Universität Stuttgart

Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Diplomarbeit Nr. 3444

Integration der SimTech Workflow-Umgebung mit existierenden eScience Plattformen

Daniel Huss

Studiengang:	Softwaretechnik
Prüfer:	Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer:	Dipl.-Inf., Dipl.-Wirt. Ing.(FH) Karolina Vukojevic
begonnen am:	27.12.2012
beendet am:	09.08.2013
CR-Klassifikation:	D.2.12, H.4.1, I.6.7

Zusammenfassung

In den letzten Jahren sind vor allem in den USA und in Großbritannien neue digitale Plattformen und Softwarewerkzeuge für die Zusammenarbeit von Wissenschaftlern und zur gemeinsamen Nutzung von Rechenkapazitäten entstanden. Die Plattformen nanoHUB, myExperiment und das wissenschaftliche Workflow-Management-System Taverna sind für diese Arbeit ausgewählt worden, um eine Integration der angebotenen Dienste und Funktionen mit der SimTech Workflow-Umgebung zu prüfen. Für jeden der drei Kandidaten wurden mehrere mögliche Integrationsansätze erarbeitet und eine technische Realisierung der Integration umrissen. Anschließend wurden die Integrationsansätze einer Bewertung hinsichtlich ihres Nutzens und der Machbarkeit unterzogen. Es musste festgestellt werden, dass keine der untersuchten Plattformen die Auslagerung von SimTech-Infrastruktur auf die Rechenressourcen der Plattform ermöglicht, obwohl dies ein wichtiges Ziel der Arbeit war. Als Alternative wurde deshalb nach Abwägung der sonstigen Integrationsansätze eine Integrationslösung konstruiert, welche die Ausführung von datenflussorientierten Taverna-Workflows als Bestandteil von BPEL-Prozessen ermöglicht.

Abstract

Over the last few years, new digital platforms and software tools for the global collaboration between scientists and for the sharing of computing resources have emerged, primarily in the US and in Great Britain. In this work, the platforms nanoHUB and myExperiment, as well as the scientific workflow management system Taverna haven been selected for analysis of potential integration opportunities with the SimTech workflow environment. For each of those candidates, several integration approaches have been identified and a draft for a potential technical realization has been developed. Each integration approach has subsequently been evaluated for its usefulness and feasibility within the given time frame. As a result of this evaluation, none of the presented platforms seems to offer a way to satisfyingly externalize existing SimTech infrastructure or applications. The main objective of this work could therefore not be achieved, and an alternative approach had to be chosen. As the best remaining integration approach, the inclusion of Taverna's data flow centric workflows within BPEL processes has been implemented.

Inhalt

1 Einleitung

Im vergangenen Jahrzehnt sind vor allem in den USA und in Großbritannien neue digitale Plattformen sowie Softwarewerkzeuge für die organisations- und länderübergreifende Zusammenarbeit von Wissenschaftlern entstanden. Wissenschaftliche Einrichtungen bringen dort unter der Flagge von *eScience* und *cyberinfrastructure* Förderprogrammen neue Arbeitsweisen sowie IT-Infrastruktur hervor, die eine globale Vernetzung von Wissenschaftlern und die gemeinsame Nutzung von Rechenressourcen ermöglichen sollen. Die Simulation als etablierte Säule des wissenschaftlichen Erkenntnisgewinns nimmt in diesem Kontext eine zentrale Rolle ein: Simulationswerkzeuge, die verteilte Ausführung von Experimenten im Rechner und die gemeinsame Auswertung von Simulationsergebnissen sind wiederkehrende Themen der neuen Plattformen.

In dieser Arbeit soll herausgearbeitet werden, inwiefern sich die neuen Plattformen, Dienste und Softwarewerkzeuge mit der bestehenden SimTech Workflow-Umgebung integrieren lassen. Besonderes Augenmerk soll dabei auf solchen Angeboten liegen, welche die Nutzung der Plattformen als Laufzeitumgebung für Simulation Workflows ermöglichen.

Der Exzellenzcluster Simulation Technology (SimTech) ist Teil einer Förderlinie der Exzellenzinitiative des Bundes und der Länder zur Förderung von Wissenschaft und Forschung an deutschen Hochschule und arbeitet in interdisziplinären Projekten an neuartigen Werkzeugen für Computersimulationen. Diese Werkzeuge sollen es ermöglichen, skalenübergreifende Simulationen miteinander zu verknüpfen und in eine gemeinsame Entwicklungs- und Ausführungsumgebung zu integrieren. Das Institut für Architektur von Anwendungssystemen befasst sich in SimTech mit der Nutzung von Workflow-Technologie zur Modellierung und Ausführung wissenschaftlicher Simulationen.

1.1 Ziele der Arbeit

Abbildung 1 zeigt die Teilziele dieser Arbeit und die grundsätzliche Vorgehensweise: Zunächst werden wir existierende eScience Plattformen auf Möglichkeiten zur Integration mit der SimTech Workflow-Umgebung *untersuchen*. Für jeden gefundenen Integrationsansatz wird grob die Herangehensweise für eine technische Realisierung sowie Hindernisse, welche die technische Realisierung erschweren, beschrieben.

Die gefundenen Integrationsansätze *bewerten* wir anschließend mittels nachvollziehbarer Kriterien hinsichtlich ihres Nutzens für SimTech. Zusätzlich versuchen wir für alle positiv bewerteten Ansätze die Machbarkeit der Integration im Zeitfenster dieser Diplomarbeit abzuschätzen. Anschließend wird mithilfe dieser Analyse eine konkrete Integrationsaufgabe ausgewählt, geplant und durchgeführt. Dazu ermitteln wir Anforderungen an die Realisierung, und bewerten anhand dieser Anforderungen verschiedene Implementierungsalternativen. Eine entsprechende Softwarelösung wird *konstruiert*, und deren Ergebnisse anschließend diskutiert.

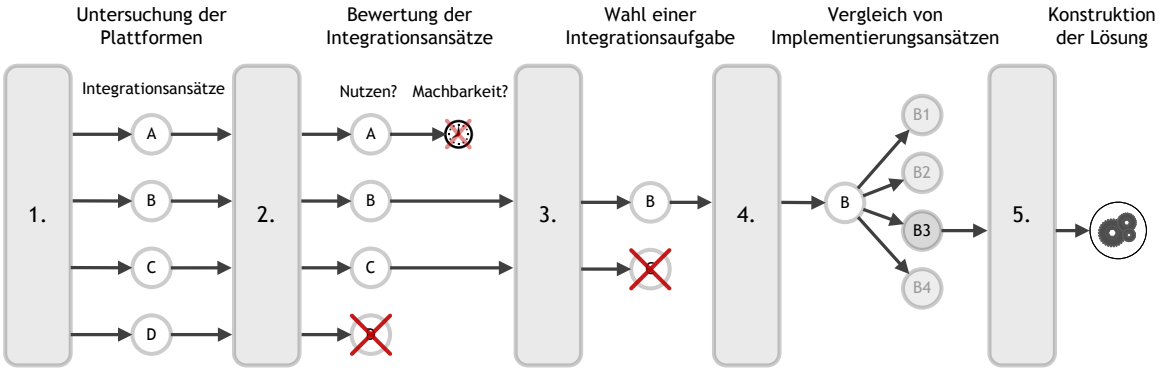


Abbildung 1. Vorgehensweise

1.2 Gliederung

Auf dieses Einleitungskapitel folgt im zweiten Kapitel eine kurze Einführung in die Themengebiete, die wir im Verlauf dieser Ausarbeitung ansprechen werden. Die verwendeten Technologien werden ebenfalls erläutert.

Im dritten Kapitel werden die untersuchten Plattformen nanoHUB.org und myExperiment sowie das Workflow-Management-System Taverna vorgestellt. Für jede Plattform beschreiben wir - noch möglichst wertungsfrei - die entdeckten Integrationsansätze und skizzieren jeweils die grundsätzliche Herangehensweise einer technischen Realisierung.

In Kapitel vier erfolgt die Analyse der gefunden Integrationsansätze im Hinblick auf ihren Nutzen für SimTech und auf die Machbarkeit im Zeitfenster dieser Diplomarbeit. Hier soll möglichst nachvollziehbar dargelegt werden, warum die Wahl auf Integration der Taverna Workflow Engine gefallen ist.

Das fünfte Kapitel dokumentiert Anforderungsanalyse, Entwurf und Implementierung einer Lösung zur Integration der Taverna Workflow Engine.

Im sechsten und letzten Kapitel bewerten wir abschließend, inwiefern die Ziele der Diplomarbeit erreicht wurden. Es wird diskutiert, was an der implementierten Lösung verbessert werden kann und welche weiteren Integrationsschritte im Anschluss an diese Diplomarbeit folgen könnten.

2 Grundlagen

Dieses Kapitel soll das nötige Grundwissen über die angesprochenen Themengebiete vermitteln sowie die verwendeten Technologien beschreiben, soweit es für das Verständnis dieser Ausarbeitung notwendig ist. Die Beschreibungen gehen von einem Vorwissensstand aus, den man in etwa bei einem Studenten der Informatik oder eines ähnlichen Studiengangs im vierten Semester erwarten könnte. Wir verzichten beispielsweise darauf, die Grundlagen der Modelltheorie oder der objektorientierten Programmierung zu wiederholen.

2.1 eScience

In der Forschung nimmt die EDV heute oft eine zentrale Rolle ein. Besonders in Fachgebieten wie der Physik oder den Biowissenschaften beruht der Gewinn neuer Erkenntnisse immer mehr auf komplexen Computerberechnungen und auf der Auswertung großer Datenmengen, die bei Experimenten oder Simulationen entstehen. In diesem Kontext beschreibt Wouters [27] den Begriff *eScience* (auch e-Science oder E-Science) als die Verknüpfung mehrerer Entwicklungen: Erstens, die gemeinsame Arbeit weltweit verteilter wissenschaftlicher Einrichtungen an Forschungsprojekten, um massive Datenmengen, wie sie z.B. vom LHC[1] oder der Kepler Mission[2] produziert werden, in absehbarer Zeit zu verarbeiten. Zweitens, die Nutzung spezialisierter Internet-Plattformen zur Kommunikation und Zusammenarbeit. Drittens, die Freigabe und gemeinsame Nutzung von Rechnerkapazitäten in Form von Grid-Computing. Beim Grid Computing bildet eine Menge von vernetzten Rechnern, die von verschiedenen Inhabern betrieben und kontrolliert werden, eine logische Einheit: das Grid [9]. Die Rechner, aus denen ein Grid besteht, können sich in ihrer geographischen Lage und in ihrer Leistungsfähigkeit sehr unterscheiden. Durch den Einsatz von Middleware werden diese Unterschiede vor den Benutzern des Grids aber verborgen, sodass das Grid nach außen als ein hochverfügbarer Dienst zur Abarbeitung von Computerberechnungen erscheint. Über eine Softwareschnittstelle werden Berechnungsaufgaben an das Grid übermittelt und dann so abgearbeitet, dass sich die Rechenlast möglichst gleichmäßig auf die Rechner des Grids verteilt. Der Zugang zu einem Grid ermöglicht Wissenschaftlern, bei Bedarf die Rechenkapazität eines virtuellen Höchstleistungsrechners zu nutzen.

Ein weiterer wichtiger Teilaspekt der eScience-Bewegung sind die *Werkzeuge* zur Kommunikation, Datenerhebung, Datenverarbeitung und Automatisierung. Die Allgegenwärtigkeit von Software-Werkzeugen bei der Forschungsarbeit hat Auswirkungen auf die Arbeitsweise von Wissenschaftlern: Ingenieurmäßiges Vorgehen wird wichtiger, ebenso die Kooperation mit der Industrie. Nicht zuletzt ist eScience aber auch ein Schlagwort zur Förderung und Ausrichtung wissenschaftlicher Einrichtungen, ursprünglich in britischen Kreisen, seit etwa 2003 aber auch in Deutschland [23, 2].

2.2 Geschäftsprozesse und Workflows

Workflow-Technologie ist die Grundlage für Simulation Workflows. Dieser Abschnitt soll keine erschöpfende Auseinandersetzung mit dem Thema bieten, sondern nur die wichtigsten Grundbegriffe vermitteln. Workflows haben ihren Ursprung in der Automatisierung von Geschäftsprozessen. Ein Geschäftsprozess ist ein wiederholbarer Vorgang innerhalb einer Organisation, mit dem (mindestens) ein wohldefiniertes Ziel erreicht wird, z.B. der Bestellvorgang bei einem Versandhändler. An Geschäftsprozessen können verschiedene Parteien teilnehmen, z.B. ein Kunde, Abteilungen der eigenen

[1] Large Hadron Collider, derzeit weltweit größter Teilchenbeschleuniger
[2] NASA-Projekt zur Suche nach Planeten außerhalb des Sonnensystems über ein Weltall-Teleskop

Organisation und andere Organisationen. Workflows bilden Geschäftsprozesse im Computer ab, wobei Leymann und Roller [17] zwischen drei Dimensionen unterscheiden:

1. *What* - Aus welchen Einzelschritten besteht der Workflow und in welcher Reihenfolge sind sie auszuführen?
2. *Who* - Welche Teilnehmer hat der Workflow und welche Rollen nehmen sie ein?
3. *With* - Welche Ressourcen, Daten, Programme etc. benötigt der Workflow zur Ausführung?

Die Ausführung von Workflows im Rechner geschieht, indem aus einem Workflow-Modell eine Workflow-Instanz erzeugt wird. Die Bezeichnungen „Prozess“ und „Workflow“ werden im realen Sprachgebrauch oft synonym, und je nach Kontext sowohl für das Modell als auch für die Instanz verwendet. Abbildung 2 zeigt dagegen die korrekte Terminologie.

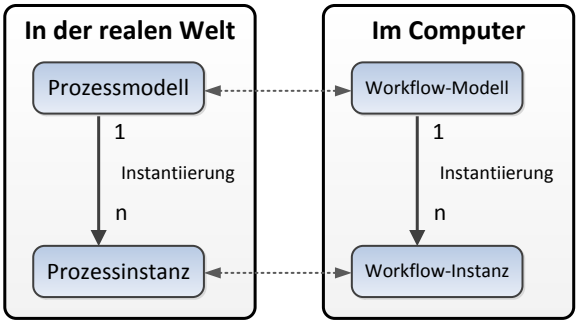


Abbildung 2. Terminologie für Modelle und Instanzen basierend auf [17]

Workflow-Management-Systeme

Ein Workflow-Management-System (WfMS) ist Software zur Verwaltung von Workflow-Modellen und Instanzen. Es lässt sich grob in zwei Bereiche unterteilen: Entwurf und Ausführung. Im Entwurfsbereich werden Workflow-Modelle mit einem Softwarewerkzeug, dem Modeling Tool, modelliert und abgespeichert. Im Ausführungsbereich werden abgespeicherte Workflow-Modelle zur Ausführung vorbereitet und instantiiert. Laufende Workflow-Instanzen können überwacht werden. Zusätzlich sammeln WfMS oft Daten zur Auditierung beendeter Workflow-Instanzen. Zur Speicherung persistenter Daten, wie z.B. den Zustand laufender Workflow-Instanzen, setzen WfMS in der Regel relationale, transaktionale Datenbanksysteme ein.

2.3 Simulation Workflows

In der Wissenschaft wird zunehmend Workflow-Technologie eingesetzt, um wissenschaftliche Arbeitsvorgänge zu automatisieren [27, 10]. Indem Experimente und Simulationen als Workflows modelliert werden, können andere Wissenschaftler sie mit wenig Aufwand wiederholen und anpassen, was die Kollaboration zwischen Wissenschaftlern an unterschiedlichen Standorten fördert. Da im wissenschaftlichen Umfeld viele unterschiedliche Technologien und Software-Werkzeuge zum Einsatz kommen, ist Workflow-Technologie auch als Form der Anwendungsintegration nützlich.

Simulation Workflows sind eine spezielle Form der wissenschaftlichen Workflows, bei denen der Schwerpunkt auf der Ausführung und Auswertung von Computersimulationen liegt. Sie unterscheiden sich in einigen Punkten von Business-Workflows [24]. Wissenschaftliche Workflows sind datenzentriert, während Business-Workflows sich am Kontrollfluss orientieren. Die verarbeitete Datenmenge je Instanz ist in wissenschaftlichen Workflows deutlich größer. Auch die Phasenmodelle (siehe Abbildung 3) unterscheiden sich stark voneinander: Bei Business-Workflows werden die Schritte Modellierung, Installation, Ausführung, Überwachung und Analyse in der Regel von unterschiedlichen Spezialisten ausgeführt, in wissenschaftlichen Workflows übernimmt der Wissenschaftler all diese Aufgaben. Einige

wissenschaftliche WfMS wie Taverna unterscheiden zudem nicht zwischen Installation und Instantiierung von Workflow-Modellen. Der Grund dafür ist, dass Wissenschaftler oft nach dem Prinzip von Versuch und Irrtum vorgehen und daher aus einem Workflow-Modell nicht selten nur eine Instanz erzeugt wird, bevor es Änderungen am Modell gibt. Bei Business-Workflows werden einmal installierte Workflow-Modelle dagegen vielfach instantiiert, bevor Änderungen am Modell eine erneute Installation erfordern.

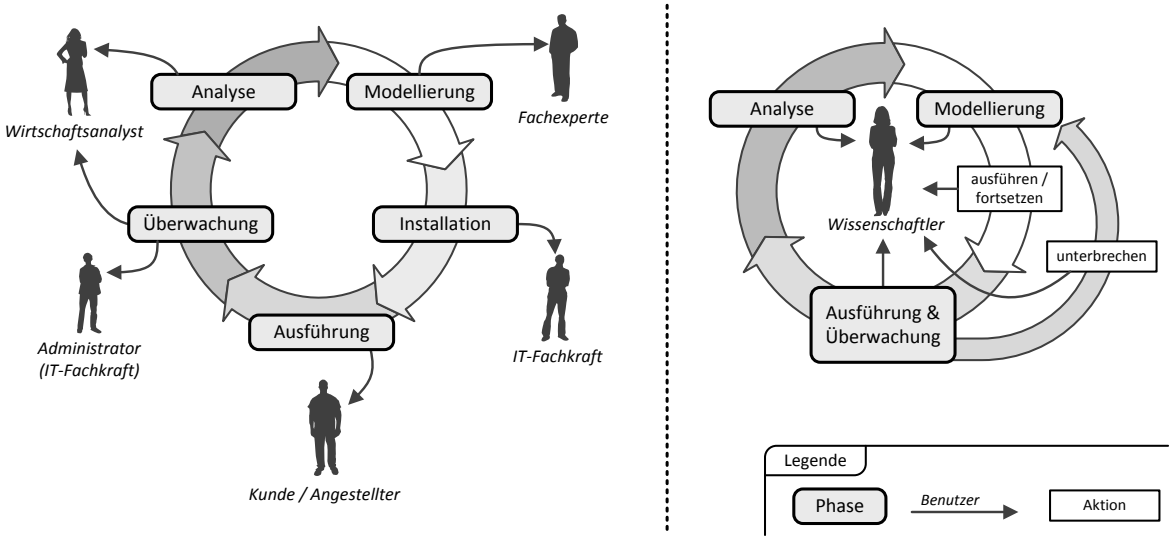


Abbildung 3. Phasenmodell für Business-Workflows (links) und wissenschaftliche Workflows (rechts) [10]

2.4 Technologien

In den folgenden Abschnitten werden die verwendeten Technologien kurz beschrieben. Die Erklärungen zeigen auf, in welchem Zusammenhang die Technologie in dieser Arbeit verwendet wird und warum das Verständnis der Technologie nötig ist.

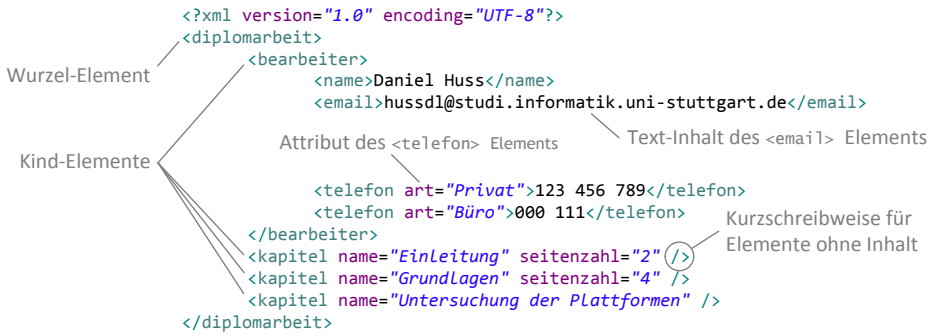
VNC

Virtual Network Computing (VNC) bezeichnet die Steuerung entfernter Rechner über ein Netzwerk mithilfe des Remote Framebuffer Protocols. Dazu überträgt der VNC Client Tastatur- und Mausereignisse an den VNC Server, dieser überträgt im Gegenzug den aktuellen Bildschirminhalt an den VNC Client. Diese Technologie ist aufgrund der geringen Bandbreite vieler Netzwerke und der vergleichsweise hohen Latenz nicht für flüssige Wiedergabe von Videos oder Animationen geeignet, sondern wird hauptsächlich zur Fernwartung von Rechnersystemen eingesetzt. Auf der Plattform nanoHUB[3] wird per VNC die Benutzung von Simulationswerkzeugen im Webbrowser ermöglicht.

XML

Die eXtensible Markup Language (XML) ist ein vom W3C standardisiertes Format, das strukturierte Daten in Form von XML-Dokumenten beschreibt [4]. XML-Dokumente enthalten hierarchische Datenstrukturen, die nach formal spezifizierten Regeln als maschinenlesbarer Text dargestellt werden können. Abbildung 4 zeigt ein einfaches Beispieldokument, in dem einige Daten über diese Ausarbeitung festgehalten sind. Die grundlegenden Bausteine, aus denen XML-Dokumente bestehen, sind dort gekennzeichnet.

