Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Studienarbeit Nr. 2414

Migrating the SimTech Workflow Management System to a Cloud Infrastructure

Lukas Reinfurt

Studiengang: Informatik

Prüfer: Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer: Karolina Vukojevic
begonnen am: 03.12.2012
beendet am: 03.06.2013

CR-Klassifikation: C.2.4, D.2.11, H.4.1, I.6.7

Abstract

The SimTech Simulation Workflow Management System has been developed to allow scientist to easily create, manage, and execute simulation workflows. It is currently executed on local computers or in traditional data centers, but it could benefit from the scalability and utility computing model that cloud computing offers. This thesis investigates the feasibility of migrating the SimTech Simulation Workflow Management System to the cloud. Cloud service models and service providers are inspected and a prototypical migration into Amazon's cloud environment is realized. The result is that all but two component can be migrated into the cloud with no or very small changes necessary. Of these components, all but two are also widely offered as PaaS solutions. In conclusion, a significant part of the system could be migrated to a cloud infrastructure right now, but some work is necessary to utilize the full potential of the cloud. For a full migration, one component has to be heavily modified or redeveloped.

Table of Contents

List of Figures

List of Tables

List of Listings

List of Abbreviations

Amazon EC2	Amazon Elastic Compute Cloud, page 46
Amazon RDS	Amazon Relational Database Service, page 61
Amazon SNS	Amazon Simple Notification Service, page 39
Amazon SQS	Amazon Simple Queue Service, page 39
AMI	Amazon Machine Image, page 64
AWS	Amazon Web Service, page 19
BPaaS	Business-Process-as-a-Service, page 21
BPEL	Business Process Execution Language, page 13
BPMN	Business Process Modeling Notation, page 13
CaaS	Components as a Service, page 21
DaaS	Data-Storage as a Service, page 21
DNS	Domain Name System, page 63
EaaS	Everything as a Service, page 20
EBS	Elastic Block Storage, page 47
GUI	Graphical User Interface, page 36
HuaaS	Humans as a Service, page 21
IAAS	Institute of Architecture of Application Systems, page 12
IaaS	Infrastructure as a Service, page 20
JMS	Java Message Service, page 38
NIST	National Institute of Standards and Technology, page 18
ODE-PGF	ODE Pluggable Framework, page 16

PaaS	Platform as a Service, page 20
RAP	Remote Application Platform, page 30
SaaS	Software as a Service, page 21
SCP	Secure Copy, page 51
SIMPL	Simulation Data Management System, page 14
SimTech	Simulation Technology, page 12
SimTech Modeler	SimTech Workflow Modeling & Monitoring Tool, page 15
SimTech SWfMS	SimTech Simulation Workflow Management System, page 10
SLA	Service Level Agreement, page 18
SOA	Service Oriented Architecture, page 21
SSH	Secure Shell, page 50
VNC	Virtual Network Computing, page 36
VPN	Virtual Private Network, page 49
WS	Web Service, page 21
XaaS	Anything as a Service, page 20

1 Introduction

In the last years, cloud computing emerged as an alternative to the classic data center approach to hosting distributed applications. This development is especially beneficial for many applications that have unpredictable or fluctuating usage patterns. In the past, planning for these patterns was difficult and would often lead to over-provisioning, and hence wasting resources, or under-provisioning, resulting in bad performance and lost opportunities. The high flexibility, easy scalability, and a pay-per-use model offered by cloud computing makes it a better fit for those systems than the more rigid data centers. To benefit from cloud computing, existing applications have to be migrated into this new environment. Depending on the applications architecture and functionality, this can be a more or less demanding task that has to be carefully planned and executed.

1.1 Task of this Student Thesis

The task of this student thesis is to analyze, if and how components of the SimTech Simulation Workflow Management System (SimTech SWfMS) can be migrated to a cloud infrastructure. Questions of architectural and technical feasibility, as well as licensing, security and economics should be investigated. Cloud providers and cloud service models should be examined for compatibility with all components. Necessary changes to the software to allow a migration to the cloud should be explained. Selected components should be prototypically migrated to a cloud environment.

1.2 Structure of this Document

Chapter 2 introduces the SimTech and SimTech SWfMS. First, some background information on SimTech, workflows, and workflow management systems is presented. Then, the functionality and architecture of the SimTech SWfMS is described in more detail.

Chapter 3 provides an introduction to cloud computing. A definition of cloud computing is presented and its essential characteristics are explained. Cloud service and deployment models, as well as cloud offerings and cloud application architecture are also described.

Chapter 4 explains the migration process. The reasons for a migration are presented and the general approach to software migration is described. Some aspects more specific to cloud migration are detailed.

Chapter 5 assesses the migratability of the SimTech SWfMS. First, a possible migration of each component is investigated. A fitting service model for each component is selected and reasonable groups of components are devised.

