business Process Execution Language for Web Services*. From technologies to solutions. Packt Publishing, 2006. (Zitiert auf den Seiten 5, 26 und 27)

[Jos06] S. Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648 (Proposed Standard), 2006. URL http://www.ietf.org/rfc/rfc4648.txt. (Zitiert auf Seite 17)

[KBS05] D. Krafzig, K. Banke, D. Slama. *Enterprise SOA: service-oriented architecture best practices*. The Coad series. Prentice Hall Professional Technical Reference, 2005. (Zitiert auf den Seiten 5 und 12)

[Ley05] F. Leymann. The (Service) Bus: Services Penetrate Everyday Life. In *ICSOC*, Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2005. (Zitiert auf Seite 13)

[LR00] F. Leymann, D. Roller. *Production workflow: concepts and techniques*. Prentice Hall PTR, 2000. (Zitiert auf den Seiten 20 und 21)

[Lun08] J. Lunze. *Regelungstechnik 1, Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Springer, Berlin Heidelberg, 2008. doi: http://dx.doi.org/10.1007/978-3-540-68909-6. (Zitiert auf Seite 33)

[Mat] Matlab R2010b Documentaion. URL http://www.mathworks.com/help/releases/R2010b/techdoc/matlab_product_page.html. (Zitiert auf den Seiten 5, 45 und 69)

[MBF+04] F. McCabe, D. Booth, C. Ferris, D. O. M. Champion, E. Newcomer, H. Haas. Web Services Architecture. W3c note, W3C, 2004. Http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. (Zitiert auf Seite 14)

[ND07] D. Nicklas, F. Dürr. Presentation Slides for Lecture 'Net based Applications', 2007. (Zitiert auf Seite 16)

[NFE11] C. Nowakowski, J. Fehr, P. Eberhard. Model Reduction for a Crankshaft Used in Coupled Simulations of Engines. In *Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics 2011, Brussels, Belgium*. 2011. (Zitiert auf den Seiten 5 und 48)

[OADH06] C. Ouyang, W. M. van der Aalst, M. Dumas, A. H. ter Hofstede. From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way, 2006. URL http://eprints.qut.edu.au/5266/. (Zitiert auf Seite 25)

[Oh10] S. Oh. *Verwendung von modernen Modellordnungsreduktionsverfahren für Ride-Simulationen*. Master's thesis, Institut für Technische und Numerische Mechanik, Universität Stuttgart, 2010. (Zitiert auf Seite 42)

[OMG10] OMG. BPMN 2.0 by Example. Technical report, OMG, 2010. URL http://www.omg.org/spec/BPMN/2.0/examples/PDF. (Zitiert auf den Seiten 5, 24 und 25)

[OMG11] OMG. Business Process Model and Notation (BPMN) Version 2.0. Technical report, OMG, 2011. URL http://www.omg.org/spec/BPMN/2.0/. (Zitiert auf den Seiten 23 und 25)

[OPM+09] E. Ogasawara, C. Paulino, L. Murta, C. Werner, M. Mattoso. Experiment Line: Software Reuse in Scientific Workflows. In M. Winslett, editor, *Scientific and Statistical Database Management*, volume 5566 of *Lecture Notes in Computer Science*, pp. 264–272. Springer Berlin / Heidelberg, 2009. (Zitiert auf Seite 7)

[PH07] M. P. Papazoglou, W.-J. van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16:389–415, 2007. (Zitiert auf den Seiten 5, 11, 12, 13 und 15)

[Ran08] R. Rannacher. Numerische Mathematik 2 (Numerik Partieller Differentialgleichungen), Vorlesungsskriptum, 2008. (Zitiert auf Seite 40)

[RHKW11] M. Reiter, M. Hlawatsch, D. Karastoyanova, D. Weiskopf. Unifying Simulation and Visualization environments with Workflow Technology. In *SimTech Konferenz*. 2011. (Zitiert auf Seite 79)

[RRS+11] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, F. Leymann. SIMPL - A Framework for Accessing External Data in Simulation Workflows. In *BTW*, pp. 534–553. 2011. (Zitiert auf den Seiten 7 und 64)

[Rut09] J. Rutschmann. *Generic Web Service Interface*. Master's thesis, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, 2009. (Zitiert auf den Seiten 30 und 32)

[Sch99] T. Schramm. Matlab - eine Einführung, 1999. (Zitiert auf den Seiten 6 und 44)

[Sim08] P. Simeon. Berechnung von Eigenwerten mit CUDA. 2008. (Zitiert auf Seite 57)

[SW99] R. Schwertassek, O. Wallrapp. *Dynamik flexibler Mehrkörpersysteme*. Vieweg, Braunschweig, 1999. (Zitiert auf Seite 42)

[Tan03] A. Tanenbaum. *Computernetzwerke*. Pearson Studium, 2003. (Zitiert auf Seite 15)

[Tay07] I. Taylor. *Workflows for e-science: scientific workflows for grids*. Springer, 2007. (Zitiert auf den Seiten 8 und 38)

[TMF+10] W. Tan, P. Missier, I. Foster, R. Madduri, D. De Roure, C. Goble. A comparison of using Taverna and BPEL in building scientific workflows: the case of caGrid. *Concurrency and Computation Practice and Experience*, 22(9):1098–1117, 2010. (Zitiert auf Seite 39)

[Wal94] O. Wallrapp. Standardization of Flexible Body Modeling in Multibody System Codes,Part I: Definition of Standard Input Data. *Mechanics of Structures and Machines*, 22(3):283–304, 1994. (Zitiert auf Seite 60)

[Wei02] K. Weicker. *Evolutionäre Algorithmen*. Leitfäden der Informatik. Teubner, 2002. (Zitiert auf Seite 38)

[WF04] P. Walmsley, D. C. Fallside. XML Schema Part 0: Primer Second Edition. W3c recommendation, W3, 2004. Http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/. (Zitiert auf Seite 18)

[Wie04] G. Wiehler. *Mobility, Security und Web Services: neue Technologien und Service-orientierte Architekturen für zukunftsweisende IT-Lösungen*. Publicis Corporate Publ., 2004. (Zitiert auf Seite 15)

Alle URLs wurden zuletzt am 19.10.2011 geprüft.

Erklärung

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

(Simon Remppis)