[3] `http://nanohub.org`




```xml
<?xml version="1.0" encoding="UTF-8"?>
<diplomarbeit>
    <bearbeiter>
        <name>Daniel Huss</name>
        <email>hussdl@studi.informatik.uni-stuttgart.de</email>
        <telefon art="Privat">123 456 789</telefon>
        <telefon art="Büro">000 111</telefon>
    </bearbeiter>
    <kapitel name="Einleitung" seitenzahl="2" />
    <kapitel name="Grundlagen" seitenzahl="4" />
    <kapitel name="Untersuchung der Plattformen" />
</diplomarbeit>
```

Abbildung 4. XML-Beispieldokument

In der Praxis ist es für die Verarbeitung von XML-Dokumenten wichtig, Annahmen über den Aufbau und den Inhalt eines bestimmten Dokumenten-Typs treffen zu können. Dazu muss z.B. eingeschränkt werden, welche Elemente in diesem Dokumenten-Typ vorkommen dürfen, wie die Elemente angeordnet werden müssen, welchen Inhalt sie besitzen dürfen und welche Attribute die Elemente aufweisen dürfen. Es existieren mehrere Varianten, mit denen man solche Einschränkungen für XML-Dokumente formulieren kann. Für diese Arbeit sind aber lediglich *XML Schema Definitions* (XSD) von Bedeutung [7]. Mit der Einführung von Schemadefinitionen wird XML zur Metasprache [25]: Jede Schemadefinition ist selbst ein XML-Dokument und beschreibt eine Klasse von *gültigen* XML-Dokumenten. Die Menge aller *gültigen* XML-Dokumente für ein bestimmtes Schema bilden eine Sprache. Abbildung 5 zeigt eine solche Schemadefinition für das Beispiel aus Abbildung 4. XML sowie Schemadefinitionen sind für diese Arbeit wichtig, weil andere Technologien wie BPEL und Web services darauf aufbauen, und da wir XML häufig als Dateiformat zur Speicherung hierarchischer Datenstrukturen wiederfinden.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="diplomarbeit">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element ref="bearbeiter" maxOccurs="2" minOccurs="1"/>
                  <xsd:element ref="kapitel" minOccurs="3" maxOccurs="10"/>
            </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="bearbeiter">
      <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="name" type="xsd:string"/>
                  <xsd:element name="email" type="xsd:string"/>
                  <xsd:element name="telefon" minOccurs="0" maxOccurs="unbounded">
                        <xsd:complexType>
                              <xsd:simpleContent>
                                    <xsd:extension base="xsd:string">
                                          <xsd:attribute name="art" type="xsd:string"/>
                                    </xsd:extension>
                              </xsd:simpleContent>
                        </xsd:complexType>
                  </xsd:element>
            </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="kapitel">
      <xsd:complexType>
            <xsd:attribute name="name" type="xsd:string" use="required"/>
            <xsd:attribute name="seitenzahl" type="xsd:int" use="optional"/>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Abbildung 5. Mögliches XML Schema für das Beispieldokument aus Abbildung 4

Web Services

Der folgende Abschnitt ist der Versuch, einen Kompromiss zwischen den Definitionen von Weerawarana et al. [25] und dem W3C [3] zu finden: Web services sind eigenständige Softwarekomponenten, die ihre Funktionalität als Dienst über ein Netzwerk bereitstellen. Web service Technologie beschäftigt sich nicht mit der konkreten Implementierung von Funktionalität, sondern liefert ein plattform- und herstellerunabhängiges Modell für Beschreibung, Auffindung und Abruf der *Schnittstellen* von Softwarekomponenten. Implementierungsdetails und Eigenheiten ihrer Laufzeitumgebung verbergen Web sevices hinter öffentlichen Schnittstellen, die durch den Austausch von Nachrichten angesprochen werden. Dadurch wird eine lose Kopplung zwischen Web services und ihren Abrufern erzwungen. Jeder Web service besitzt eine Identität und eine maschinenlesbare Beschreibung, die es einem Rechner ermöglicht, die Funktionalität des Web services abzurufen. Web services haben im Gegensatz zu anderen Dienstausprägungen kein Konzept von Abschaltung oder Pausierung: wenn ein Web service verfügbar bzw. erreichbar ist, dann ist er auch angeschaltet. Obwohl der Name es suggeriert, müssen Web services nicht unbedingt über das Web erreichbar sein.

Die Integrationslösung, welche im Rahmen dieser Arbeit konstruiert wird, soll als Web service bereitgestellt werden. Für diese Arbeit ist nur eine bestimmte technische Ausprägung von Web services von Bedeutung: Sie basieren auf XML, verwenden SOAP/HTTP zur Kommunikation und werden mit WSDL beschrieben.

SOAP

Web services kommunizieren mit der Außenwelt, indem sie Nachrichten empfangen und versenden. SOAP ist ein vom W3C standardisiertes, erweiterbares Protokoll zur Zustellung solcher Nachrichten ausgehend von ihrem Absender über eine Menge von Zwischenstationen bis zu ihrem endgültigen Empfänger [11]. SOAP-Nachrichten sind XML-Dokumente mit dem Wurzelelement `Envelope`, darunter können im Element `Header` Metainformationen zur Nachricht abgelegt werden, die eigentlichen Nutzdaten der Nachricht befinden sich im Element `Body` (s. Abbildung 6). SOAP gibt zwar keinen Transportmechanismus für die Datenübermittlung zwischen zwei Stationen vor, in der Praxis wird aber häufig HTTP eingesetzt.

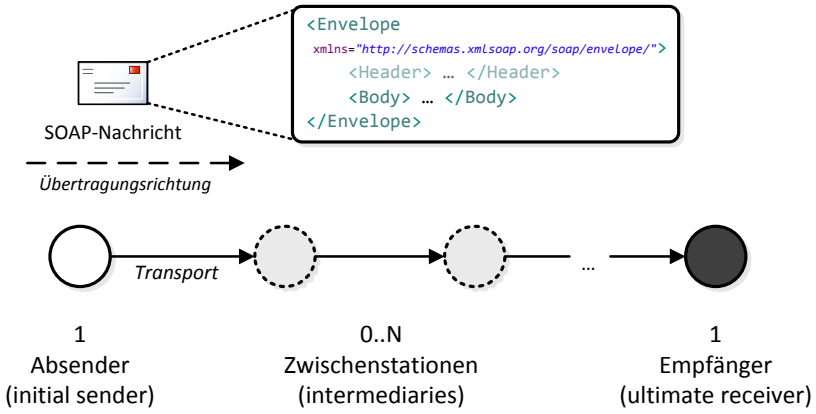


Abbildung 6. Zustellung einer SOAP-Nachricht

WSDL

Die maschinenlesbare Beschreibung eines Web services sollte alle Informationen enthalten, die man benötigt, um den Web service zu benutzen. Die Web Services Description Language (WSDL) übernimmt diese Aufgabe: Sie ist ein XML-Format zur Beschreibung der *funktionalen* Aspekte von Web services [25].

WSDL-Dokumente bestehen aus einem abstrakten, wiederverwendbaren Teil und einem konkreten Teil. Im abstrakten Teil wird beschrieben, *was* der Web service kann: welche Operationen verfügbar sind, welche Eingaben und Ausgaben sie haben. Im konkreten Teil wird beschrieben, *wo* der Web service gefunden werden kann und über welchen Aufrufmechanismus der Web service benutzt wird.

Die aktuelle Version des WSDL-Standards ist zwar 2.0, da aber BPEL mit dieser Version nicht kompatibel ist, wird in dieser Arbeit ausschließlich WSDL 1.1 verwendet. Abbildung 7 zeigt die wichtigsten Elemente, aus denen WSDL-Dokumente der Version 1.1 bestehen: PortTypes sind abstrakte Schnittstellendefinitionen, die eine oder mehrere Operationen enthalten. Jede Operation besitzt eine Eingabe oder eine Ausgabe oder beides. Zur Ausnahmebehandlung können Operationen zusätzlich eine Menge von Fehlerzuständen (faults) definieren. Jeder Eingabe, Ausgabe und jedem Fehler wird eine Nachricht (message) zugeordnet. Das Binding beschreibt, wie Operationen eines bestimmten PortTypes aufgerufen werden und definiert Kodierung sowie Transport von Nachrichten über das Netzwerk. Jedes Binding bezieht sich auf genau einen PortType, aber für jeden PortType können mehrere Bindings definiert werden. Ein mögliches Binding, und das in dieser Arbeit ausschließlich verwendete, ist das SOAP-Binding im Document-Literal-Stil. Ein Port gibt den Ort einer Implementierung eines PortType an und nennt das Binding, mit dem diese Implementierung aufgerufen werden kann. Das Service-Element fasst schließlich einen oder mehrere Ports zu einem identifizierbaren Web service zusammen. Nicht abgebildet ist das Element `types`, welches die verfügbaren Datenstrukturen für Nachrichteninhalte beschreibt. In der Regel wird XML Schema als Typsystem verwendet. Ebenfalls nicht abgebildet ist das Element `import`, welches erlaubt, sich innerhalb eines WSDL-Dokuments auf den Inhalt anderer WSDL-Dokumente zu beziehen.

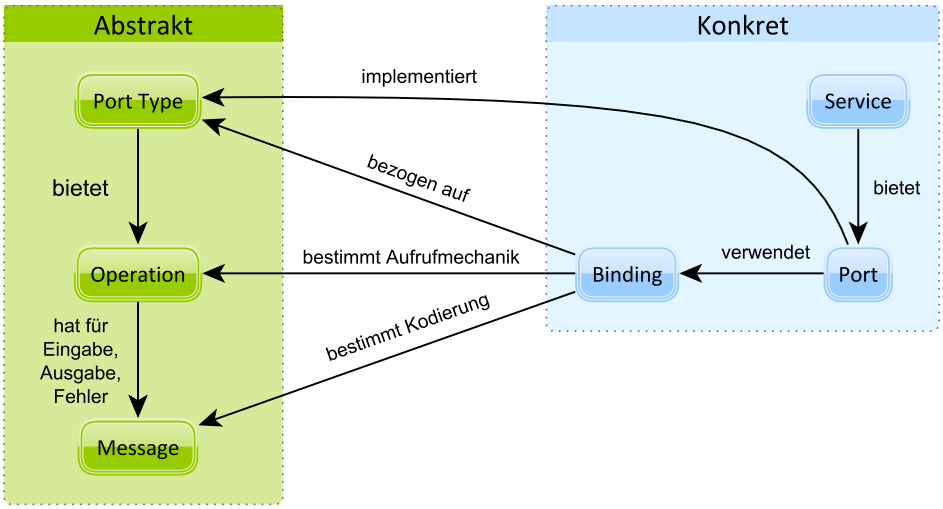


Abbildung 7. WSDL 1.1 - Elemente und deren Beziehungen

BPEL

Die Web Services Business Process Execution Language 2.0 (BPEL) ist ein erweiterbares, XML-basiertes Metamodell für Workflow- und Prozessmodelle [1]. BPEL ist das Metamodell der SimTech Simulation Workflows. Gleichzeitig wird BPEL aber auch in der Industrie eingesetzt, da es sich um einen OASIS[4]-Industriestandard handelt.

BPEL-Prozesse nutzen die Funktionalität bestehender Web services. Das Metamodell bietet selbst nur generische Strukturen zur Steuerung des Kontrollflusses, zur Interaktion mit Web services und zur Manipulation von XML-Datenstrukturen [25]. Zur Modellierung des Kontrollflusses können sowohl blockbasierte Elemente wie Sequenzen, Schleifen oder If-Abfragen, als auch Graphen verwendet

[4] Organization for the Advancement of Structured Information Standards: `https://www.oasis"=open.org`

werden. Der Datenfluss eines BPEL-Prozesses ist implizit durch den modellierten Kontrollfluss gegeben. Die im Rahmen dieser Arbeit erstellte Software zur Integration der Taverna Workflow-Engine ist darauf ausgerichtet, von BPEL-Prozessen benutzt zu werden, daher werden wir in Kapitel 5 die Interaktion von Web services mit BPEL-Prozessinstanzen genauer betrachten.

Java

Java ist eine Software-Plattform und eine populäre Programmiersprache, die im Kontext dieser Diplomarbeit an zwei Stellen relevant ist: Einerseits wird das Taverna WfMS mit Java entwickelt, andererseits erfolgt die Konstruktion der Integrationslösung in Java Standard Edition, Version 6. Wir gehen im Folgenden lediglich von Grundkenntnissen der Syntax, der Verwendung von Annotationen sowie der häufig verwendeten Datentypen wie `String` und `byte[]` aus. Ebenfalls von Bedeutung ist das JAR-Dateiformat, in dem Java-Programme und Softwarebibliotheken ausgeliefert werden. Eine JAR-Datei ist ein ZIP-Archiv, das eine Menge von Java-Klassen und sonstigen Dateien enthält, die während der Ausführung einer Java-Anwendung geladen werden können. In einem speziellen Ordner namens *META-INF* werden Metadaten abgelegt, welche z.B. die digitale Signatur des Inhalts einer JAR-Datei ermöglichen.

JAXB

Die Java API for XML Binding (JAXB) ist eine Standard-Softwarebibliothek zur bidirektionalen Abbildung von XML Schemadefinitionen auf Java-Klassen [14, 28]. Über mitgelieferte Tools lassen sich aus einer bestehenden Schemadefinition eine Menge von annotierten Java-Klassen generieren. Jede generierte Klasse repräsentiert ein Element oder einen Typ aus der Schemadefinition. Umgekehrt lassen sich aber auch bestehende Java-Klassen mit JAXB-Annotationen anreichern, um die statische Struktur der Klasse auf XML Schema abzubilden. In beiden Fällen können über JAXB Instanzen der annotierten Klassen aus eingelesenen XML-Dokumenten erzeugt werden. Instanzen der annotierten Klassen können wiederum zu XML-Dokumenten serialisiert werden. Natürlich besteht auch die Möglichkeit, solche Objekte programmatisch zu erzeugen oder zu verändern.

JAXB wird in dieser Arbeit in Version 2.1 bei der Konstruktion der Integrationslösung verwendet. Es wird ausschließlich der Schema-zu-Java-Ansatz eingesetzt, d.h. es werden XML Schemadefinitionen modelliert und Java-Klassen daraus generiert. Der generierte Code wird nicht manuell geändert, sondern bei jeder Anpassung der Schemadefinition müssen die Java-Klassen erneut generiert werden.

JAX-WS

Die Java API for XML Web services (JAX-WS) ist eine Standard-Softwarebibliothek für die Arbeit mit Web services unter Java[16]. JAX-WS unterstützt den Entwickler sowohl bei der Verwendung bestehender Web services in Java-Anwendungen als auch bei der Implementierung neuer Web services. Um die Datenstrukturen aus XML-basierten Web services auf Java-Klassen abzubilden, wird JAXB verwendet. Wie auch bei JAXB gibt es bei JAX-WS zwei grundsätzliche Herangehensweisen: Entweder werden aus einem bestehenden WSDL-Dokument Java-Klassen generiert, oder es wird bereits existierender Java-Code mit JAX-WS-Annotationen versehen.

Alle Web services, die im Rahmen dieser Diplomarbeit konstruiert werden, setzen JAX-WS in der Version 2.1 ein. Bei der Implementierung neuer Web services wird, außer zu Testzwecken, immer zuerst ein WSDL-Dokument modelliert und der Java-Code aus diesem Dokument generiert (siehe Abbildung 8). Die generierten Service Clients sind zu ignorieren oder zu löschen, weil der generierte Code Mängel aufweist: Er verweist per absoluter Pfadangabe auf die WSDL-Datei, aus welcher der Client generiert wurde und ist somit an einen einzigen Rechner gekoppelt.

Apache Maven

Maven ist u.a. ein Werkzeug für die automatisierte Erstellung von Softwareartefakten aus Quellcode und wird hauptsächlich zur Verwaltung von Java-Projekten eingesetzt. Sowohl Taverna als auch die

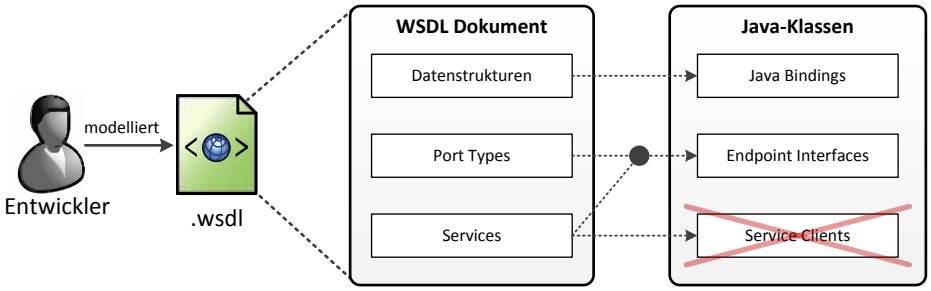


Abbildung 8. Verwendung von JAX-WS in dieser Arbeit

implementierte Integrationslösung verwenden Maven in der Version 2 oder 3. Einige Begriffe aus der Maven-Terminologie, die für das Verständnis der Ausarbeitung wichtig sind, werden hier vorgestellt.

Projekt Ein Projekt ist eine kohärente Menge von Programmcode-Dateien, sonstigen Ressourcen sowie Informationen über das Projekt selbst. Aus einem Maven-Projekt geht in der Regel ein Softwareartefakt hervor, z.B. ein ausführbares Programm, eine Programmbibliothek oder eine Website. Jedes Maven-Projekt wird durch eine XML-Datei namens `pom.xml` beschrieben, welche sich im Stammverzeichnis des Projekts befindet. Die Beschreibung enthält u.a. Name, Version, Besitzer und Art des Projekts, den Typ des Hauptartefakts, Abhängigkeiten von anderen Maven-Projekten sowie eine Beschreibung der Aktivitäten, die zur Erstellung der Artefakte durchgeführt werden müssen.

Build Der Buildprozess ist die Menge und Reihenfolge der Aktivitäten, die Maven durchführen muss, um die Zielartefakte aus dem Quellcode und anderen Ressourcen eines Projekts zu erstellen. Dazu gehört in der Regel die Kompilierung des Quellcodes in Binärdateien, Ausführung von Softwaretests, Paketierung der Binärdateien, digitale Signatur der Artefakte, etc. Aus jedem Maven-Projekt geht genau ein Hauptartefakt hervor, z.B. das ausführbare Programm in Form einer JAR-Datei. Zusätzlich können weitere Nebenartefakte entstehen, z.B. die Dokumentation oder der Quellcode.

Repository Maven-Artefakte werden in Repositories installiert, um sie als Abhängigkeit für andere Projekte verfügbar zu machen. Dazu wird einfach die Artefaktdatei in ein bestimmtes Verzeichnis kopiert. Der genaue Verzeichnispfad lässt sich anhand des Namens, Typs und Versionsnummer des zugehörigen Maven-Projekts eindeutig bestimmen. Abhängigkeiten, die im lokalen Repository nicht verfügbar sind, werden während des Builds bei Bedarf aus einem entfernten Repository per HTTP nachgeladen und in das lokale Repository kopiert.

2.5 SimTech Workflow-Umgebung

Der Ausgangspunkt unserer Integrationsaufgabe ist die SimTech Workflow-Umgebung (Abbildung 9). Sie besteht im Wesentlichen aus einem Workflow Modeling Tool (im Bild links) und einer Serverseite (im Bild rechts). Der Datenaustausch zwischen Modeling Tool und der Serverseite findet zum Teil direkt, zum Teil über Apache ActiveMQ statt. ActiveMQ ist ein Dienst zur Übertragung von Nachrichten zwischen verschiedenen Anwendungen.

Serverseite

Auf der Serverseite werden Dienste und sonstige Software betrieben, welche die Modellierung und Ausführung von Simulation Workflows unterstützen. Die Laufzeitumgebung für diese Dienste ist der Apache Tomcat Web Container. ODE-PGF ist eine modifizierte Version der Apache ODE BPEL-Engine und ist für die Ausführung von BPEL-Prozessen zuständig. Apache Axis2 ist ein Container für Web services, der u.a. die Implementierung einiger Simulation-Workflow-Aktivitäten beinhaltet.

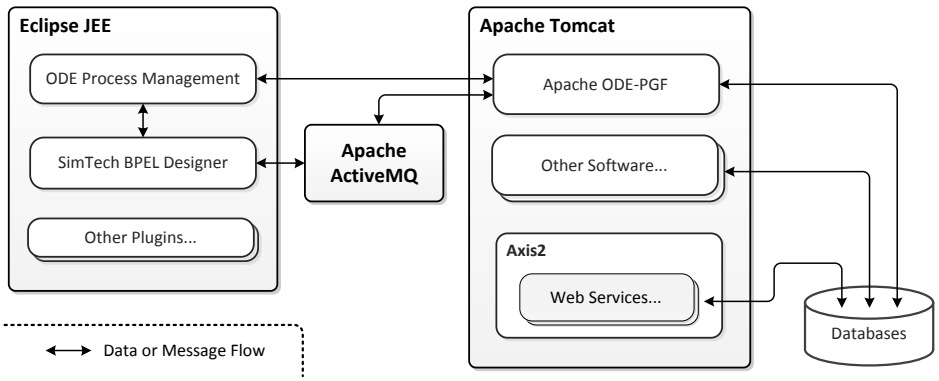


Abbildung 9. Vereinfachte Darstellung der SimTech-Architektur [12]

Zur Speicherung persistenter Daten kommen die Datenbanksysteme MySQL und PostgreSQL zum Einsatz.

Modeling Tool

Das Modeling Tool basiert auf der Eclipse Rich Client Platform und deren Plugin-System, die SimTech Werkzeuge zur Modellierung von Simulation Workflows sind als Eclipse-Plugins realisiert. Im Mittelpunkt steht der SimTech BPEL Designer, der eine modifizierte Version des Eclipse BPEL Designers ist. Der SimTech BPEL Designer visualisiert BPEL-Prozessmodelle und erlaubt die Bearbeitung der Prozessmodelle über eine graphische Oberfläche. Um die modellierten Prozesse aus der Eclipse-Entwicklungsumgebung heraus zu instantiieren, wird das ODE Process Management Plugin verwendet. Es erlaubt die Verwaltung einer lokalen Installation der Apache ODE. Indem das Modeling Tool mit der BPEL-Engine kommuniziert, lassen sich über die graphische Oberfläche auch laufende BPEL-Prozessinstanzen überwachen oder debuggen, die Ergebnisse beendeter Instanzen können analysiert werden.

3 Untersuchung der eScience-Plattformen

In diesem Teil der Arbeit werden die bereits vor Beginn dieser Arbeit ausgesuchten Plattformen nanoHUB.org und myExperiment sowie das wissenschaftliche WfMS Taverna vorgestellt. Anschließend betrachten wir die in Frage kommenden Integrationsansätze, wobei für jeden Integrationsansatz ein grobes Konzept zur technischen Realisierung umrissen werden soll. Eine ausführliche Bewertung der gefundenen Integrationsansätze erfolgt separat in Kapitel 4.

Die Integrationsmöglichkeiten lassen sich, wie in Abbildung 3 dargestellt, grundsätzlich in zwei Kategorien unterteilen: Zum einen können nützliche Funktionen oder Daten, welche von den Plattformen angeboten werden, in das bestehenden SimTech-System aufgenommen oder dort verfügbar gemacht werden. Andererseits besteht die Möglichkeit, SimTech-Infrastruktur auf den Ressourcen des Plattform-Betreibers zur Verfügung zu stellen. Beide Richtungen sind Gegenstand dieser Arbeit.

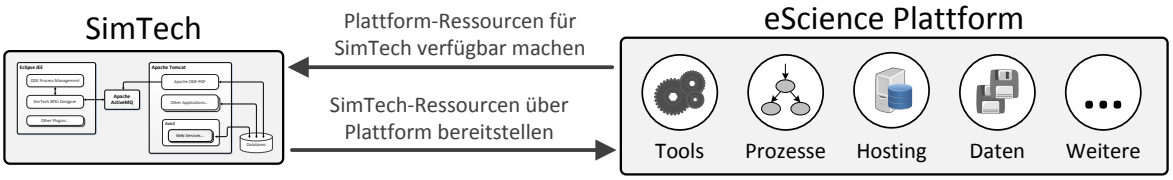


Abbildung 10. Mögliche Ausrichtung der Integrationsansätze

Zur allgemeinen Vorgehensweise bei der Untersuchung gibt es zwei Anmerkungen: Integrationsansätze, bei denen die Plattform offensichtlich nicht im Sinne ihres Betreibers benutzt wird oder bei denen gegen die Nutzungsbedingungen verstoßen wird, kommen nicht in Betracht. Ebenfalls nicht Teil dieser Arbeit sind solche Integrationsvorhaben, die eine besondere Kooperationsvereinbarung der Universität Stuttgart mit dem jeweiligen Plattform-Betreiber benötigen würden. Wir beschränken uns im Folgenden daher auf jene Teile der Plattformen, die für jeden Plattform-Benutzer zugänglich sind.

3.1 nanoHUB

nanoHUB.org[5] ist eine Web-Plattform für Wissenschaftler und Studenten mit dem Schwerpunkt Nanotechnologie. Die Nutzer der Plattform stellen auf nanoHUB Ressourcen mit Bezug zur Nanotechnologie zur Verfügung: Interaktive Simulationstools, Präsentationen, Lehrmaterialien und sonstige Dateien wie Quellcode oder Kalkulationstabellen. Für sämtliche Inhalte, die von Benutzern hochgeladen werden, fordern die Betreiber ein uneingeschränktes Nutzungsrecht ein[6].

Die interaktiven Simulationstools sind das herausragende Merkmal der Plattform. Sie erlauben dem Benutzer, aus dem Webbrowser heraus rechenintensive Operationen auf Grid-Ressourcen der Plattform-Betreiber und deren Partnern auszuführen. Des Weiteren bietet nanoHUB einige Social-Networking-Funktionen wie ein persönliches Profil, Nachrichtenaustausch mit anderen Benutzern, Gruppen und Kontakten. Ein Teil dieser Funktionen ist auch ohne Registrierung abrufbar. Für den Upload von Dateien sowie für die Benutzung der Simulationstools ist jedoch die Anmeldung mit einem Benutzerkonto erforderlich. Die Registrierung eines neuen Benutzerkontos ist kostenlos und durch ein CAPTCHA[7] vor Automatisierung geschützt.

nanoHUB wird seit 2002 vom Network for Computational Nanotechnology (NCN) entwickelt und betrieben [15]. Das NCN ist ein Netzwerk US-amerikanischer Universitäten, in welchem die Purdue University als Betreiber von nanoHUB.org eine führende Rolle einnimmt. Andere Mitglieder des

[5] `http://nanohub.org`

[6] `https://nanohub.org/legal/license`; `http://archive.is/adGAU`

[7] Completely Automated Public Turing test to tell Computers and Humans Apart - Sicherheitsmechanismus, der erzwingen soll, dass ein Vorgang von einem Menschen ausgeführt wird.

NCN wie die University of Illinois oder das Massachusetts Institute of Technology tragen zur Plattform hauptsächlich bei, indem sie Simulationstools, Präsentationen und Lehrmaterialien bereitstellen. Geldgeber des NCN ist die National Science Foundation (NSF), welche die Plattform im Zeitraum von 2002 bis 2010 mit knapp 14 Mio. USD gefördert hat. Für den Zeitraum von 2013 bis voraussichtlich 2017 werden ca. weitere 14 Mio. USD zur Verfügung stehen[8].

Abbildung 11 zeigt die grobe Funktionsweise der Plattform: Benutzer interagieren mit dem System hauptsächlich per Webbrowser, auf der nanoHUB.org Website werden die Inhalte über ein Content-Management-System (CMS) präsentiert. Auch die Simulationstools werden im Webbrowser bedient. Der Zugriff auf die Simulationstools erfolgt durch ein Java-Applet, welches einen VNC-Client implementiert. Das Applet stellt eine Verbindung zu einem virtuellen Desktop her, welcher auf den Rechnern der Purdue-Universität oder deren Partnern ausgeführt wird. Für Berechnungen, die von den Simulationstools durchgeführt werden, steht also die Kapazität des entfernten Rechners zur Verfügung. Bis zu drei solcher VNC-Sitzungen lassen sich je Benutzer gleichzeitig starten. Eine begonnene Sitzung kann pausiert und später wiederaufgenommen werden.

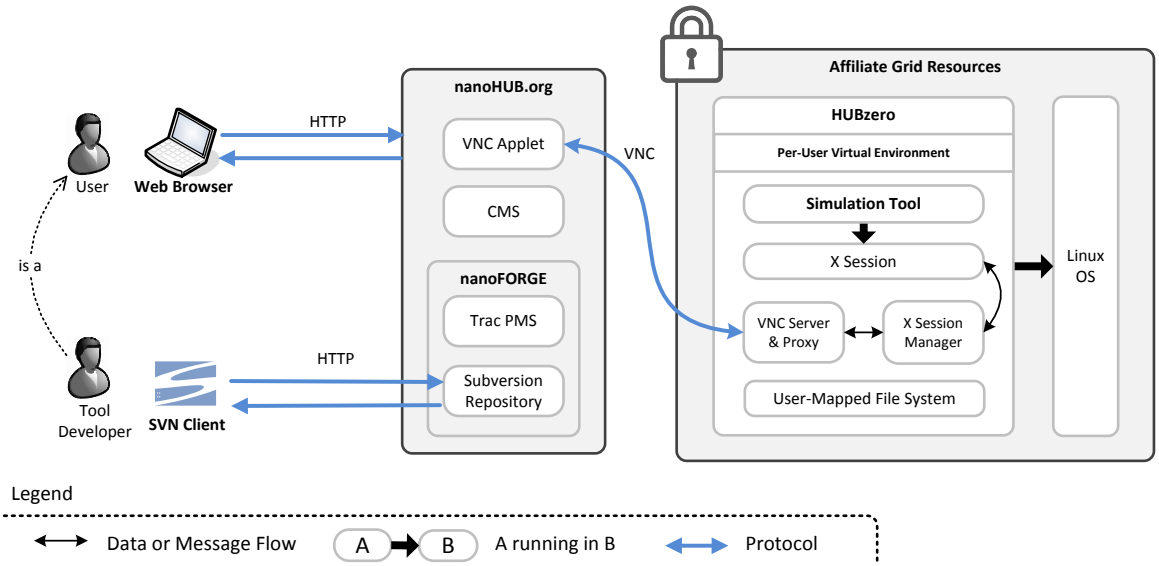


Abbildung 11. Aufbau der nanoHUB Plattform

Für jeden im System registrierten Benutzer wird serverseitig bei Bedarf ein isolierter, virtueller Linux-Rechner angelegt, in dem der virtuelle Desktop läuft. In dessen virtuellen Dateisystem werden auch Tool-Sitzungen abgespeichert und Simulationsergebnisse abgelegt. Die eingesetzte Virtualisierungssoftware ist OpenVZ[9]. Benutzer können Simulationstools zur Plattform beitragen, sie nehmen dann die Rolle eines Tool-Entwicklers ein und erhalten Zugriff auf die nanoFORGE, einen gesonderten Bereich der Plattform. Die nanoFORGE bietet Tool-Entwicklern Projektverwaltung via Trac[10] sowie Zugriff auf ein Subversion-Repository je Simulationstool.

HUBzero

nanoHUB.org basiert auf dem Softwarestack HUBzero [18]. Bei HUBzero handelt es sich um eine wiederverwendbare Sammlung von Tools und Middleware, mit der sich eine Plattform wie nanoHUB

[8] Zugehörige Förderungen der NSF:

- `http://www.nsf.gov/awardsearch/showAward?AWD_ID=0228390`; `http://archive.is/W881N`
- `http://www.nsf.gov/awardsearch/showAward?AWD_ID=1227110`; `http://archive.is/2QQlA`

[9] `http://openvz.org`

[10] Web-basierte Software zur Projektverwaltung: `http://trac.edgewall.org/`

aufbauen und warten lässt. Um eine HUBzero-basierte Plattform zu skalieren, wird eine Menge von Rechnern (Nodes), auf denen die HUB-Software installiert ist, zu einem Cluster verbunden. Dazu teilt ein Administrator dem System die Adressen bzw. Netzwerknamen dieser Rechner per Weboberfläche oder Kommandozeilenprogramm mit.

Zum Teil ist HUBzero eine Weiterentwicklung des Purdue University Network Computing Hub aus dem Jahr 1996 [13]. Alle Komponenten, die in HUBzero zum Einsatz kommen, sind freie Software. In Abbildung 12 ist zu sehen, wie diese Komponenten aufeinander aufbauen: Die Grundlage bildet das Linux-Betriebssystem, eine Virtualisierungssoftware stellt die nötige Isolation zwischen verschiedenen Benutzern und deren virtuellen Desktops her. Über eine entsprechende Firewall-Konfiguration wird sicherheitshalber verhindert, dass Simulationstools eingehende TCP/IP Verbindungen oder UDP-Pakete empfangen können. Eine Clustering-Komponente sorgt für die Synchronisierung verschiedener HUBzero-Nodes und ermöglicht so die Skalierung des Systems, indem einfach die Software auf jedem Rechner im Cluster installiert wird. Um die von jedem Benutzer verbrauchten Ressourcen wie Speicherplatz und CPU-Zeit zu begrenzen, werden Quotas benutzt. Zur Ausführung von Simulationstools und deren Bedienung im Webbrowser erzeugt das System X[11]-Sessions und leitet sie per VNC an den Browser weiter. Das Rappture-Framework ermöglicht hierbei die Definition von graphischen Oberflächen für bestehende Kommandozeilen-Simulationsprogramme. Auf diesen Funktionen setzt schließlich ein Content-Management-System auf, welches auf dem Joomla-Framework basiert.

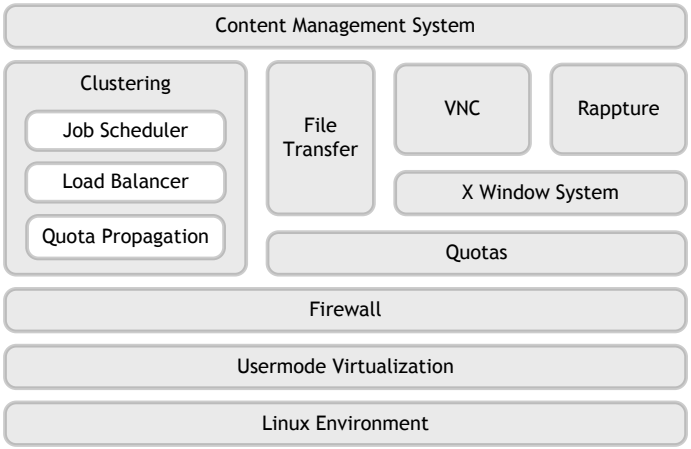


Abbildung 12. HUBzero Software Stack

Neben nanoHUB existieren weitere Plattformen, die auf HUBzero aufbauen, z.B. pharmaHUB[12] und NEEShub[13]. Bei HUBzero handelt es sich allerdings um eine reine Softwarelösung, d.h. die Ressourcen, welche z.B. zur Ausführung der Simulationstools benötigt werden, muss der Betreiber selbst bereitstellen. Gemeinnützige Organisationen können auf Anfrage direkt bei der Purdue University Hosting sowie Training der Mitarbeiter erhalten[14].

Rappture

Ein erwähnenswertes Teilprojekt der nanoHUB/HUBzero-Plattform ist die **R**apid **App**lication Infrastruc**ture** : ein Framework zur Kapselung von Programmen, die selbst keine graphische Benutzerschnittstelle haben. Rappture übernimmt die Generierung einer GUI, das Sammeln der zum Start nötigen Benutzereingaben, den Aufruf des Programms und die Darstellung der Ausgabe. Für jedes zu kapselnde Programm legt der Entwickler hierfür eine XML-Konfigurationsdatei `tool.xml` an, in der

[11] Basis für graphische Oberflächen unter Linux
[12] Plattform für Pharmatechnik: `https://pharmahub.org/`
[13] Plattform für Erdbebentechnik: `http://nees.org/`
[14] http://hubzero.org/hosting/purdue

u.a. die möglichen Ein- und Ausgaben des Programms beschrieben werden. Falls die Kommandozeilenschnittstelle zur Übergabe der Benutzereingaben und der Programmausgabe nicht ausreicht, existiert eine Rappture-API, die explizit aus dem betroffenen Programm heraus angesteuert werden kann. Das erfordert natürlich, dass der Entwickler Zugriff auf den Quellcode des gekapselten Programms hat. Anwendungen, die auf Rappture aufbauen, können auf nanoHUB.org und anderen HUBzero-basierten Portalen mit wenig Aufwand installiert werden. Rappture ist das bevorzugte Entwicklungswerkzeug für Simulationstools, das man auf nanoHUB.org findet (siehe Abbildung 13).

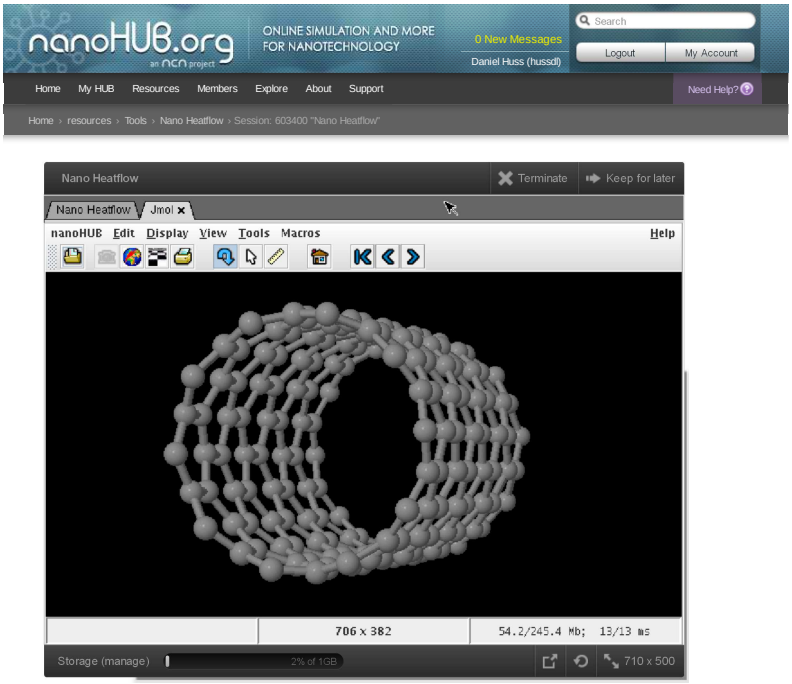


Abbildung 13. Ausführung eines Rappture-getriebenen Simulationstools auf nanoHUB.org

Integration mit nanoHUB

In den folgenden Abschnitten betrachten wir die gefundenen Integrationsansätze für nanoHUB.

Ausführung von SimTech-Anwendungen auf nanoHUB.org

Was nanoHUB von anderen Plattformen unterscheidet, sind vor allem die serverseitig ausgeführten, interaktiven Simulationstools, die per VNC bedient werden. Jeder registrierte Benutzer kann Projekte für neue Simulationstools anlegen, Quellcode und Binärdateien hochladen und seine Tools veröffentlichen. Es liegt daher nahe, zu prüfen, inwiefern sich dieses Angebot nutzen lässt, um SimTech-Anwendungen auf nanoHUB auszuführen.

Abbildung 14 zeigt den Ablauf einer neuen Tool-Registrierung: Um eigene Simulationstools auf nanoHUB anzulegen, wird auf Anfrage in einem separaten Bereich (nanoFORGE) ein Subversion-Repository sowie ein Trac-Projekt eingerichtet. Der Tool-Entwickler muss seinen Quellcode in dieses Repository hochladen, unabhängig davon, ob er sein Tool als Open-Source oder Closed-Source veröffentlichen möchte. Die Installation des Tools erfolgt ebenfalls manuell auf Anfrage. Ein nanoHUB-Mitarbeiter prüft dann den hochgeladenen Code und führt den Build-Vorgang durch. Wenn es sich um eine Rappture-Anwendung handelt, kann man davon ausgehen, dass keine besondere Dokumentation für die Installation nötig ist. Ansonsten muss der Tool-Entwickler die nötigen Schritte für den Build im Trac Projekt-Wiki für den nanoHUB-Mitarbeiter vorher nachvollziehbar dokumentieren. Ändert

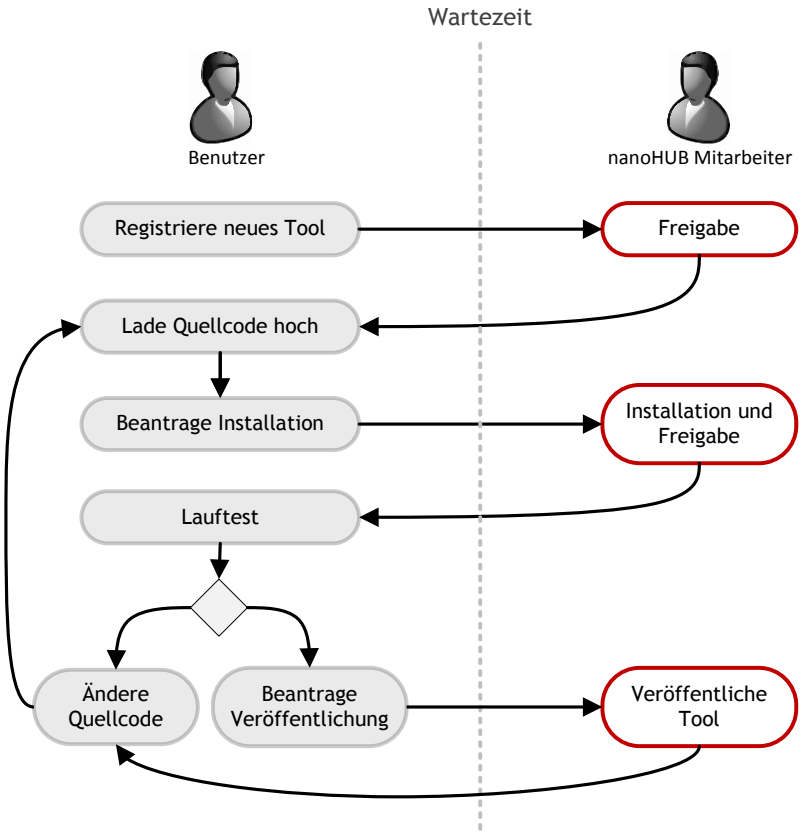


Abbildung 14. Registrierung und Installation eines neuen Simulationstools

der Entwickler etwas am Tool-Quellcode, muss erneut die Installation beantragt werden. Ist der Entwickler mit der Installation zufrieden, beantragt er die Veröffentlichung des Tools. Es kann immer nur eine Version des Tools veröffentlicht sein.

Weil dieser Prozess eher träge ist, existiert für die Anwendungsentwicklung zusätzlich die Möglichkeit, einen „Workspace“ einzurichten. Dabei handelt es sich um einen kompletten, virtuellen Linux-Desktop inklusive IDE, auf den man via VNC zugreift. Über den Workspace hat der Entwickler Zugriff auf seine Software und kann Lauftests durchführen, bevor das Tool von einem nanoHUB-Mitarbeiter installiert wird. Dem gewöhnlichen nanoHUB-Benutzer steht dieser Workspace allerdings nicht zur Verfügung[15].

Um SimTech-Dienste und Anwendungen auf nanoHUB auszuführen, muss der zuvor beschriebene Tool-Registrierungsprozess mindestens einmal durchgeführt werden. Bei jeder Änderung muss außerdem erneut die Installation und die Freigabe durch einen nanoHUB-Mitarbeiter angefordert werden. Die Ausführung von Serverdiensten und lange laufenden Programmen ist allerdings problematisch:

- Die Lebensdauer aller serverseitig ausgeführten Programmen ist an die zugehörige VNC-Sitzung gekoppelt. Wird die VNC-Sitzung im Browser beendet oder pausiert, stoppt auch die serverseitige Ausführung des Simulationstools.
- Die Architektur der Plattform ist darauf ausgerichtet, *interaktive* Programme auszuführen.
- Eingehende Netzwerkverbindungen sind nicht möglich. Ob serverseitige Programme ausgehende Verbindungen zum Internet herstellen können, ist unklar[16] (siehe Abbildung 15).
- Die Verwendung der Plattform als Hoster für Web services ist mit hoher Wahrscheinlichkeit nicht im Sinne der Betreiber und würde schnell unterbunden werden.

[15] Quelle: `https://nanohub.org/tools/workspace/`
[16] Eine Supportanfrage sowie eine öffentliche Frage auf der nanoHUB Website blieben unbeantwortet: `https://nanohub.org/answers/question/1210`

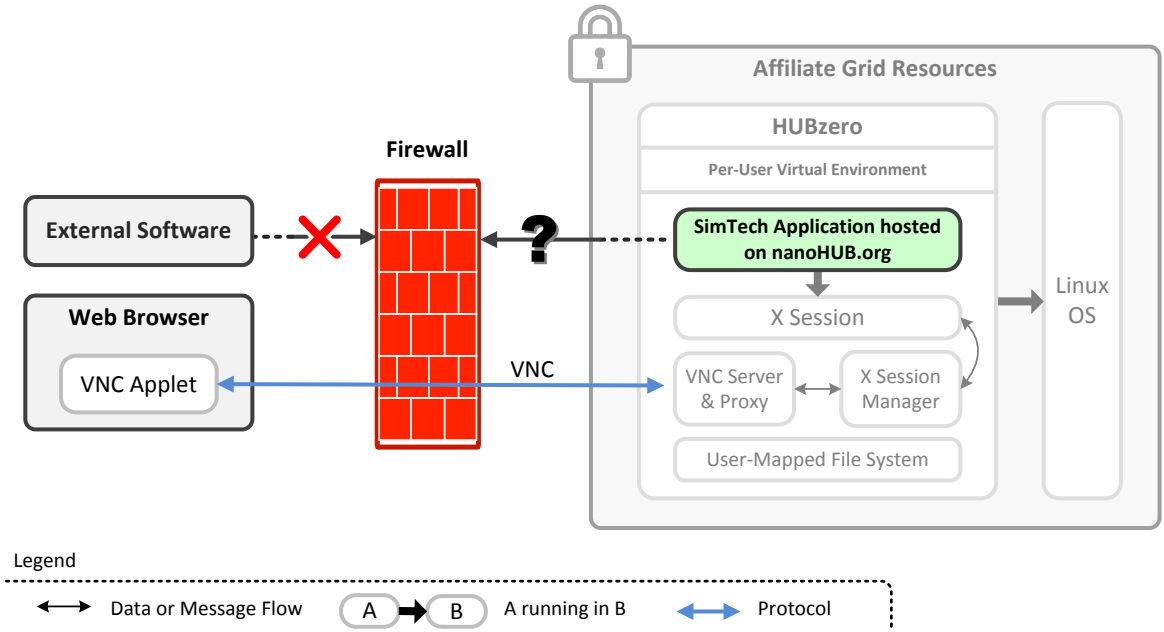


Abbildung 15. Integrationsansatz: Ausführung von SimTech-Anwendungen auf nanoHUB.org

Verwendung der interaktiven Simulationstools in Simulation Workflows

nanoHUB bietet etwa 260 Simulationstools für das Fachgebiet Nanotechnologie an. Andere HUBzero-basierte Plattformen verwenden denselben Mechanismus, um Simulationstools für weitere Fachgebiete zur Verfügung zu stellen. Es ist denkbar, dass einige dieser bestehenden Simulationstools nützlich genug sind, um sie als Komponenten in Simulation Workflows einzusetzen. Die einzige Schnittstelle zu den Simulationstools ist VNC, daher muss jede technische Realisierung für den Zugriff auf die Simulationstools einen VNC-Client verwenden. Natürlich soll der Aufruf eines Simulationstools aus einem Simulation Workflow keine Interaktion des Wissenschaftlers mit dem Tool erfordern, sondern automatisiert ablaufen. Um die Eingaben an das Tool abzusetzen, müssen also Tastatur- und Mausereignisse simuliert werden, die Ausgabe muss per Texterkennung eingelesen werden, oder, sofern das Tool es ermöglicht, als Dateidownload gespeichert werden.

Ein Konzept für die technische Umsetzung wird in Abbildung 16 dargestellt: Simulation Workflows (1) rufen in diesem Ansatz per BPEL-Invoke (2) einen Web service, der für die Ausführung von Simulationstools beliebiger HUBzero-basierter Plattformen zuständig ist. Dieser neue Web service besteht im Wesentlichen aus zwei Komponenten (3): Eine VNC-Automatisierungskomponente muss mithilfe einer Tool-spezifischen Beschreibungsdatei in der Lage sein, für ein bestimmtes Simulationstool die nötigen Eingaben zu machen und die Ausgabe des Simulationstools abzugreifen. Die Beschreibungsdatei muss die visuellen Positionen aller Eingabefelder enthalten, ggf. in Abhängigkeit der Fensterproportionen. Um einen gültigen Kontext für die VNC-Sitzung zu schaffen, wird eine zweite Komponente benötigt, die sich per HTTP-Client auf der HUB-Website einloggt. Zusätzlich zu diesen beiden Mechanismen muss evtl. ein Load Balancer oder eine Warteschlange zum Einsatz kommen, da pro eingeloggtem Benutzer nur eine begrenzte Zahl von VNC-Sitzungen gleichzeitig aktiv sein darf.

Nutzung der Datei-Ressourcen

Jeder nanoHUB-Benutzer erhält 1 GiB Speicher für die Veröffentlichung von Dateien wie Simulationsergebnissen oder Lehrmaterial. Für die Tool-Entwicklung stehen außerdem Subversion-Repositories zur Verfügung, in denen sich Dokumente versioniert einpflegen lassen. Prinzipiell lässt sich dieses Angebot nutzen, um in Simulation Workflows bestehende Datensätze der Plattform abzurufen oder z.B. Simulationsergebnisse auf der Plattform zu speichern und mit anderen Benutzern zu teilen.

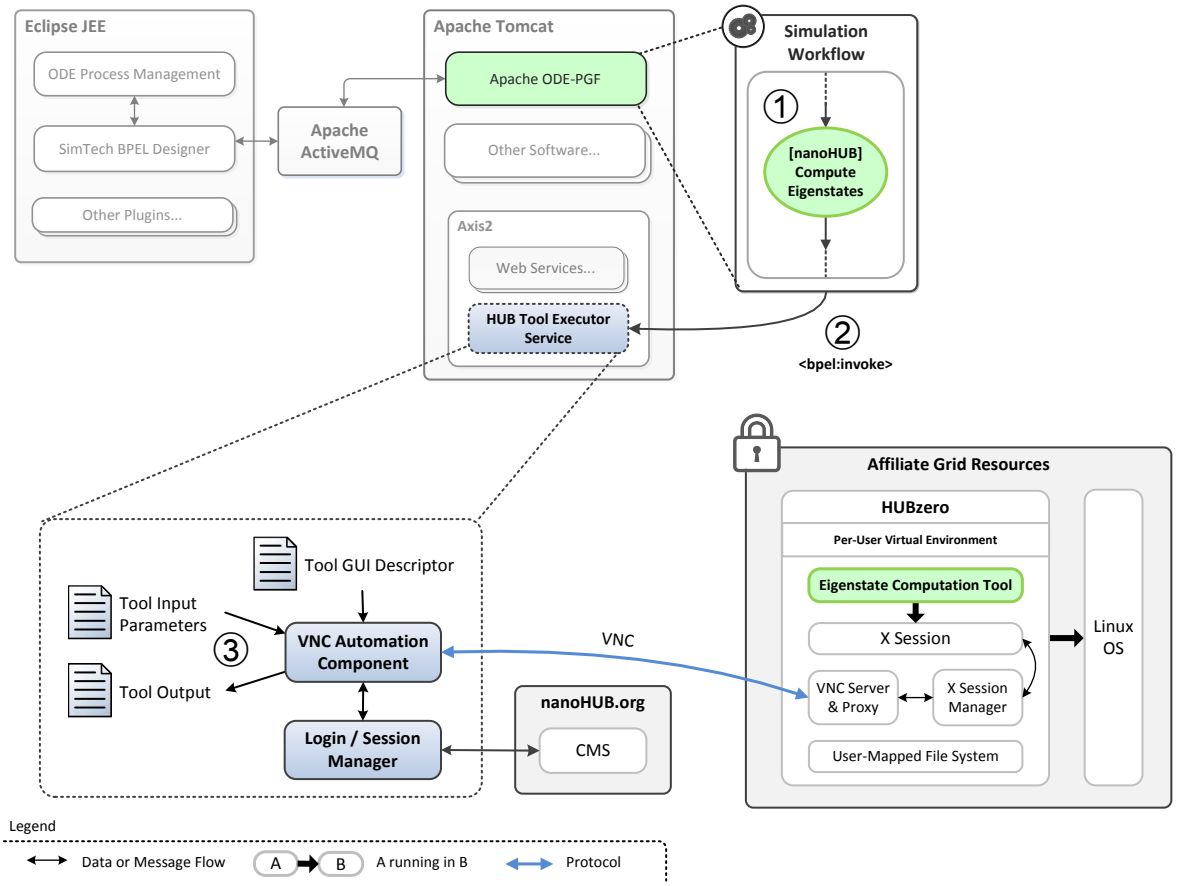


Abbildung 16. Integrationsansatz: Verwendung der interaktiven Simulationstools in Simulation Workflows

Für den lesenden und schreiben Zugriff auf Dateien bietet nanoHUB zwar WebDAV[17] und SFTP[18] als zusätzliche Schnittstellen an, im Versuch waren diese aber nicht erreichbar. Eine Integrationslösung müsste daher wie in Abbildung 17 dargestellt per HTTP die Anmeldung auf der Website automatisieren und den Dateitransfer ebenfalls per HTTP vornehmen. Problematisch für diesen Integrationsansatz ist jedoch, dass jeder Upload eines regulären nanoHUB-Benutzers erst von einem nanoHUB-Mitarbeiter manuell freigegeben werden muss.

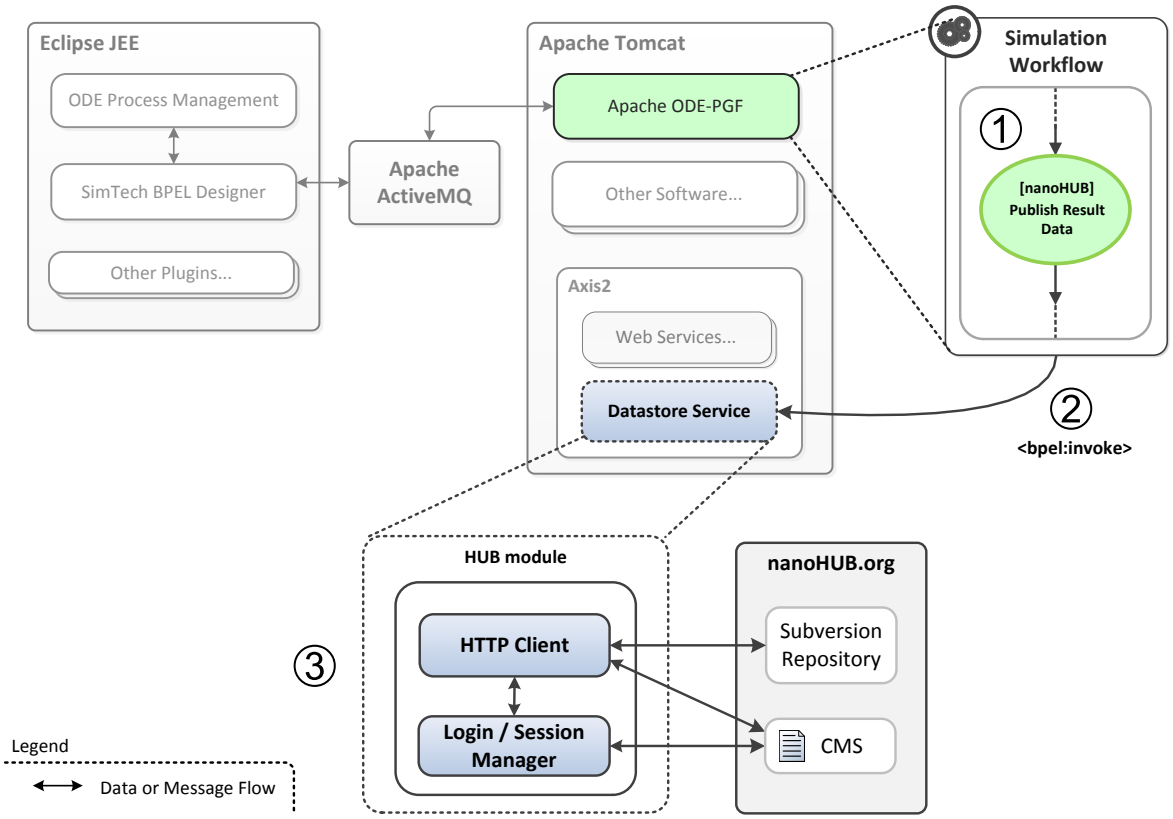


Abbildung 17. Integrationsansatz: Nutzung der Datei-Ressourcen

3.2 myExperiment

myExperiment ist eine Web-Plattform zur gemeinsamen Nutzung und Verteilung von wissenschaftlichen Workflows und der dazugehörigen Ressourcen. Die Plattform ist ein Projekt der eScience-Organisation myGrid[19] und wird hauptsächlich von der University of Southampton und der University of Manchester entwickelt und betrieben. Haupt-Geldgeber ist der Engineering and Physical Sciences Research Council (EPSRC). Seit 2001 wurden die myGrid-Projekte mit etwa 6,5 Mio. GBP gefördert, im Januar 2014 läuft der aktuelle Zuschuss aus[20]. Das Hauptziel des Projekts ist es, den Benutzern zu ermöglichen, ihre Workflows aktiv mit anderen Wissenschaftlern zu teilen. myExperiment beschreibt

[17] Web Distributed Authoring and Versioning: Erweiterung von HTTP für Dateizugriffe.
[18] Secure File Transfer Protocol: Protokoll zur verschlüsselten Übertragung von Dateien über ein Netzwerk.
[19] `http://www.mygrid.org.uk/`
[20] Zugehörige Förderungen des EPSRC:

- `http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/G026238/1`; `http://archive.is/PxH8G`
- `http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/D044324/1`; `http://archive.is/3HLqp`
- `http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=GR/R67743/01`; `http://archive.is/iUtIP`

sich selbst als virtuelle Forschungsumgebung, konkret handelt es sich um ein Repository für wissenschaftliche Workflows und sonstige Dateien. Man findet außerdem grundlegende Social Networking Funktionen wie Benutzerprofile, Kontaktlisten, Gruppen und Nachrichtenversand. Des Weiteren bietet myExperiment eine Sicht auf die Daten des Service-Katalogs BioCatalogue [21], in dem Web services zu verschiedenen Themenbereichen der Biowissenschaften registriert sind. [5]

Ein herausragendes Merkmal der Plattform ist die automatische Erkennung und Aufbereitung von Metadaten beim Upload von Workflows. Für eine Reihe von Workflow-Sprachen generiert die Website automatisch Vorschau-Diagramme und Statistiken, z.B. über die verwendeten Workflow-Aktivitäten. Unter den unterstützten Workflow-Sprachen befinden sich auch Taverna 1 und Taverna 2 Workflows, BPEL allerdings nicht[22].

Die Funktionen der Website stehen zum großen Teil auch über eine HTTP-Schnittstelle zur Verfügung. Der Lesezugriff auf Ressourcen der Plattform ist vollständig implementiert. Eine detaillierte und aktuelle Auflistung der implementierten Funktionen befindet sich im myExperiment Entwickler-Wiki[23]. Die Ruby on Rails Webanwendung hinter myExperiment ist freie Software unter der BSD-Lizenz und kann prinzipiell von jedermann in Betrieb genommen werden. Im Rahmen dieser Arbeit wurde außer `http://www.myexperiment.org` allerdings kein weiterer ernsthafter Betreiber entdeckt.

Integration mit myExperiment

Die Möglichkeiten zur Integration mit myExperiment sind etwas eingeschränkt, da myExperiment Workflow-Modelle lediglich speichert, versioniert und indexiert, aber keine Ausführung der hochgeladenen Workflows ermöglicht. Das Workflow-Repository könnte dennoch in die SimTech-Modellierungsumgebung integriert werden, auch wenn keine Metadaten-Erkennung für BPEL-Workflows zur Verfügung steht. Des Weiteren sind die Nutzung der Datei-Ressourcen und des Service-Katalogs denkbare Integrationsansätze.

Einbindung des Workflow-Repositories

Wird das Workflow Repository in das SimTech Modeling Tool integriert, sind mehrere Anwendungsfälle denkbar. Zum einen kann der Benutzer lokal gespeicherte Workflows mit dem myExperiment Repository synchronisieren und versionieren. Zum anderen lassen sich Taverna Workflows anderer myExperiment-Benutzer auffinden und als Sub-Prozess aufrufen, falls Taverna Workflows ebenfalls integriert werden. Diese Funktionen müssten mit einem neuen Eclipse-Plugin realisiert werden, welches mit der myExperiment HTTP- API oder mit der Web-Oberfläche interagiert.

Nutzung der Datei-Ressourcen

Wie auch bei nanoHUB (siehe Abschnitt 3.1) steht es jedem Benutzer der Plattform frei, beliebige Dateien hochzuladen. Im Unterschied zu nanoHUB findet auf myExperiment jedoch keine manuelle Freigabe der hochgeladenen Daten statt und die technische Realisierung wird durch die vorhandene HTTP-API etwas vereinfacht. myExperiment behält sich vor, Kopien aller hochgeladenen Daten zu erstellen.

Nutzung des Service-Katalogs

Der Service-Katalog BioCatalogue kann in das SimTech Modeling Tool integriert werden, um dem Modellierer wissenschaftlicher Workflows eine durchsuchbare Auswahl von Web services anzubieten. Nützliche Dienste können dann z.B. per Drag&Drop direkt in ein Workflow-Modell eingefügt werden. Die entsprechenden BPEL-Fragmente, z.B. `invoke`-Aktivitäten, werden automatisch erzeugt. Um dies zu realisieren, müssten ein oder mehrere Eclipse-Plugins entwickelt werden, die Suchanfragen des

[21] `http://biocatalogue.org`
[22] Liste der unterstützen Typen: `http://www.myexperiment.org/content_types`; `http://archive.is/Qjpx7`
[23] `http://wiki.myexperiment.org/index.php/Developer:API`; `http://archive.is/a19Ym`

Modellierers an den Service-Katalog weiterleiten, das Suchergebnis auf geeignete Weise präsentieren und bei Bedarf BPEL-Fragmente für den Aufruf der Web services erzeugen.

3.3 Taverna

Taverna ist ein Workflow-Management-System mit einer proprietären Workflow-Beschreibungssprache [26]. Wie myExperiment ist auch Taverna ein Projekt der eScience-Organisation myGrid. Das Projekt hat zwar enge Verbindungen zu den Biowissenschaften, dennoch sind Taverna-Workflows universell einsetzbar und nicht an eine bestimmte Domäne gebunden [20]. In dieser Arbeit betrachten wir das Release 2.4, was in der ersten Hälfte 2013 der aktuelle Stand war. Taverna ist vollständig in Java geschrieben und ist freie Software unter der GNU Lesser General Public License 2.1.

Abbildung 18 zeigt die Hauptbestandteile der Taverna-Softwaresammlung. Sie besteht aus einem Modeling Tool, in dem auch Workflows ausgeführt werden können und aus zwei separaten Workflow-Ausführungsumgebungen. In den folgenden Abschnitten werden das zugrundeliegende Workflow Metamodell und die einzelnen Softwarekomponenten kurz beschrieben.

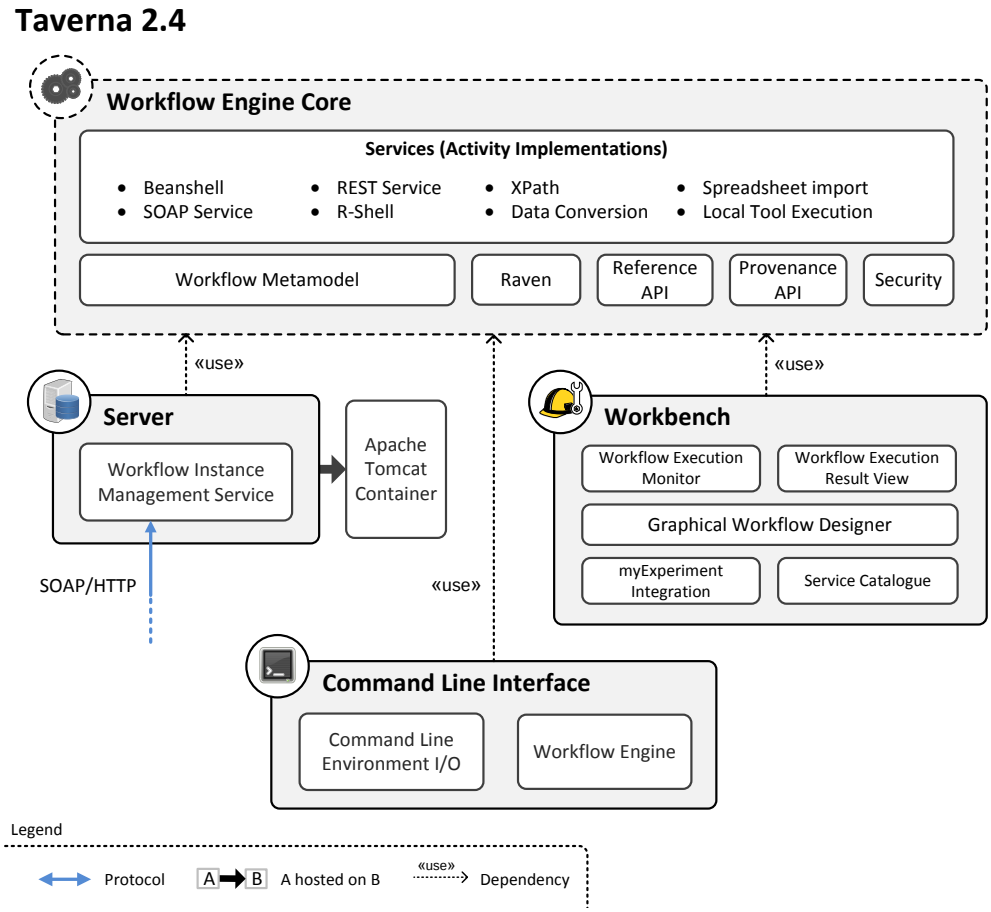


Abbildung 18. Hauptbestandteile des Taverna 2.4 WfMS

Ein Taverna 2 Workflow ist ein gerichteter, azyklischer Graph, dessen Knoten Verarbeitungseinheiten (processors) sind. Jede Verarbeitungseinheit wandelt eine Menge von Eingaben in eine Menge von Ausgaben um. Es gibt zwei Kantentypen: einen für den Datenfluss zwischen den Verarbeitungseinheiten und einen weiteren für den Kontrollfluss. Den Kontrollfluss kann, muss man aber nicht modellieren, weil ein impliziter Kontrollfluss aus dem Datenfluss hervorgeht. Die Orientierung am Datenfluss ermöglicht einen relativ effizienten Umgang mit Datenströmen (siehe Abschnitt Services) sowie die automatische Parallelisierung des Kontrollflusses. Abbildung 19 zeigt die graphische Darstellung eines einfachen

Taverna-Workflows, bei dem der Kontrollfluss durch eine explizite Kontrollfluss-Kante eingeschränkt wird. Die Kante stellt eine Kausalordnung zwischen den verbundenen Verarbeitungseinheiten her.

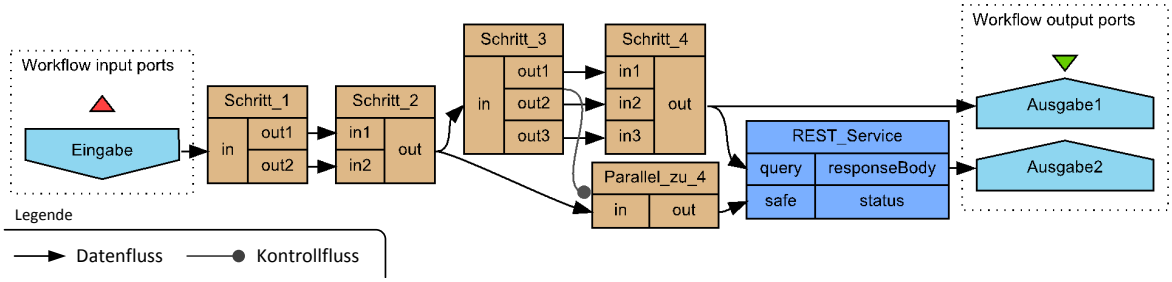


Abbildung 19. Einfacher Taverna-Workflow mit einer expliziten Kontrollfluss-Kante

Das Taverna Workflow Metamodell kennt drei Datenstrukturen: Skalare vom Typ `String` oder `byte[]`, Fehlerdokumente und Listen. Listen können alle Datentypen enthalten und bis zu einer Tiefe von 100 geschachtelt werden. Im Gegensatz zu BPEL 2.0 bietet das Taverna Metamodell keine blockbasierten Programmierstrukturen. For-Schleifen müssen nicht explizit modelliert werden, sondern können emuliert werden, indem ein ausgehender Listen-Datenfluss an eine Verarbeitungseinheit angelegt wird, die als Eingabe Skalare erwartet. Repeat-Until-Schleifen lassen sich über einen gesonderten Mechanismus implementieren, bei dem ein verschachtelter Workflow so lange wiederholt wird, bis eine bestimmte Bedingung eintritt.

Workflow Engine

Der Kern des Taverna WfMS umfasst knapp 40 Maven-Projekte. Die Artefakte, welche aus diesen Projekten hervorgehen, ergeben aber noch keine lauffähige Software. Es handelt sich um Libraries, auf denen eine konkrete Ausführungsumgebung wie der Taverna Server aufbauen kann. Aus diesem Grund ist die Workflow Engine in Abbildung 18 zur Unterscheidung von lauffähigen Softwareprodukten mit einer gestrichelten Linie dargestellt. Zum Kern gehört auch das XML-basierte Exportformat für Taverna 2 Workflows, welches die Dateierweiterung `t2flow` trägt. Version 1 der Taverna Engine verwendet mit SCUFL ein anderes Workflow Metamodell und Exportformat. Im Folgenden ist mit der Bezeichnung „Taverna Workflow“ immer `t2flow` gemeint.

Services Workflow-Aktivitäten heißen im Taverna-Vokabular Services. Ein Service ist eine Verarbeitungseinheit und hat somit $0..n$ Eingaben und $0..m$ Ausgaben. Verschachtelte Workflows sowie Aktionen zur Konvertierung, Aufteilung oder Zusammenführung von Datenstreams sind ebenfalls Verarbeitungseinheiten, aber keine Services. Taverna ermöglicht Streaming der Ein- und Ausgaben zwischen Verarbeitungseinheiten. Ein Service, welcher von der Ausgabe eines anderen Services abhängt, muss daher nicht warten, bis die vorherige Berechnung komplett abgeschlossen ist, sondern kann einen eingehenden Datenstrom stückchenweise verarbeiten, wie ihn der vorgeschaltete Service produziert. So entstehen innerhalb eines Taverna Workflows automatisch Daten-Pipelines. Allerdings unterstützen nicht alle Service-Typen Streaming.

Die folgenden Service-Implementierungen stehen jedem Taverna Workflow zur Verfügung:

- Ausführung eines Beanshell-Skripts. Beanshell ist eine Skriptsprache mit vereinfachter Java-Syntax. Der Skript-Code wird in das Workflow Modell eingebettet. Pipelining der Ein- und Ausgaben von Beanshell-Skripten ist in Version 2.4 nicht möglich[24].
- Web service Aufruf per SOAP/HTTP anhand einer WSDL-Datei
- Aufruf von HTTP-Schnittstellen

[24] Offene Aufgabe für Taverna 3: `http://dev.mygrid.org.uk/issues/browse/T3-87`; `http://archive.is/VcU9Q`

- Statistische Berechnungen und Grafiken per R-Shell[25]
- Ausführung von XPath-Queries
- Ausführung lokaler Programme per Kommandozeile
- Import von Kalkulationstabellen

Weitere Service-Implementierungen lassen sich auf drei Arten realisieren: In der einfachsten Variante kann über den Beanshell-Service Java-Code aus einer JAR-Datei ausgeführt werden. Dazu muss die JAR-Datei zusammen mit dem Workflow ausgeliefert werden. Eine weitere Möglichkeit ist die Verwendung des API Consumer Tools [26], mit dem sich Teile einer Java-API als Service abbilden lassen. Das Tool erstellt eine XML-Datei, welche in der Taverna Workbench importiert werden kann. Auch bei dieser Variante müssen die gerufenen Java-Klassen und deren Abhängigkeiten für die Ziel-Ausführungsumgebung verfügbar gemacht werden. Die dritte und aufwändigste Möglichkeit ist die Erstellung eines Taverna-Plugins, das einen neuen Service-Typ implementiert.

Reference & Provenance APIs Die Workflow-Engine behandelt in ihrer Java-API alle Datensätze zunächst als Referenzen. Um an die dahinterliegenden Datenstreams zu gelangen, muss eine Referenz über die Reference API aufgelöst werden. Der konkrete Speicherort der Daten ist implementierungsabhängig: die Daten können lokal im Arbeitsspeicher oder in einer entfernten Datenbank abgelegt sein. Für Daten, die von einem Taverna Workflow produziert werden, kann die Engine über die Provenance API automatisch Herkunftsnachweise erzeugen und diese separat in einer Datenbank speichern.

Raven Fest in die Workflow Engine integriert ist ein eigener Java Classloading-Mechanismus, der sich an Maven-Artefakten orientiert. Raven setzt den Java Classpath-Mechanismus teilweise außer Kraft und lädt Klassen stattdessen aus einem Maven-Repository. Für jedes registrierte Maven-Artefakt wird ein eigener Classloader instantiiert, welcher alle Klassen aus der transitiven Hülle der Maven-Abhängigkeiten des Artefakts laden kann. Somit wird es möglich, verschiedene Versionen von Softwarebibliotheken gleichzeitig zu nutzen.

Server

Der Taverna Server ist eine Java-Webanwendung zur Ausführung von Taverna Workflows, auf die entfernt über eine HTTP- oder eine SOAP Web service Schnittstelle zugegriffen werden kann. Über diese Schnittstelle werden Workflow-Instanzen sowie deren Ein- und Ausgabedaten verwaltet, und die Konfiguration des Servers geändert. Anders als bei der Apache ODE wird beim Taverna Server keine Menge von installierten Workflow-Modellen verwaltet. Der Taverna Server nimmt bei jedem Start einer neuen Workflow-Instanz das zugehörige Workflow-Modell als Parameter der `submitWorkflow` Operation entgegen und ordnet das Modell ausschließlich der gestarteten Instanz zu. Jede gestartete Workflow-Instanz erhält eine eindeutige Kennung, über die sich nach dem Start u.a. das dazugehörige Workflow-Modell und die erzeugten Ausgaben abrufen lassen. Der Server führt jede Workflow-Instanz in einer eigenen Java Virtual Machine (JVM) aus, indem er einen neuen Betriebssystem-Prozess abzweigt.

Command Line Interface

Neben dem Server existiert mit der Kommandozeilen-Schnittstelle eine einfachere Möglichkeit, Taverna Workflows auszuführen. Auch hier entsteht je Workflow-Instanz ein neuer Betriebssystem-Prozess. Die Ein- und Ausgabedaten werden als Dateien übergeben, wobei man sehr kleine Eingaben auch direkt als Kommandozeilenargument übergeben kann.

[25] Softwareumgebung für statistische Berechnungen: `http://www.r-project.org/`
[26] `http://www.taverna.org.uk/documentation/taverna-2-x/api-consumer-tool/`

Workbench

Die Taverna Workbench besteht im Wesentlichen aus einem graphischen Workflow Designer, einer Ansicht zur Ausführung von Workflows und zur Auswertung der Ergebnisse, Integration mit myExperiment und dem Servicekatalog BioCatalogue.org.

Der Servicekatalog und die myExperiment-Integration unterstützen den Anwender bei der Auswahl geeigneter Workflow-Aktivitäten und enthalten hauptsächlich Web services für Aufgaben aus den Biowissenschaften. Die gefundenen Services lassen sich direkt als neue Aktivität in den aktuell bearbeiteten Workflow einfügen. Workflows werden im Workflow Designer per Drag&Drop modelliert, das Layout des Datenfluss-Graphen (siehe Abbildung 20) wird dabei stets automatisch berechnet und lässt sich nicht vom Benutzer ändern.

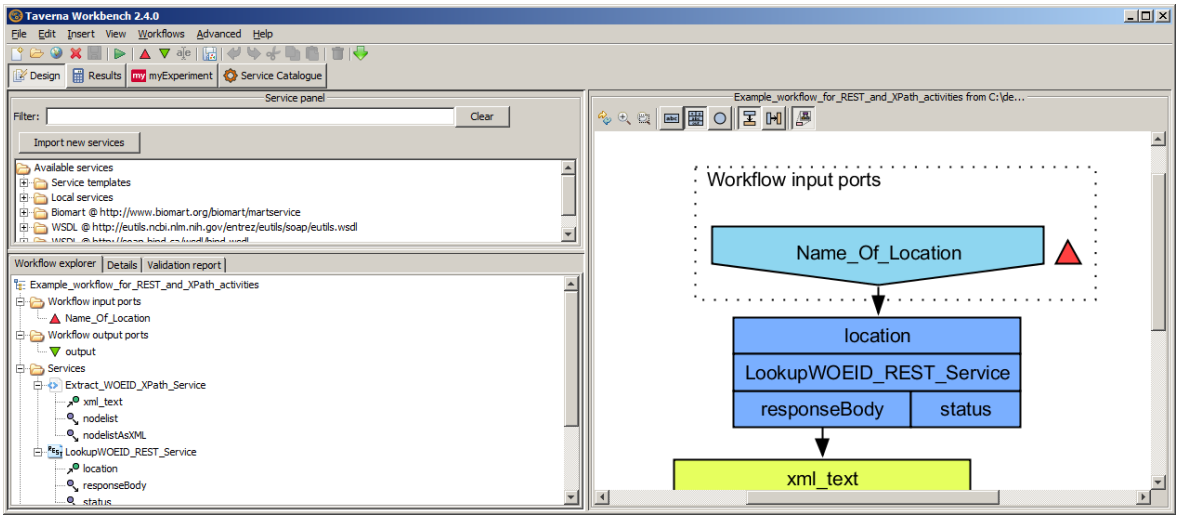


Abbildung 20. Taverna Workflow Designer

Wird ein Workflow in der Workbench ausgeführt, erhält man eine graphische Darstellung des Ausführungsfortschritts. Durch Färbung des Graphen werden ausgeführte Aktivitäten, noch nicht ausgeführte Aktivitäten und Fehler bei der Ausführung unterschieden. Noch während der Ausführung können Teilausgaben untersucht werden, sobald sie vorliegen. Bisher unterstützt die Workbench nur die Ausführung auf dem lokalen Rechner, es gibt keine Integration mit dem Taverna Server.

Zukünftige Entwicklung

In der zweiten Hälfte des Jahres 2013 soll die erste Alphaversion[27] von Taverna 3.0 erscheinen, in der es zwei signifikante Änderungen geben wird: Zum einen wird SCUFL2[28] `t2flow` als Workflow-Beschreibungssprache ersetzen, zum anderen wird anstelle von Raven in Zukunft das OSGi-Framework als Modulsystem eingesetzt.

Integration von Taverna

Taverna ist im Gegensatz zu den anderen untersuchten Plattformen eine reine Softwareplattform ohne dazugehörige Infrastruktur. Die Integration von Taverna in die SimTech Workflow-Umgebung ist daher eine (nicht weniger spannende) Rückfall-Lösung, falls aus der Untersuchung von nanoHUB und myExperiment keine attraktiveren Integrationsansätze hervorgehen.

[27] Taverna Roadmap: http://www.taverna.org.uk/introduction/roadmap/;http://archive.is/76xdX
[28] `http://dev.mygrid.org.uk/wiki/display/developer/SCUFL2+language`

Taverna Workflows als Bestandteil von BPEL-Prozessen

Bei diesem Integrationsansatz wird ermöglicht, beliebige Taverna-Workflows als Bestandteil in die BPEL-basierten Simulation Workflows einzubinden (siehe Abbildung 21). Es gibt verschiedene Möglichkeiten, diesen Ansatz technisch zu realisieren, z.B. durch Integration des Taverna Servers, durch Generierung von Wrapper Web services oder durch Erweiterung der Apache ODE. Wir werden diese Herangehensweisen in Kapitel 4.2 im Detail betrachten. Alle Lösungen, die Web service einsetzen, müssen lange laufende Taverna-Workflows und somit die asynchrone Ausführung mittels Callback unterstützen. Die Fähigkeit, Taverna-Workflows in Simulation Workflows einzusetzen, ist Voraussetzung für alle weiteren Integrationsansätze, die im Folgenden genannt werden.

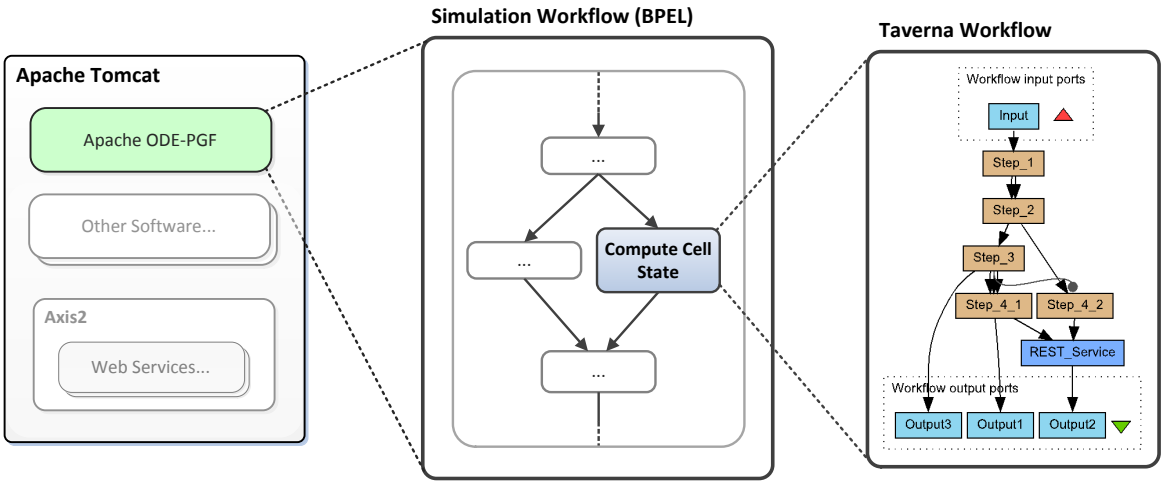


Abbildung 21. Integrationsansatz: Taverna Workflows als als Bestandteil von BPEL-Prozessen

Verwaltung von Taverna Server-Installationen im SimTech Modeling Tool

Falls der Taverna Server eingesetzt wird, um Taverna Workflows als Bestandteil von BPEL-Prozessen auszuführen, kann das SimTech Modeling Tool um einen neuen Adapter für den Taverna Server erweitert werden (siehe Abbildung 22). Mit diesem Server-Adapter können lokale oder entfernte Installationen des Taverna Servers verwaltet werden. Denkbare Anwendungsfälle sind z.B. Überwachung und manuelle Terminierung laufender Workflow-Instanzen oder die Verwaltung der Serverkonfiguration. Die Umsetzung dieses Integrationsansatzes würde analog zur ODE-Integration als Eclipse-Plugin erfolgen.

Eingliederung der Taverna Workbench in den SimTech BPEL Designer

Um in BPEL *eingebettete* Taverna-Workflows direkt aus dem SimTech Modeling Tool heraus zu editieren, können Teile der Taverna Workbench in den BPEL Designer integriert werden (siehe Abbildung 23). Dieser Integrationsansatz lässt sich z.B. realisieren, indem die betroffenen Workbench-Artefakte als Eclipse-Plugins zur Verfügung gestellt werden. Alternativ dazu kann die Workbench auch weiterhin als eigenständige Anwendung ausgeführt werden: Der bestehende Workbench-Quellcode muss dann so erweitert werden, dass Inter-Prozess-Kommunikation mit dem SimTech Modeling Tool möglich wird.

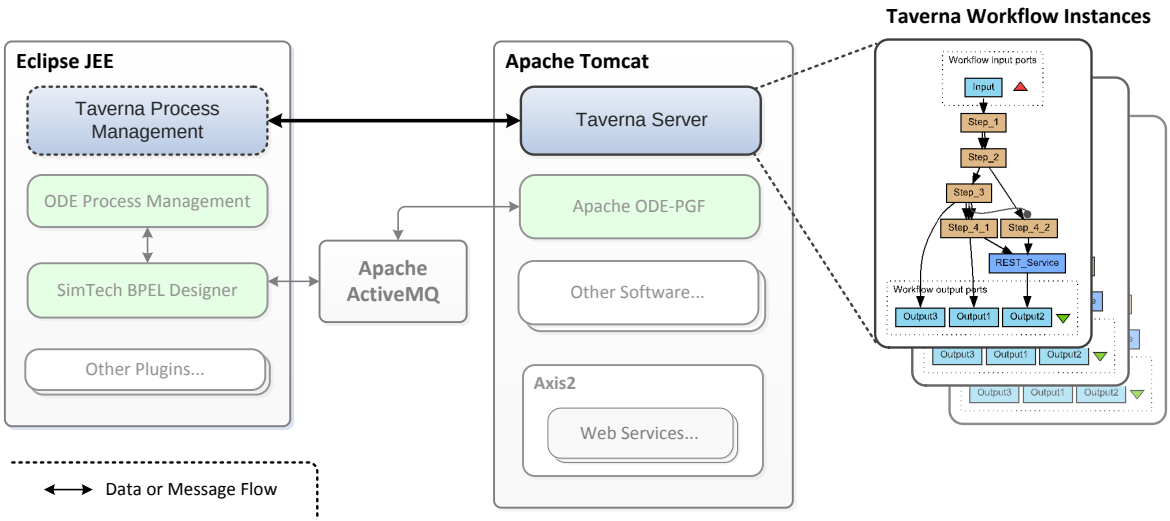


Abbildung 22. Integrationsansatz: Verwaltung von Taverna Server-Installationen aus Eclipse

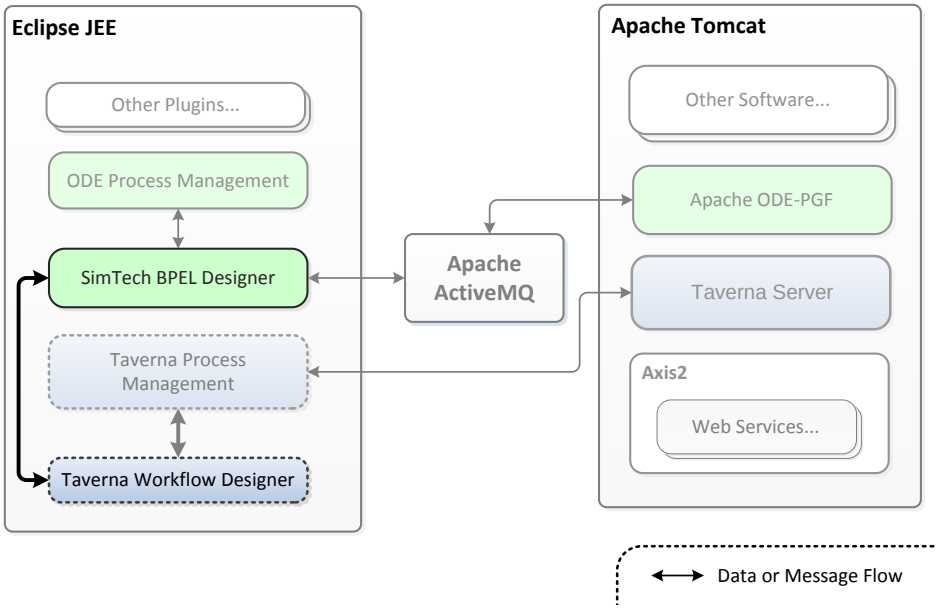


Abbildung 23. Integrationsansatz: Eingliederung der Taverna Workbench in den SimTech BPEL Designer

4 Wahl einer Integrationsaufgabe

Da im zeitlichen Rahmen dieser Diplomarbeit nicht alle Integrationsansätze aus den Kapiteln 3.1, 3.3 und 3.2 verwirklicht werden können und auch nicht jeder denkbare Integrationsansatz unbedingt sinnvoll ist, muss eine Auswahl getroffen werden. Dazu ermitteln wir zunächst die Kriterien, nach denen wir bewerten, wie attraktiv und wie machbar die einzelnen Integrationsansätze sind. Nachdem wir anhand dieser Kriterien eine Auswahl getroffen haben, werden wir in Kapitel 5 die genauen Anforderungen an die Integrationslösung ermitteln, verschiedene Lösungsalternativen vorstellen, und die Implementierung einer Lösung vorstellen.

4.1 Auswahlkriterien

Die Integrationsansätze werden im Folgenden nicht anhand einer Ordinalskala, sondern eher in Form von Gutachten miteinander verglichen. Entsprechend sind die Kriterien auch keine objektiv quantifizierbaren Einheiten, sondern von subjektiver Natur. Sie lassen sich in die Kategorien Nutzen und Machbarkeit aufteilen.

Nutzen

Anhand dieser Kriterien bewerten wir die Nützlichkeit und damit die Attraktivität eines Integrationsansatzes getrennt von der tatsächlichen Machbarkeit der Integration.

- *Nutzung von externer Infrastruktur:* Von besonders großem Interesse sind alle Ansätze, mit denen sich bestehende SimTech-Infrastruktur auf die Rechner der Plattform auslagern lassen.
- *Nähe zum Thema:* Alle Integrationsansätze sollten einen klaren Bezug zur bestehenden Workflow-Umgebung erkennen lassen.
- *Neuartigkeit:* Der ausgewählte Ansatz sollte entweder etwas ermöglichen, was vorher in der SimTech Workflow-Umgebung nicht möglich war oder erlauben, dass etwas radikal anders gemacht wird als zuvor.
- *Ersetzbarkeit:* Ein attraktiver Integrationsansatz nutzt die herausragenden Merkmale der eScience-Plattform und ist nicht leicht durch die Dienste eines spezialisierteren Anbieters zu ersetzen.
- *Zweckentfremdung:* Die integrierte Plattform sollte auf die vom Betreiber vorgesehene Weise benutzt werden und eine Integrationslösung sollte keine technischen Maßnahmen umgehen, die eine Zweckentfremdung verhindern sollen.

Machbarkeit

Anhand dieser Kriterien soll versucht werden, abzuschätzen, ob die Planung und Konstruktion einer Integrationslösung innerhalb des Zeitfensters dieser Diplomarbeit zu machen ist. Zunächst muss die *Komplexität* der Aufgabe eingeschätzt werden. Eine technisch sehr anspruchsvolle Aufgabe bedeutet ein höheres Risiko, dass der Aufwand unterschätzt wird oder unvorhergesehene Probleme bei der Konstruktion auftreten. Potenzielle technische Probleme sollen bereits im Vorfeld erkannt werden und die Wahl der Aufgabe sollte eher risikoavers gesteuert sein, da es im Zeitplan dieser Arbeit wenig Spielraum für Änderungen gibt. Der geschätzte Zeitaufwand sollte im Bereich von 160 Stunden liegen, damit die Aufgabe rechtzeitig abgeschlossen werden kann.

4.2 Analyse und Bewertung

Die Ansätze aus Kapitel 3 werden an dieser Stelle auf Eignung und Machbarkeit untersucht.

Ausführung von SimTech-Anwendungen auf nanoHUB.org

Auf den ersten Blick scheint dieser Ansatz vielversprechend, aber in der Untersuchung hat sich gezeigt, dass die gewünschte Ausführung von Serverdiensten sich mit nanoHUB nicht umsetzen lässt. Der manuelle Freischaltprozess für neue Simulationstools ist zwar problematisch, aber kein KO-Kriterium. Entscheidend sind vielmehr die Kopplung der Ausführung von Simulationstools an eine VNC-Sitzung sowie die Erwartung der Plattform-Betreiber, dass interaktive Software aufgespielt wird. Einzig das SimTech Modeling Tool könnte auf nanoHUB ausgeführt werden, wenn auch die Bedienung einer Eclipse-Anwendung im Browserfenster etwas unangenehm ist. Auf die Interaktion mit einer ODE-Instanz muss aufgrund der isolierten Ausführungsumgebung aller Simulationstools verzichtet werden, ebenso auf eingehende und möglicherweise auf ausgehende Netzwerkverbindungen. Es ist zudem sehr fraglich, ob die Installation einer kompletten Eclipse-Plattform genehmigt werden würde, weil sich deren Ressourcenverbrauchsprofil von den bestehenden Simulationstools sehr unterscheidet. Ob für Simulationstools überhaupt eine JVM zur Verfügung steht, ist unklar. Im Versuch wurde die Installation einer einfachen Java-Anwendung mit Swing-GUI verweigert. Aus den genannten Gründen erscheint deshalb auch die Auslagerung des Modeling Tools unattraktiv.

Verwendung der interaktiven Simulationstools in Simulation Workflows

Die Simulationstools sind zwar das herausragende Merkmal der Plattform, jedoch besteht an der Verwendung der bestehenden Tools nur geringes Interesse. Viele der angebotenen Tools sind lediglich GUI-Wrapper um frei verfügbare Kommandozeilenprogramme. Solche Software sollte besser über WSI[22] anstatt über VNC eingebunden werden. Die Ansteuerung der Simulationstools über VNC ist zudem sehr aufwändig, technisch nicht elegant und unzuverlässig: Kleinste Änderungen der Tool-Oberfläche können die VNC-Automatisierung zum Erliegen bringen. Im gegebenen Zeitrahmen wäre die Konstruktion vermutlich nicht machbar, da der geschätzte Aufwand bei mehr als 320 Stunden liegt.

Nutzung der Datei-Ressourcen

Dieser Ansatz scheitert an den Kriterien Neuartigkeit und Ersetzbarkeit: Für Datei-Hosting sind andere Dienste besser geeignet, auch wenn dadurch Kosten verursacht werden. Die manuelle Freigabe aller Uploads auf nanoHUB macht den Ansatz ebenfalls unattraktiv. Obwohl die technische Umsetzung mangels einer geeigneten Schnittstelle unangemessen aufwändig wäre, läge sie mit weniger als 160 Stunden dennoch innerhalb des vorgegebenen Zeitrahmens.

Einbindung des Workflow-Repositories

Dieser Ansatz ist uninteressant, solange die SimTech Workflow-Umgebung keine Taverna-Workflows unterstützt, da im Workflow-Repository von myExperiment keine BPEL-Prozesse geführt werden. Dieser Ansatz gehört mit weniger als 160 Stunden zu den einfachen Aufgaben.

Nutzung der Datei-Ressourcen

Die Nutzung der Datei-Ressourcen auf myExperiment unterscheidet sich nur in Details von nanoHUB und ist daher ebenfalls kein attraktiver Integrationsansatz.

Nutzung des Service-Katalogs

Ein Web-basierter Servicekatalog wie BioCatalogue ist bisher nicht in das SimTech Modeling Tool integriert. Die Vorteile gegenüber der direkten Benutzung der Servicekatalog-Website sind jedoch überschaubar, sodass hier wenig Neuartiges entsteht. Der Servicekatalog und insbesondere das Angebot von BioCatalogue haben sich im Gespräch mit den Betreuern als wenig interessant für das

Institut herausgestellt. Da eine ansprechende Präsentation der Suchergebnisse im Modeling Tool sehr aufwändig werden kann, liegt der geschätzte Zeitaufwand bei etwa 160 Stunden.

Taverna Workflows als Bestandteil von BPEL-Prozessen

Die Orientierung am Datenfluss in Taverna-Workflows ist ein komplementärer Ansatz zu BPEL, den es so in der existierenden SimTech Workflow-Umgebung noch nicht gibt. Mit diesem Integrationsansatz lassen sich die Stärken von Taverna-Workflows in BPEL-Prozessen nutzen. Gleichzeitig ist die Ausführung von Taverna-Workflows der Ausgangspunkt für weitere Taverna-bezogene Integrationsansätze. Der Aufwand zur Realisierung hängt von der Komplexität des gewählten Entwurfsansatzes ab. Unter den diskutierten Ansätzen sind auch solche mit etwa 160 Stunden Aufwand.

Verwaltung von Taverna Server-Installationen im SimTech Modeling Tool

Die Integration des Taverna Servers in das Modeling Tool ist für den Betrieb von Taverna-Workflows nicht essenziell, sondern bietet nur Bequemlichkeitsfunktionen. Da im Gegensatz zur ODE-Integration kein automatisches Deployment von Taverna-Workflow-Modellen benötigt wird, ist der Nutzen dieser Integration sehr fraglich. Der Konstruktionsaufwand liegt geschätzt bei weniger als 160 Stunden.

Eingliederung der Taverna Workbench in den SimTech BPEL Designer

Falls Taverna-Workflows in BPEL-Dateien *eingebettet* werden, ist dieser Integrationsansatz eine große Erleichterung, um die eingebetteten Taverna-Workflows zu bearbeiten. Dennoch bietet die Integration der Workbench keine grundlegend neue Funktionalität. Falls Taverna-Workflows per Dateireferenz in BPEL-Prozesse eingebunden werden, wäre sie überflüssig. Die Komplexität dieser Integrationsaufgabe hängt stark von der Herangehensweise ab: Die Bereitstellung der Workbench als Eclipse-Plugins ist im Vergleich sehr viel aufwändiger als die Modifikation der alleinstehenden Workbench-Anwendung. Letzteres liegt bei einem geschätzten Aufwand von unter 160 Stunden.

4.3 Entscheidung für Taverna Workflows als Bestandteil von BPEL-Prozessen

Die endgültige Wahl der durchzuführenden Integrationsaufgabe erfolgte in Absprache mit den Betreuern. Da die Untersuchung ergeben hat, dass auf nanoHUB keine Dienste ausgelagert werden können, fiel die Wahl auf Taverna. In dieser Arbeit soll die Taverna Workflow-Engine integriert werden, da die Ausführung von Taverna-Workflows als Bestandteil von BPEL-Prozessen Voraussetzung für weitere Integrationsschritte ist.

5 Integration der Taverna Workflow-Engine

In diesem Kapitel werden wir die konkreten Anforderungen an die Integrationslösung sammeln und davon ausgehend verschiedene Entwurfsansätze vorstellen. Die Entwurfsansätze werden miteinander hinsichtlich der Anforderungen verglichen und ein geeigneter Ansatz wird schließlich zur Implementierung ausgewählt.

5.1 Anforderungen

Wir unterscheiden zwischen Basisanforderungen, die auf jeden Fall von allen Ansätzen erfüllt sein müssen, und weiteren Anforderungen, deren Erfüllungsgrad mit einer dreiwertigen Ordinalskala abgebildet wird: - (schlecht), o (ohne Wertung), + (gut).

Basisanforderungen

Dies sind die zu implementierenden, elementaren Anwendungsfälle. Taverna-Workflows müssen als Bestandteil eines BPEL-Prozesses synchron als auch asynchron ausführbar sein. Per BPEL-Code müssen laufende Taverna-Workflow-Instanzen abgebrochen werden können. Es muss ein Konzept zur Fehlerbehandlung vorhanden sein, da Taverna-Workflows Fehler erzeugen können und diese in Form von Fehlerdokumenten als Teil des Ausführungsergebnisses zurückliefern. Die Workflow-Eingaben müssen vom Modellierer per BPEL-Code gesetzt und verändert werden können. Die Workflow-Ausgabe muss vom umgebenden BPEL-Prozess ausgelesen und ausgewertet werden können. Dabei muss das Taverna-Typsystem vollständig unterstützt werden, d.h. es muss zwischen Zeichenketten, Binärdaten und (verschachtelten) Listen unterschieden werden.

Weitere Anforderungen

- Standardkonformität: Werden bestehende Standards im Entwurf und in der Implementierung verwendet?
- Plattformunabhängigkeit: Kann die Lösung in unterschiedlichen Hardware- und Softwareumgebungen ausgeführt werden?
- Erweiterbarkeit/Änderbarkeit: Wie groß ist der Aufwand aus der Sicht eines Entwicklers, um neue Funktionen hinzuzufügen oder bestehende Funktionalität zu ändern?
- Implementierungsaufwand: Kann die Lösung im gegebenen Zeitrahmen konstruiert werden?
- Komplexität des Lösungsansatzes: Ist der Entwurf aus der Sicht eines Entwicklers für das Problem angemessen?
- Bedienbarkeit: Ist die Lösung aus Sicht eines Anwenders einfach zu bedienen?
- Wiederverwendung: Werden bestehende Softwarelösungen wiederverwendet, anstatt das Rad neu zu erfinden?
- Erweiterte Management-Funktionalität: Bietet die Lösung neben den Basisfunktionen zur Ausführung von Taverna-Workflows weitere Funktionalität zur Verwaltung von Workflow-Instanzen oder der Software-Konfiguration?

5.2 Beispiel

Die Integrationsaufgabe soll hier anhand eines Beispiels verdeutlicht werden. In diesem Beispiel wird ein bestehender Taverna-Workflow in einen BPEL-Workflow eingebunden und mehrmals ausgeführt. Der Taverna-Workflow (siehe Abbildung 24) ist in diesem Beispiel ein Platzhalter für eine komplexere Simulation von Zellen in einem Gewebe: Er berechnet für ein gegebenes zweidimensionales Zellgitter

einen Zeitschritt in Conway's Game of Life[29] und liefert das veränderte Zellgitter zurück. Der umgebende BPEL-Prozess hat die Aufgabe, das initiale Zellgitter entgegenzunehmen und den Taverna-Workflow in einer Schleife aufzurufen. Die Rückgabe des Taverna-Workflows wird jeweils zur Eingabe des nächsten Durchlaufs. Nach jedem Durchlauf gibt der BPEL-Prozess das Zwischenergebnis zur Visualisierung an einen speziellen Web service weiter. Dies wird fortgeführt, bis eine bestimmte Anzahl von Iterationen erreicht ist. Abbildung 25 zeigt das Beispiel als Pseudocode.

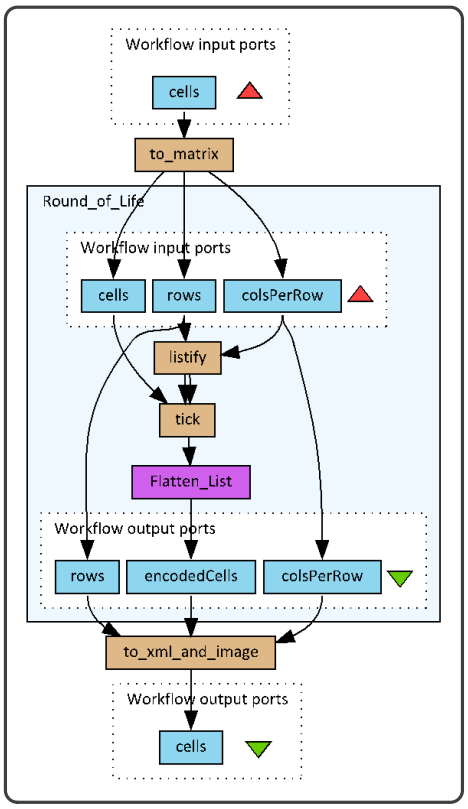


Abbildung 24. Taverna-Workflow für die Berechnung eines Zeitschritts in Conway's Game of Life

```
public void BpelProcess() {

    Taverna2Workflow cell_simulation_tick;
    WebService visualizer;

    Cells cells =
          "<cells colsPerRow='10'>
               <a/><a/><a/><d/><d/><d/><d/><d/><d/>
               <d/><a/><a/><a/><d/><d/><d/><d/><d/>
               <d/><d/><d/><d/><d/><d/><d/><a/><a/>
               <a/><d/><d/><d/><d/><d/><d/><a/><a/>
               <d/><d/><d/><d/><d/><a/><a/><a/><d/>
               <d/><d/><d/><d/><a/><d/><a/><d/><d/>
               <d/><d/><a/><a/><a/><d/><d/><d/><d/>
               <d/><a/><d/><a/><d/><d/><d/><d/><a/>
               <a/><a/><d/><d/><d/><d/><d/><d/><d/>
               <d/><d/><d/><d/><d/><d/><a/><a/><a/>
               <d/><d/><d/><d/><d/><d/><a/><a/><a/>
           </cells>"
    int iterations = 0;
    do {
        WorkflowOutput output = cell_simulation_tick.
              invoke( new WorkflowInput( cells ) );
        cells = output.cells;
        visualizer.invoke( cells, iterations );
    } while ( ++iterations <= 100 );
}
```

Abbildung 25. Pseudocode für den BPEL-Prozess des Beispiels

5.3 Entwurf

Im Folgenden werden verschiedene Entwurfsalternativen vorgestellt, mit denen sich die Basisanforderungen am gezeigten Beispiel erfüllen lassen. Der am besten geeignete Entwurfsansatz wird ausgewählt und implementiert.

Ansatz 1: Alleinstehender Web service je Taverna-Workflow

Bei diesem Ansatz wird je auszuführendem Taverna-Workflow-Modell ein alleinstehendes Web service Artefakt erzeugt, welches ohne weitere Abhängigkeiten in der Lage ist, Instanzen *eines bestimmten* Taverna-Workflow-Modells auszuführen. Der Web service übernimmt die statische Struktur des Taverna-Workflows bzgl. der Ein- und Ausgaben (siehe Abbildung 26). Die Schnittstelle des Web services ist in zwei PortTypes aufgeteilt. Der erste PortType bietet drei Operationen: Die Operation `start` beginnt die asynchrone Ausführung eines Taverna-Workflows mit der gegebenen Eingabe, liefert in der Antwort-Nachricht eine Instanz-ID zurück und sorgt dafür, dass nach dem Ende der Ausführung

[29] Bekanntes, einfaches Regelwerk für einen zweidimensionalen zellulären Automaten

ein Callback-Web-Service aufgerufen wird. Die Instanz-ID ist eine UUID und wird zur Korrelation bei asynchroner Ausführung verwendet. Mit der `abort` Operation wird die Ausführung einer asynchron gestarteten Workflow-Instanz abgebrochen. Die Operation `execute` ist die synchrone Variante, welche in der Antwort-Nachricht bereits das Ergebnis der Ausführung sowie evtl. aufgetretene Fehler enthält. Um das Ergebnis der asynchronen Ausführung zu erhalten, muss von der BPEL-Engine eine Implementierung des zweiten PortTypes bereitgestellt werden. Dieser PortType enthält nur eine Callback-Funktion ohne Rückgabewert, welche die Workflow-Ausgabe entgegennimmt.

Wie in Abbildung 28 dargestellt, wird das Web service Artefakt durch einen Code-Generator erzeugt, der aus einem gegebenen Taverna-Workflow annotierte Java-Klassen generiert und kompiliert. Der Taverna-Workflow wird als Ressource in die JAR-Datei eingebettet. Zusätzlich werden die Kernklassen der Taverna Workflow-Engine eingebettet. Bei der Instantiierung delegiert der Web service die Ausführung dann an die Taverna-Klasse `WorkflowInstanceFacade`. Zur Installation auf Axis2 wird die resultierende JAR"Datei in den Ordner `WEB-INF/servicejars` kopiert.

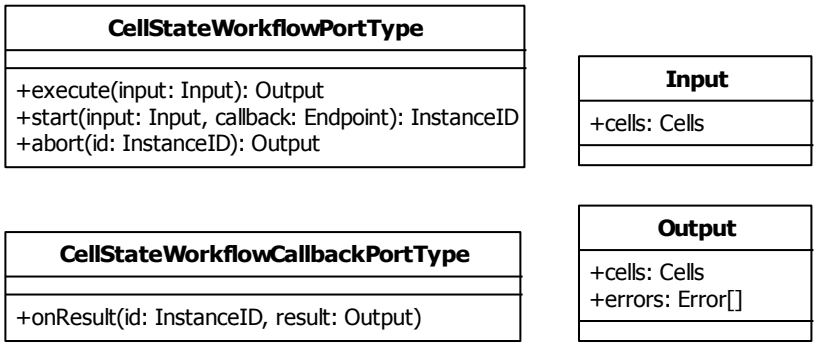


Abbildung 26. Web service Schnittstelle für Beispiel-Workflow

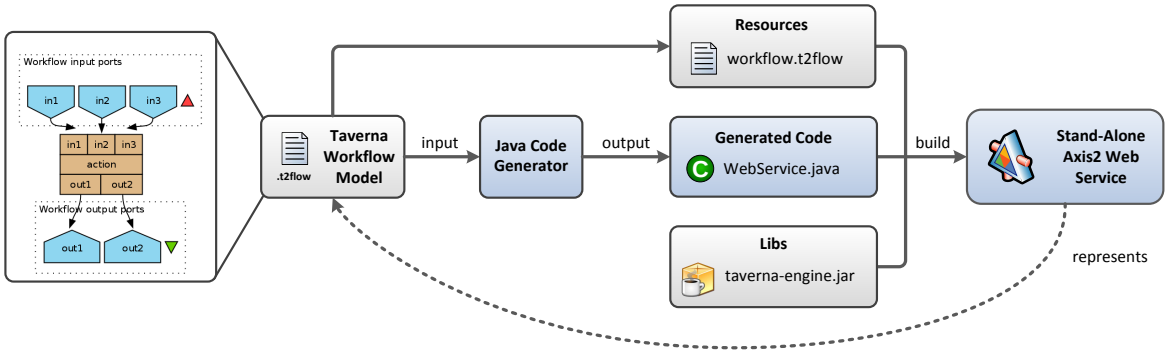


Abbildung 27. Alleinstehender Web service, der ein Taverna-Workflow-Modell repräsentiert

Ansatz 2: Generischer Web service zur Ausführung beliebiger Taverna-Workflows

Bei diesem Ansatz wird eine Alternative zum Taverna Server entwickelt. Dieser alternative Ausführungsdienst besitzt eine vereinfachte Schnittstelle, die sich gut in BPEL-Prozesse integrieren lässt. Im Gegensatz zum Taverna Server muss diese Software nicht unter den Bedingungen der GNU Lesser General Public License verteilt werden. Intern delegiert der Web service die Ausführung an die Taverna Workflow Engine, welche auf geeignete Weise in den Web service eingebettet wird.

Aufgrund der generischen Schnittstelle (siehe Abbildung 29) existiert kein Schema, das für ein bestimmtes Taverna Workflow-Modell verhindert, dass strukturell falsche Eingabeparameter abgesendet

Kriterium	Bewertung	Begründung
Standardkonformität	+	Basiert auf Web service Technologie, verwendet JAX-WS.
Plattformunabhängigkeit	+	In jeder Java Virtual Machine ab Java SE 1.6 lauffähig.
Erweiterbarkeit / Änderbarkeit	-	Änderungen an der Code-Generierung sind aufwändig, bei jeder Änderung müssen alle Web services neu generiert werden.
Implementierungsaufwand	O	Code-Generierung ist vergleichsweise aufwändig.
Komplexität des Lösungsansatzes	-	Code-Generierung ist aufwändig und bringt nur geringe Vorteile im Vergleich zu Ansatz 2.
Bedienbarkeit	+	Taverna Workflow I/O wird durch XML Schema 1:1 abgebildet und eingeschränkt.
Wiederverwendung	-	Der Taverna Server wird nicht für die Ausführung wiederverwendet, die Taverna Workflow Engine wird eingebettet.
Erweiterte Management-Funktionalität	-	Nicht vorhanden.

Tabelle 1: Erfüllungsgrad weiterer Anforderungen (Ansatz 1)

werden. Die verfügbaren Operationen ähneln denen aus Ansatz 1 stark: Um einen beliebigen Taverna-Workflow synchron auszuführen, wird die `execute` Operation verwendet. Das Workflow-Modell, aus dem die Instanz erzeugt werden soll, ist Teil der Eingabe. Zusätzlich wird die Workflow-Eingabe in Form von data port Belegungen übergeben. Die Workflow-Ausgabe wird ebenfalls in Form von data port Belegungen zurückgeliefert. Für die asynchrone Ausführung muss die Workflow-Engine wiederum den Callback-PortType implementieren und wird vom Web service nach dem Aufruf der `start` Operation über das Ergebnis der Ausführung informiert. Auch in diesem Szenario werden Workflow-Instanzen über UUIDs korreliert.

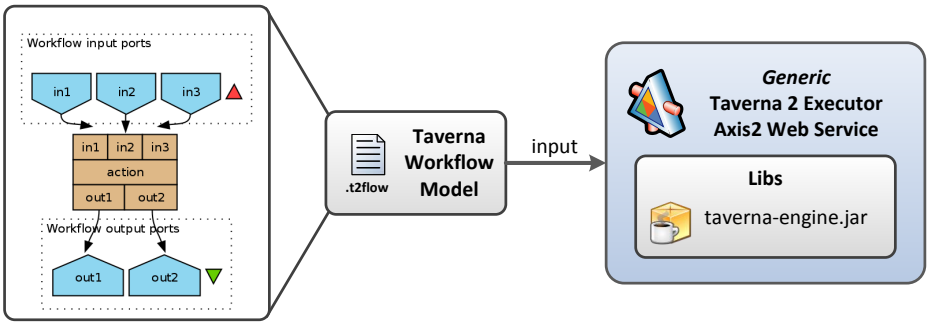


Abbildung 28. Generischer Web service zur Ausführung von Taverna-Workflows

Ansatz 3: Direkte Verwendung des Taverna Servers

Bei diesem Ansatz wird die SOAP-Schnittstelle des Taverna Servers verwendet, um in BPEL-Prozessen Taverna-Workflows auszuführen. Eine neue Softwarekomponente generiert zu diesem Zweck für ein gegebenes Taverna Workflow-Modell BPEL-Fragmente, in denen die nötige Kommunikation mit einer Instanz des Taverna Servers stattfindet.

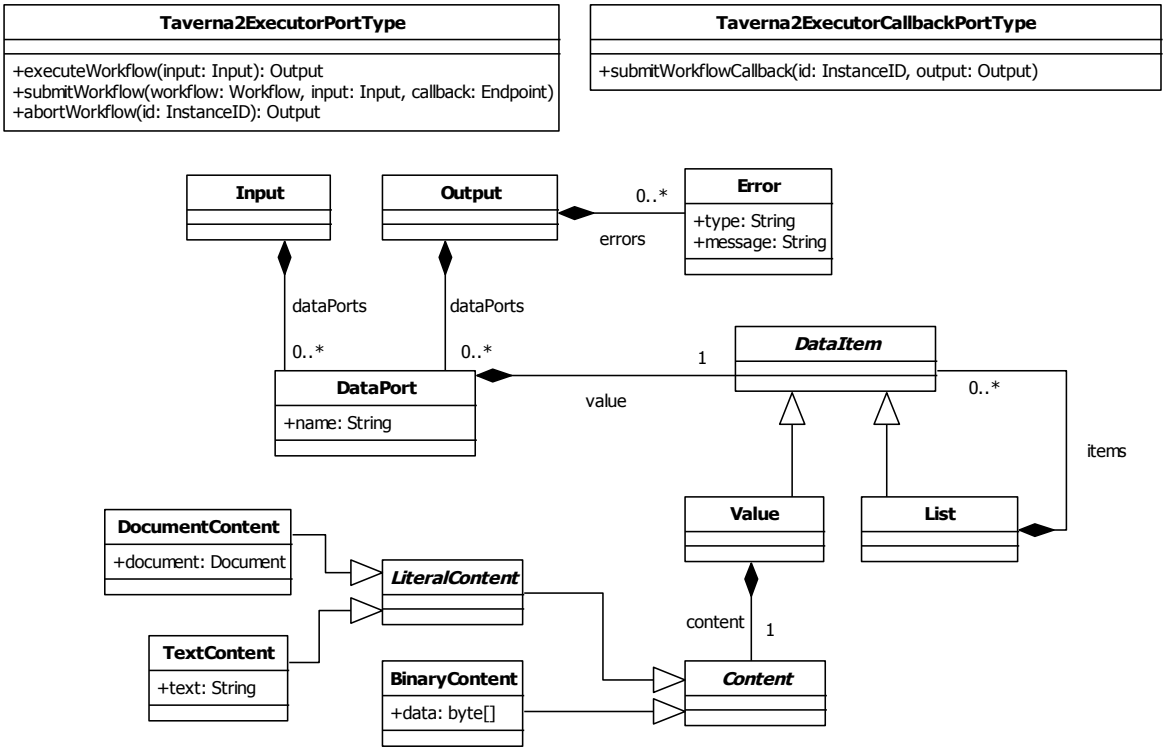


Abbildung 29. Generische Web service Schnittstelle für Taverna-Workflows

Kriterium	Bewertung	Begründung
Standardkonformität	+	Basiert auf Web service Technologie, verwendet JAX-WS.
Plattformunabhängigkeit	+	In jeder Java Virtual Machine ab Java SE 1.6 lauffähig.
Erweiterbarkeit / Änderbarkeit	o	Änderungen wirken sich sofort auf alle neu ausgeführten Taverna-Workflows aus. SimTech hat vollständige Kontrolle über den Quellcode.
Implementierungsaufwand	+	Vergleichsweise gering.
Komplexität des Lösungsansatzes	+	Ansatz ist optimiert auf möglichst einfache Integration in BPEL-Prozesse. Der Web service ist eine simple, generische Lösung, die für alle Taverna Workflows funktioniert.
Bedienbarkeit	o	Es existiert ein XML Schema, welches die Taverna Workflow I/O abbildet. Allerdings ist dieses Schema generisch und es kann vorkommen, dass trotz richtiger Syntax falsche Eingaben an den Workflow übergeben werden (z.B. eine Liste statt eines Skalars).
Wiederverwendung	-	Hier wird ein Teil der Funktionalität des Taverna Servers unter einer besser geeigneten Schnittstelle nachgebaut.
Erweiterte Management-Funktionalität	-	Nicht vorhanden, muss durch Erweiterung hinzugefügt werden.

Tabelle 2: Erfüllungsgrad weiterer Anforderungen (Ansatz 2)

Kriterium	Bewertung	Begründung
Standardkonformität	o	Der Taverna Server bietet eine Web service Schnittstelle über SOAP/HTTP und kann somit in BPEL eingebunden werden. Zur Authentifizierung muss aber eine proprietäre Erweiterung der Apache ODE verwendet werden.
Plattformunabhängigkeit	-	Der Taverna Server muss in Tomcat 6 in einer Unix-artigen Umgebung ausgeführt werden. Andere Umgebungen werden nicht unterstützt.
Erweiterbarkeit / Änderbarkeit	o	Änderungen am Taverna Server müssen aufgrund der LGPL-Bestimmungen der myGrid Community zugänglich gemacht werden. Die Dokumentation für Entwickler ist schlecht oder nicht vorhanden.
Implementierungsaufwand	-	Es müssen zur Integration des Taverna Servers komplexe BPEL-Fragmente erzeugt werden, welche die Interaktion mit dem Server steuern. Die Dokumentation der Taverna Server API ist schlecht, was viel Experimentierungsaufwand zur Folge hat. Im Versuch konnte der Taverna Server nicht erfolgreich in Betrieb genommen werden.
Komplexität des Lösungsansatzes	-	Die Taverna Server API ist schlecht geeignet, um in BPEL-Prozesse eingebunden zu werden. Das Resultat sind lange BPEL-Fragmente für einfachste Operationen. Da kein Callback-Mechanismus existiert, muss Polling zur Abfrage der Ergebnisse implementiert werden.
Bedienbarkeit	-	Der Modellierer von Workflows muss für jede Ausführung eines Taverna-Workflows viele einzelne Web service Aufrufe überblicken. Der einzige Mechanismus, um diese Aufrufe logisch zu gruppieren, ist das BPEL `scope` Element.
Wiederverwendung	+	Der Taverna Server wird verwendet, um Taverna-Workflows auszuführen
Erweiterte Management-Funktionalität	+	Vorhanden: Verwaltung von Workflow-Instanzen und Serverkonfiguration

Tabelle 3: Erfüllungsgrad weiterer Anforderungen (Ansatz 3)

Ansatz 4: Generierung von BPEL-Wrapper-Prozessen

Dieser Ansatz ist eine Erweiterung von Ansatz 2 und 3, bei der trotz Existenz eines generischen Ausführungsdienstes einzelne Taverna-Workflow-Modelle und deren spezifische Eingaben- und Ausgaben auf Web services abgebildet werden (siehe Abbildung 30). Hier generiert eine neue Softwarekomponente aus einer Taverna-Workflowdefinition einen Wrapper-BPEL-Prozess, der wiederum die Ausführung des Workflows an den Taverna Server oder die alternative Implementierung aus Ansatz 2 delegiert. Der auszuführende Taverna-Workflow wird dabei in den generierten BPEL-Prozess eingebettet. Am Erfüllungsgrad der zusätzlichen Anforderungen ändert sich im Vergleich zu Ansatz 2 und 4 nichts, daher verzichten wir an dieser Stelle darauf, die Tabellen zu wiederholen.

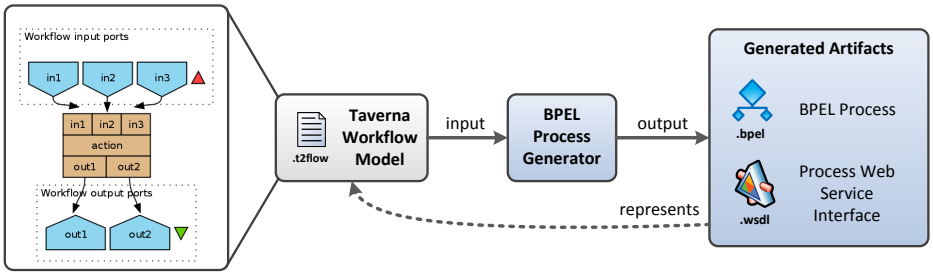


Abbildung 30. Generierung von BPEL-Wrapper-Prozessen

Ansatz 5: Erweiterung der Apache ODE

Die Ausführungsumgebung der BPEL-Prozesse kann theoretisch so erweitert werden, dass sie Taverna-Workflows direkt, d.h. ohne Interaktion mit einem Web service ausführen kann. Dazu muss eine BPEL-Erweiterungsaktivität (siehe Abbildung 31) definiert werden, welche von der ODE erkannt und unterstützt wird. Intern wird die ODE bei einer solchen Erweiterungsaktivität die Ausführung an eine eingebettete Taverna Workflow-Engine delegieren. Mit diesem Ansatz ist man allerdings auf die ODE als Ausführungsumgebung beschränkt, bzw. das Feature muss für jede BPEL-Engine erneut implementiert werden.

```
<bpel:extensionActivity xmlns:t2="http://taverna2.simtech.uni-stuttgart.de">

      <t2:executeWorkflow faultOnError="false" mode="synchronous" maxExecutionTime="PT1H">
            <t2:workflowModel>
                  <t2:resource>http://localhost/resources/my-workflow.t2flow</t2:resource>
            </t2:workflowModel>
            <t2:input  dataPort="cells" from-variable="cells" />
            <t2:output dataPort="cells" to-variable="cells" />
            <t2:errors to-variable="lastErrors"/>
      </t2:executeWorkflow>

</bpel:extensionActivity>
```

Abbildung 31. Variante einer BPEL-Erweiterungsaktivität für das Beispiel aus 5.2

5.4 Auswahl eines Entwurfsansatzes

Die Umsetzung als BPEL- bzw. als ODE-Erweiterung kommt nicht in Frage, da zur Wahrung der Standardkonformität die Lösung als Web service implementiert werden soll. Der Entwurfsansatz zur

Kriterium	Bewertung	Begründung
Standardkonformität	-	Es muss eine BPEL `<extensionActivity>` eingeführt werden, die zunächst nur von der Apache ODE verstanden wird.
Plattformunabhängigkeit	-	Die Lösung ist an die Apache ODE gekoppelt und funktioniert nicht mit anderen BPEL-Engines.
Erweiterbarkeit / Änderbarkeit	o	Der Quellcode der Apache ODE ist verfügbar, benötigt aber Einarbeitung.
Implementierungsaufwand	-	Die Apache ODE muss auf nicht-triviale Weise modifiziert werden, was das Verständnis der ODE-Interna voraussetzt.
Komplexität des Lösungsansatzes	o	Die Einschränkung auf eine bestimmte Workflow-Engine erscheint nicht angemessen.
Bedienbarkeit	+	Aus der Sicht des Workflow-Modelliereres ist dies der einfachste Ansatz, da der gesamte Vorgang der Übergabe von Workflow-Eingaben, Ausführung und Speicherung der Worfklow-Ausgaben in nur einer Aktivität stattfindet. Es gibt keine sichtbare Interaktion mit Drittkomponenten.
Wiederverwendung	-	Der Taverna Server wird bei diesem Ansatz nicht zur Ausführung eingesetzt.
Erweiterte Management-Funktionalität	+	Management-Funktionen der Apache ODE können verwendet werden.

Tabelle 4: Erfüllungsgrad weiterer Anforderungen (Ansatz 5)

Ansteuerung des Taverna Servers (Ansatz 3) ist aus mehreren Gründen unattraktiv: Zum einen könnte die Realisierung aufgrund des hohen Aufwands zur Inbetriebnahme des Servers und zur Fehlerbeseitigung bei der Integration der Server-API zeitlich scheitern. Andererseits kann diese Lösung nur mit zusätzlicher Tool-Unterstützung vom Workflow-Modellierer verwendet werden, da nicht-triviale BPEL-Fragmente generiert werden müssen. Ansatz 1 und 2 sind dagegen simpel genug, um Taverna-Workflows auch händisch per Texteditor oder mit dem bestehenden Modeling Tool in BPEL-Code einzubinden. Ansatz 1 sowie Ansatz 4 scheiden jedoch aus, da sie über zusätzlichen Aufwand eine 1:1 Beziehung zwischen Web service und Taverna-Workflow-Modell herstellen, obwohl sich dieses Merkmal im Verlauf der Arbeit als nicht erwünscht erwiesen hat. **Der zu implementierende Entwurfsansatz ist somit Ansatz 2: ein generischer Web service zur Ausführung beliebiger Taverna-Workflows.**

5.5 Implementierung

In diesem Kapitel wird die Implementierung des generischen Web services zur Ausführung beliebiger Taverna-Workflows (im Folgenden: Taverna 2 Executor) beschrieben. Wir werden dabei weniger auf Implementierungsdetails eingehen, sondern wollen eher die Konzepte hinter der implementierten Lösung vorstellen. Dieses Kapitel soll keinen Ersatz für die mitgelieferte Dokumentation der Software darstellen.

Die wichtigsten Probleme und Lösungen, die im Verlauf der Konstruktion aufgetreten sind, werden hier genannt. Dies betrifft hauptsächlich die Interaktion zwischen BPEL-Prozessen und Web services, die Einbettung der Taverna Workflow Engine in eine fremde Softwarekomponente sowie das Deployment von Web services auf Apache Axis2. Zum Schluss wird die Funktionsweise der fertiggestellten Lösung anhand des Beispiels aus Kapitel 5.2 demonstriert.

Interaktion BPEL-Prozess - Web service

Um die Probleme und Lösungen bei der Kommunikation zwischen dem Web service zur Ausführung von Taverna-Workflows und BPEL-Prozessen zu beschreiben, muss zuerst die generelle Interaktion erklärt werden. BPEL-Prozessinstanzen können über die `invoke`-Aktivität initiativ eine Operation eines dem Prozess bekannten Web services aufrufen. Der konkrete Ort, an dem sich dieser Web service befindet, kann z.B. zum Zeitpunkt der Installation des Prozessmodells auf der BPEL-Engine festgelegt werden. Umgekehrt kann die Außenwelt mit einem BPEL-Prozess über dessen Web-Service-Schnittstelle kommunizieren. Für jedes installierte Prozessmodell startet die BPEL-Engine einen oder mehrere Web services, über deren Operationen eingehende Nachrichten an Prozessinstanzen gesendet werden können (siehe Abbildung 32).

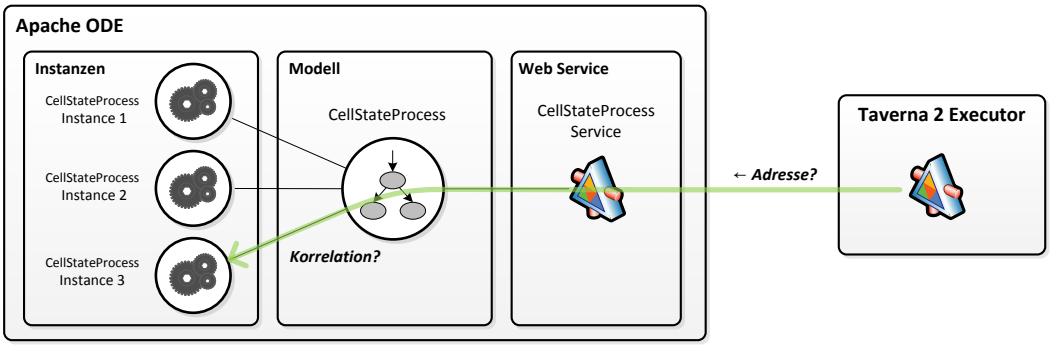


Abbildung 32. Interaktion mit BPEL-Prozess: Korrelation und Callback-Adresse

Durch diesen Mechanismus treten zwei Probleme auf, welche auch die Implementierung des Taverna 2 Executors betreffen: Erstens müssen alle Nachrichten, die an BPEL-Prozesse gesendet werden,

Informationen enthalten, die der BPEL-Engine ermöglichen, die Nachricht einer Prozessinstanz zuzuordnen. Falls keine Zuordnung möglich ist, muss die BPEL-Engine entweder eine neue Prozessinstanz erstellen oder die Nachricht verwerfen. Im Fall der asynchronen Ausführung von Taverna-Workflows (siehe Abbildung 33) wird deshalb von der `submitWorkflow`-Operation eine Instanz-ID zurückgeliefert. Der Callback-Operation, welche von der BPEL-Engine bereitgestellt werden muss, übergibt der Taverna 2 Executor diese Instanz-ID, sodass die BPEL-Engine über den Korrelationsmechanismus jene Prozessinstanz aktivieren kann, welche den ursprünglichen `submitWorkflow`-Aufruf initiiert hat.

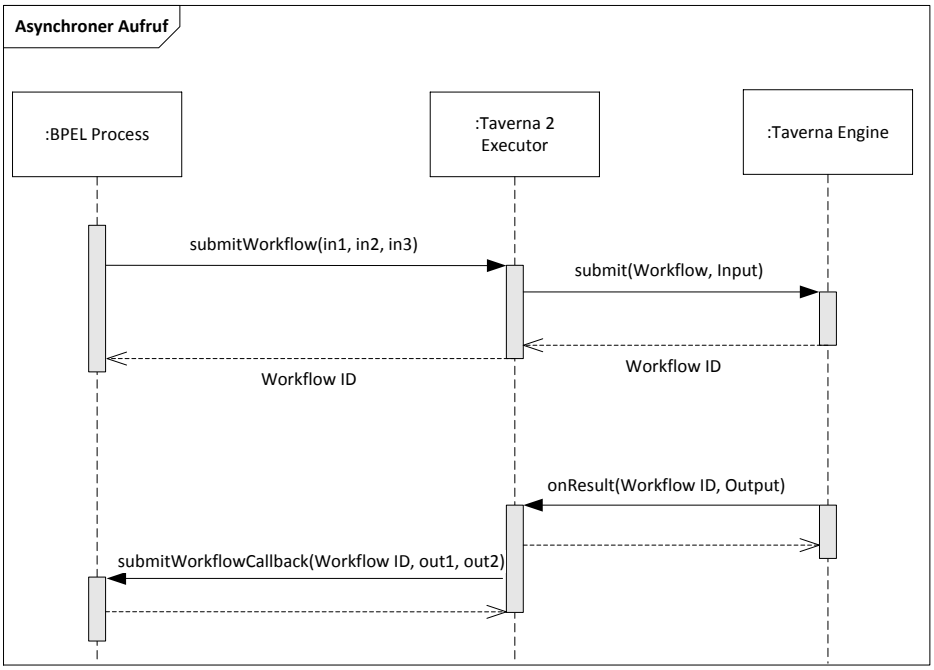


Abbildung 33. Asynchroner Aufruf eines Taverna-Workflows aus einem BPEL-Prozess

Das zweite Problem in diesem Szenario ist die Übergabe der konkreten Callback-Adresse. Soll ein Taverna-Workflow asynchron ausgeführt werden, dann muss der Taverna 2 Executor am Ende der Ausführung eine Nachricht an die Web-Service-Schnittstelle des BPEL-Prozesses senden, um der Prozessinstanz das Ergebnis und die Ausgabe der Ausführung mitzuteilen. Dazu benötigt der Taverna 2 Executor eine Empfängeradresse, die ihm von der Prozessinstanz explizit mitgeteilt werden muss. Dieses Problem kann gelöst werden, indem der BPEL-Prozess zur Laufzeit den Wert eines Partner-Links ausließt, welcher die benötigten Adressdaten enthält. Partner-Links werden in BPEL verwendet, um die Teilnehmer an einer Prozessinstanz abzubilden und ihnen konkrete Verbindungsendpunkte zuzuordnen. Abbildung 34 zeigt die konkrete Umsetzung dieser Lösung in unserem Beispielprozess.

Probleme während der Implementierung

In diesem Abschnitt sollen Ereignisse festgehalten werden, die zu einer nennenswerten Verzögerung oder zum Abbruch eines Implementierungsschrittes geführt haben.

Einbettung der Taverna Workflow Engine Der gewählte Entwurfsansatz sah ursprünglich vor, dass mehrere Instanzen der Taverna Workflow Engine parallel in einem Servlet-Container wie Tomcat ausgeführt werden sollten. Nach etwa einem Drittel des geplanten Implementierungsaufwands stellte sich jedoch heraus, dass dieser Ansatz zum Scheitern verurteilt war. Der Grund dafür ist die intensive Nutzung des Singleton-(Anti-)Patterns innerhalb der Taverna Kernklassen. Basismechanismen wie die Datenbank-Konfiguration sind so implementiert, dass es je JVM nur eine Instanz davon geben kann. Die Einbettung der Workflow Engine in den Taverna 2 Executor war somit nicht praktikabel.

```
<bpel:assign>
      <bpel:copy>
            <bpel:from>
            <bpel:literal>
              <!-- ... -->
                  <t2t:callback>
                    <t2t:soap version="1.2" style="document-literal" address="" />
                  </t2t:callback>
            </bpel:literal>
            </bpel:from>
            <bpel:to>$submitWorkflowRequest.payload</bpel:to>
      </bpel:copy>
      <bpel:copy>
            <!-- Kopiere Verbindungsendpunkt-Adresse aus Partner-Link in callback-Element -->
            <bpel:from partnerLink="t2executor" endpointReference="myRole"></bpel:from>
            <bpel:to>$submitWorkflowRequest.payload//*[local-name()="soap"]/@address</bpel:to>
      </bpel:copy>
</bpel:assign>
```

Abbildung 34. Vorbereitung der Callback-Informationen im BPEL-Prozess

Erschwerend kam hinzu, dass die Softwarebibliothek JDOM 1.0 durch Taverna auf den Java Classpath geladen wird. Dadurch kommt es in Verbindung mit JAX-WS zu einem Fehler, der verhindert, dass ausgehende SOAP-Nachrichten gesendet werden können. Ein XML-Namespace-Präfix wird in dieser Konstellation mehrfach vergeben, was das XML-Dokument ungültig macht.

Die Lösung dieses Problems besteht darin, die Ausführung von Taverna-Workflows an einen separaten Prozess mit eigener JVM zu delegieren, anstatt die Taverna Workflow Engine in die eigene Software einzubetten. Nach einigen Versuchen stellte sich heraus, dass die Verwendung der Taverna Command Line die einfachste Lösung darstellt. Die Umstellung auf den Start eines neuen Taverna Command Line Prozesses je Workflow-Instanz bringt allerdings Performanceprobleme mit sich: Da für den Durchlauf einer neuen Workflow-Instanz eine JVM hochgefahren werden muss, verzögert sich der Start um etwa fünf Sekunden.

Deployment auf Apache Axis2 Axis2 ermöglicht nicht, in einer Web service Implementierung wie dem Taverna 2 Executor Softwarebibliotheken zu verwenden, die gleichzeitig von Axis2 in einer anderen Version eingebunden werden. Verwendet man dennoch eine andere Version, können subtile Fehler auftreten, da nicht die Bibliothek aktiv ist, gegen die ursprünglich entwickelt wurde. In diesem konkreten Fall wurde vom Taverna 2 Executor die aktuelle Version 2.4 der Apache Commons IO Bibliothek verwendet, während Axis2 die veraltete Version 1.4 im Classpath bereitstellte. Dies führte dazu, dass nach einem eigentlich erfolgreichen Durchlauf einer Taverna-Workflow-Instanz aufgrund einer nicht vorhandenen Methode ein Fehler auftrat, welcher den Durchlauf zum Scheitern brachte. Die Suche nach der Fehlerursache gestaltete sich frustrierend und vor allem zeitaufwändig.

Das Problem konnte jedoch gelöst werden, indem bei der Erstellung des Axus2-Servicearchivs das Maven Shade Plugin verwendet wird. Mit diesem Build-Plugin ist es durch Bytecode-Manipulation möglich, die konfliktbeladenen Klassen in einen anderen Namensraum zu verschieben, sofern im Vorfeld bekannt ist, welche Klassen Probleme verursachen. Anstelle der konfliktbeladenen Klasse `org.apache.commons.io.IOUtils` lädt der Taverna 2 Executor jetzt die konfliktfreie Klasse `notaxis2.org.apache.commons.io.IOUtils` (siehe Abbildung 35).

```
<configuration>
      <relocations>
            <relocation>
                  <pattern>org.apache.commons</pattern>
                  <shadedPattern>notaxis2.org.apache.commons</shadedPattern>
            </relocation>
      </relocations>
</configuration>
```

Abbildung 35. Konfiguration des Maven Shade Plugins für Axis2

Projektstruktur

Die konstruierte Integrationslösung besteht aus einer Reihe von Maven-Projekten, die einen Abhängigkeitsgraphen bilden (siehe Abbildung 36). Auf der untersten Ebene liegt das Projekt `t2-types`, welches ausschließlich generierte Java-Bindings für häufig verwendete XML-Datenstrukturen und das `t2flow` Format enthält. Alle XML-Bindings werden für JAXB 2.1 generiert. Eine Ebene darüber befindet sich das Projekt `t2-executor-service-api`, welches den Schnittstellenteil des Taverna 2 Executors enthält. Die Java Klassen werden per `wsimport` aus den WSDL-Definitionen generiert und enthalten noch keine Funktionalität. Die eigentliche Implementierung liegt im Projekt `t2-executor-service-impl`. Auf dieser Implementierung bauen zwei verschiedene Distributions-Artefakte für den Web service auf: eines für Apache Axis2 und ein weiteres für die JAX-WS Referenzimplementierung Metro. Beim Projekt `t2-bpel-process-generator` handelt es sich um ein obsoletes Projekt zur Generierung von BPEL-Prozessen anhand der I/O-Schnittstellen von Taverna-Workflow-Modellen. Damit sollte ursprünglich Ansatz 4 aus Kapitel 5.3 umgesetzt werden, was aber nicht weiter verfolgt wird.

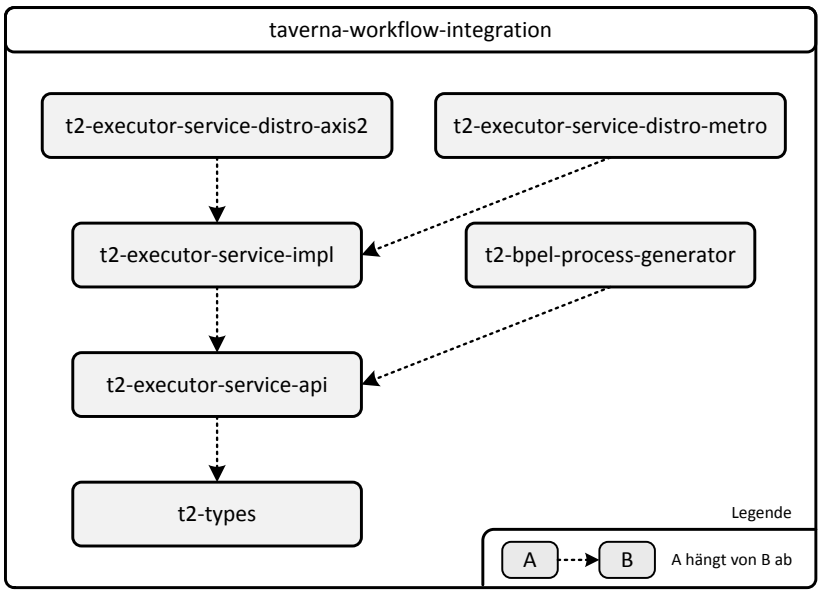


Abbildung 36. Projektstruktur und Abhängigkeiten

Klassenstruktur

Anstelle einer erschöpfenden Darstellung aller Klassen soll hier eine kompakte Beschreibung der zentralen Elemente der Implementierung gegeben werden. Ausgangspunkt der Beschreibung ist die Klasse `Taverna2ExecutorService` (siehe Abbildung 37). Diese Klasse implementiert alle funktionalen Aspekte des Taverna 2 Executors und verwendet dazu bei Bedarf spezialisierte Komponenten, die zusammengehörige Funktionalität bündeln. Im Moment existiert mit dem `WorkflowInstanceManager` genau eine solche Komponente. Der `WorkflowInstanceManager` hat die Aufgabe, eine Menge von Taverna-Workflow-Instanzen zu verwalten. Seine Funktionsweise ähnelt der eines Java `ExecutorService`: Neue Workflow-Instanzen können über die `submitWorkflow`-Methode zur asynchronen Ausführung übergeben werden. Der Aufrufer erhält ein `Future`-Objekt zurück, mit dem sich das Ergebnis der Ausführung abfragen lässt. Zusätzlich kann der Aufrufer ein Callback registrieren lassen, welches nach erfolgreicher oder fehlerhafter Beendigung des Durchlaufs aufgerufen wird. Über diesen Callback-Mechanismus wird auch die Rückmeldung des Ausführungsergebnisses an die aufrufende BPEL-Prozessinstanz realisiert.

Die Schnittstelle `WorkflowInstance` repräsentiert das Wissen, wie Taverna-Workflows auszuführen sind und hat die Aufgabe, jeder Workflow-Instanz eine Identität sowie einen Ausführungszustand zuzuordnen. Momentan existiert mit der `CommandLineWorkflowInstance` nur eine konkrete Ausprä-

gung dieser Schnittstelle. Eine `CommandLineWorkflowInstance` verwendet eine lokale Installation der Taverna Command Line zur Ausführung der Taverna-Workflows. Ein- und Ausgaben der Workflows werden über das lokale Dateisystem geschrieben und gelesen. Der `WorkflowInstanceManager` sorgt für die nötige Isolation zwischen verschiedenen Workflow-Instanzen, indem er jeder Instanz ein privates Arbeitsverzeichnis zuweist.

Die Klasse `Taverna2ExecutorService` trägt selbst keine `@WebService`-Annotation, sondern dient als Basis für konkrete Service-Distributionen, die von der Klasse erben und mit `@WebService` annotiert werden. Die abgeleiteten Distributionsklassen können keine Funktionalität überschreiben, sondern sind lediglich dazu gedacht, Anpassungen an die Eigenheiten der Ziel-Installationsumgebung wie Axis2 oder Metro zu implementieren.

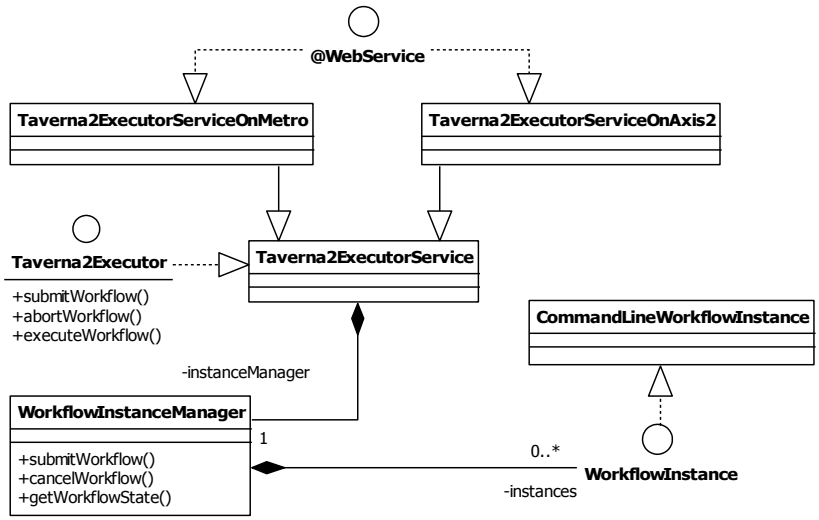


Abbildung 37. Vereinfachtes Klassendiagramm der Implementierung

Anwendung auf das Beispiel

Um das Implementierungskapitel abzuschließen, wird an dieser Stelle der vollständig integrierte Beispielprozess mit einem Taverna-Bestandteil vorgestellt. Zur Erinnerung: Das Beispiel simuliert für eine gegebene Startkonfiguration eine Menge von Zeitschritten in Conway's Game of Life. Die Einzelschritt-Logik ist als Taverna-Workflow implementiert und wird vom umgebenden BPEL-Prozess in einer Schleife aufgerufen. Anhand der Markierungen in Abbildung 38 kann die Funktionsweise des Taverna 2 Executors nachvollzogen werden: Da es sich um einen asynchronen Aufruf mit Callback handelt, wird in Schritt (1) zunächst die Verbindungsendpunkt-Adresse der Prozessinstanz aus dem zugehörigen Partner-Link extrahiert. In Schritt (2) wird innerhalb der Schleife per `invoke`-Aktivität die `submitWorkflow`-Operation des Taverna 2 Executors aufgerufen. Der auszuführende Workflow, der aktuelle Zustand des Zellgitters und die Callback-Informationen werden als Parameter mitgegeben. Im dritten Schritt startet der Taverna 2 Executor den Taverna-Workflow, indem er die Ausführung an die Taverna Command Line delegiert. Nachdem die Ausführung beendet ist, wird die BPEL-Prozessinstanz in Schritt (4) durch den Taverna 2 Executor über das Ausführungsergebnis benachrichtigt. Der Taverna 2 Executor sendet dazu die Instanz-ID sowie den neuen Zustand des Zellgitters an die in Schritt (1) hinterlegte Callback-Adresse. Durch die Angabe der Instanz-ID ist die BPEL-Engine in der Lage, die Antwort der korrekten Prozessinstanz zuzuordnen. Als letzte Aktion jedes Schleifendurchlaufs ruft die BPEL-Prozessinstanz per `invoke`-Aktivität einen Visualisierungs-Web-Service auf, welcher das zuletzt berechnete Zellgitter auf dem lokalen Rechner graphisch darstellt.

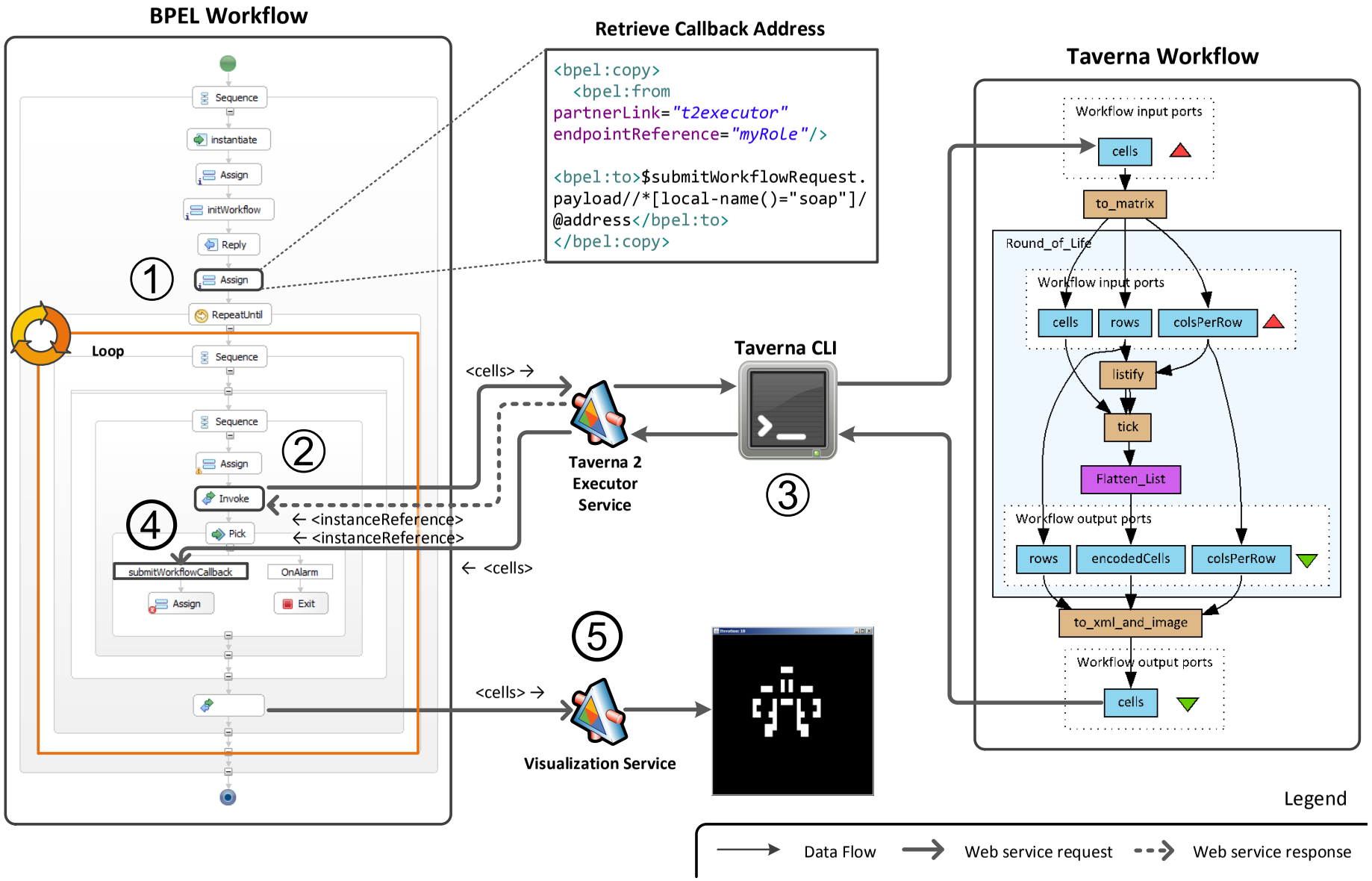


Abbildung 38. Vollständiges Beispiel unter Verwendung des Taverna 2 Executors

6 Ergebnis

In dieser Arbeit wurden die eScience-Plattformen nanoHUB und myExperiment sowie das wissenschaftliche WfMS Taverna auf Möglichkeiten zur Interaktion mit der SimTech Workflow-Umgebung untersucht. Verschiedene Integrationsansätze wurden herausgearbeitet und einer Bewertung unterzogen. Die Untersuchung hat ergeben, dass die gewünschte Auslagerung von SimTech-Anwendungen und Infrastruktur auf die Ressourcen der Plattformbetreiber leider nicht machbar ist, da kein geeignetes Angebot existiert. Als beste verbleibende Lösung wurde die Integration des Taverna WfMS ausgewählt. Mehrere Entwurfsansätze zur Integration der Taverna Workflow Engine wurden aufgestellt und miteinander verglichen. Mit der konstruierten Integrationslösung lassen sich die datenflussorientierten Taverna-Workflows als Bestandteil der BPEL-basierten Simulation Workflows ausführen.

Die Schwierigkeit der Integrationsaufgabe wurde unterschätzt. Eine Einbettung der Taverna Workflow Engine in eine Web service Implementierung war geplant, konnte aber aufgrund technischer Schwierigkeiten nicht durchgeführt werden. Der Lösungsansatz musste durch eine Alternative mit Performanceeinbußen ersetzt werden. Für lange laufende Taverna-Workflows ist die Verlangsamung jedoch nicht signifikant.

6.1 Ausblick

Ein wichtiges Ziel der Arbeit war die Suche nach Hosting-Gelegenheiten für die SimTech Workflow-Umgebung. Da unter den betrachteten Plattformen keine zufriedenstellende Möglichkeit gefunden wurde, könnten sich Folgearbeiten mit anderen eScience-Plattformen oder Hosting-Diensten befassen. Was die erfolgte Taverna-Integration betrifft, so bietet sich eine Reihe von Verbesserungsmöglichkeiten und Fortsetzungsarbeiten an:

Performance-Verbesserungen

In der vorliegenden Lösung wird für jede Workflow-Instanz eine neue JVM erzeugt. Ein einzelner, dedizierter Betriebssystem-Prozess wäre aber bereits ausreichend, um die Probleme der fehlgeschlagenen Einbettung zu umgehen. Der Taverna 2 Executor könnte die Ausführung des Workflows z.B. durch geeignete Inter-Prozess-Kommunikation an diesen Prozess delegieren, anstatt die Taverna Command Line zu verwenden. Noch besser wäre es natürlich, wenn es einem (fähigeren) Programmierer gelänge, die Taverna Workflow Engine trotz der genannten Probleme einzubetten.

Hinzufügen von Management-Funktionalität

In einer Folgearbeit können Management-Funktionen zur erweiterten Verwaltung von Workflow-Instanzen in den Taverna 2 Executor eingebaut werden. Dazu müssen natürlich zunächst die Anforderungen an die zu erstellenden Erweiterungen erhoben werden. Mit dem bestehenden Code lassen sich einfache Funktionen wie die Auflistung aller Instanzen und deren Ausführungsstatus schnell realisieren, da die benötigten Datenstrukturen bereits vorhanden sind.

Verbesserung der Callback-Mechanik

Im Moment unterstützt der Taverna 2 Executor Service zwei Callback-Mechanismen: die automatische Suche eines passenden Verbindungsendpunktes anhand eines gegebenen WSDL-Dokuments und die Übergabe einer Endpunkt-URL. Die Datenstrukturen des WS-Addressing-Standards werden bisher vom Dienst noch nicht verstanden. Unterstützung für diese und weitere Verbindungsendpunkt-Beschreibungen könnten mit weniger als 20 Stunden Aufwand hinzugefügt werden.

Evaluation des Taverna Servers

Falls genügend Zeit zur Verfügung steht, kann geprüft werden, ob eine Implementierung des Entwurfsansatzes 3 aus Kapitel 5.3 zufriedenstellende Ergebnisse liefert. Eine Implementierung auf Basis des Taverna Servers könnte den Taverna 2 Executor ersetzen und die Erstellung eines Eclipse-Serveradapters rechtfertigen.

Integration weiterer Taverna-Komponenten

Mit der Möglichkeit, Taverna-Workflows in Simulation Workflows einzusetzen, können darauf aufbauende Integrationsansätze wie z.B. die Einbindung des myExperiment Workflow Repositories oder die Eingliederung der Workbench durchgeführt werden.

Abbildungsverzeichnis

Abkürzungen

API	Application Programming Interface
BPEL	Business Process Execution Language
EDV	Elektronische Datenverarbeitung
EPSRC	Engineering and Physical Sciences Research Council
HTTP	Hypertext Transfer Protocol
JAXB	Java Architecture for XML Binding
JAX-WS	Java API for XML Web Services
JVM	Java Virtual Machine
LGPL	GNU Lesser General Public License
OASIS	Organization for the Advancement of Structured Information Standards
ODE	Orchestration Director Engine
SFTP	Secure File Transfer Protocol
SCUFL	Simple Conceptual Unified Flow Language
VNC	Virtual Network Computing
W3C	World Wide Web Consortium
WebDAV	Web-based Distributed Authoring and Versioning
WfMS	Workflow Management System
WSDL	Web Services Description Language
WWW	World Wide Web
XML	Extensible Markup Language

Literatur

[1] Alexandre Alves et al. *OASIS Web Services Business Process Execution Language (WS-BPEL) 2.0.* URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

[2] Bundesministerium für Wirtschaft und Arbeit und Bundesministerium für Bildung und Forschung. *Informationsgesellschaft Deutschland 2006, Aktionsprogramm der Bundesregierung.*

[3] David Booth et al. *Web Services Architecture.* URL: http://www.w3.org/TR/ws-arch/.

[4] Tim Bray et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition).* URL: http://www.w3.org/TR/xml/.

[5] David De Roure, Carole Goble und Robert Stevens. "The Design and Realisation of the myExperiment Virtual Research Environment for Social Sharing of Workflows". In: *Future Generation Computer Systems* 25 (2008), S. 561–567.

[6] Raymond Dormien. "Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen". Diplomarbeit. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2011, S. 67.

[7] David C. Fallside und Priscilla Walmsley. *XML Schema Part 0: Primer Second Edition.* URL: http://www.w3.org/TR/xmlschema-0/.

[8] *Forschungsfelder SRC SimTech und Exzellenzcluster Simulation Technology.* URL: http://www.simtech.uni-stuttgart.de/forschung/forschungsfelder/index.html; http://archive.is/YZca7.

[9] Ian Foster und Carl Kesselman, Hrsg. *The grid: blueprint for a new computing infrastructure.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[10] K. Görlach et al. "Guide to e-Science". In: Hrsg. von Y. Yang, L. Wang und W. Lie. Springer Verlag, 2011. Kap. Conventional Workflow Technology for Scientific Simulation.

[11] Martin Gudgin et al. *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).* URL: http://www.w3.org/TR/soap12-part1/.

[12] Michael Hahn und Karolina Vukojevic. *Prototyp SWMS.* SimTech. 2013.

[13] N.H. Kapadia und J. A B Fortes. "On the design of a demand-based network-computing system: the Purdue University Network-Computing Hubs". In: *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on.* 1998, S. 71–80.

[14] Kohsuke Kawaguchi, Sekhar Vajjhala und Joe Fialli. *The Java™ Architecture for XML Binding (JAXB) 2.2.* 2009.

[15] G. Klimeck et al. "nanoHUB.org: Advancing Education and Research in Nanotechnology". In: *Computing in Science Engineering* 10.5 (2008), S. 17–23.

[16] Jitendra Kotamraju. *The Java API for XML-Based Web Services (JAX-WS) 2.2 Rev a.* 2011.

[17] Frank Leymann und Dieter Roller. *Production workflow: concepts and techniques.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[18] M. McLennan und R. Kennell. "HUBzero: A Platform for Dissemination and Collaboration in Computational Science and Engineering". In: *Computing in Science Engineering* 12.2 (2010), S. 48–53.

[19] nanoHUB.org. *Workspace.* 2006. URL: https://nanohub.org/resources/1242; http://archive.is/WeS4P.

[20] Thomas Oinn et al. "Taverna: lessons in creating a workflow environment for the life sciences". In: *Concurrency and Computation: Practice and Experience* 18.10 (2006), S. 1067–1100.

[21] *PN 8: Integrated data management, workflow and visualisation to enable an integrative systems science.* URL: http://www.simtech.uni-stuttgart.de/forschung/pn/pn8/index.html; http://archive.is/YGs6L.

[22] Jens Rutschmann. "Generisches Web Service Interface um Simulationsanwendungen in BPEL-Prozesse einzubinden". Diplomarbeit. Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Germany, 2009, S. 116.

[23] Research Council e Science Core Programme. *Defining e-Science.* URL: http://www.nesc.ac.uk/nesc/define.html; http://archive.is/AHBoF.

[24] I.J. Taylor et al. *Workflows for E-Science: Scientific Workflows for Grids.* Springer-Verlag London Limited, 2007.

[25] Sanjiva Weerawarana et al. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

[26] Katherine Wolstencroft et al. "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud". In: *Nucleic Acids Research* 41.W1 (2013), W557–W561.

[27] Paul Wouters. *What is the matter with e-Science? – thinking aloud about informatisation in knowledge creation.* 2006. URL: http://www.pantaneto.co.uk/issue23/wouters.htm; http://archive.is/y3ufg.

[28] Sharon Biocca Zakhour, Sowmya Kannan und Raymond Gallardo. *The Java Tutorial: A Short Course on the Basics.* 5. Aufl. Addison-Wesley Professional, 2013.

Alle Links wurden zuletzt am 13. Juni 2013 überprüft.

Erklärung

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

Ort, Datum	*Unterschrift*