Chapter 6 describes the prototypical migration of the SimTech SWfMS. First, necessary changes to the software are presented. Then, the setup and execution in the cloud is explained in more detail. Problems and possible improvements are also mentioned.

Chapter 7 summarizes the previous chapters. A conclusion and future perspectives are given.

2 SimTech Simulation Workflow Management System

The SimTech SWfMS is a software prototype that is developed by the Institute of Architecture of Application Systems (IAAS)[1] at the University Stuttgart. It allows scientists to easily create, manage, and execute simulation workflows. Its background, functionality, and architecture are described in the following sections.

2.1 Background

In 2005, the German federal and state government initiated the Excellence Initiative[2] with the goal to promote cutting-edge research and to raise international competitiveness of German universities. A competition was run by the German Research Foundation and the German Council of Science and Humanities, in which universities could submit proposals in three areas. 39 graduate schools, 37 clusters of excellence, and nine institutional strategies where selected and received a total of 1.9 billion euros of funding over five years [18, pp. 12-13]. In 2012, funding decisions for the second phase of the Excellence Initiative where announced, in which 45 graduate schools, 43 clusters of excellence, and 11 institutional strategies will now receive 2.4 billion euros in funding [12].

2.1.1 SimTech

Simulation Technology (SimTech) is one of the clusters of excellence that got funding by the Excellence Initiative in both the first and the second phase. It is a cooperation between the University of Stuttgart, the German Aerospace Center, the Fraunhofer Institute, and the Max Planck Institute. Its aim is to enhance existing simulation strategies and create new solutions [18, p. 91].

SimTech is divided in seven individual research areas that span various fields, such as chemistry, physics, mathematics, philosophy, social studies, and computer science (and many more) [26, p. 7, also see 37]. These research areas collaborate in nine project net-

[1] http://www.iaas.uni-stuttgart.de/indexE.php
[2] http://www.dfg.de/foerderung/programme/exzellenzinitiative/

works[3], one of which is project network 8: *Integrated data management, workflow and visualization to enable an integrative systems science*[4]. This is a joint effort of two research areas, *Integrated data management and interactive visualization* (area E)[5], and *Hybrid high-performance computing systems and software engineering* (area F)[6].

The goal of this project network is to create an infrastructure that allows non computer scientist to easily work with simulation workflows. This infrastructure should be able to incorporate existing visualization software and techniques, various different data sources, existing simulation workflow fragments, as well as the possibility for human interactions [34].

2.1.2 Workflows and Workflow Management Systems

Hollingsworth defines workflows as "the computerised facilitation or automation of a business process, in whole or part" [22, p. 6]. A workflow is defined by a process model, which describes activity steps or tasks and the order in which they should be executed [22, pp. 7-8]. It is common to visually represent process models with graphs, as shown in Figure 2.1. Here, vertices represent tasks and the edges describe the progression between the tasks.

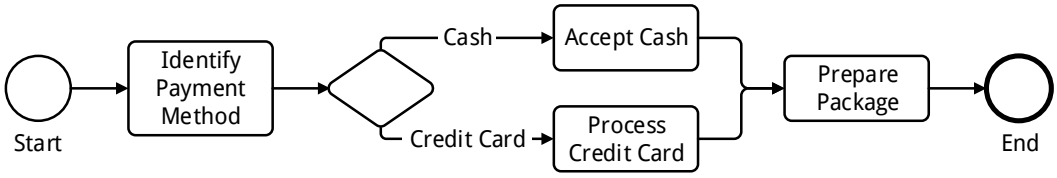

Figure 2.1: A simple ordering workflow represented as a graph.

Workflows are executed by workflow management systems. Hollingsworth defines a workflow management system as "a system that completely defines, manages and executes 'workflows' through the execution of software whose order of execution is driven by a computer representation of the workflow logic" [22, p. 6].

In the past, workflows were predominantly used by businesses to describe and automate their processes, which include human and machine-based activities. This resulted in the creation of business-centered standards, such as the Business Process Execution Language (BPEL)[7] or the Business Process Modeling Notation (BPMN)[8]. In recent years, however, workflows have been used more and more in a scientific context. These scientific workflows allow scientists to handle increasingly complex computation and data analysis tasks that

[3] http://www.simtech.uni-stuttgart.de/forschung/pn/index.en.html
[4] http://www.simtech.uni-stuttgart.de/forschung/pn/pn8/index.en.html
[5] http://www.simtech.uni-stuttgart.de/forschung/forschungsfelder/rae/index.en.html
[6] http://www.simtech.uni-stuttgart.de/forschung/forschungsfelder/raf/index.en.html
[7] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
[8] http://www.bpmn.org/

could span a multitude of different computation models, data sources, and execution environments [20].

Simulation workflows are a subcategory of scientific workflows. They are used to run simulations on often huge amounts of data, for example, to produce weather forecasts, or to simulate structural changes within human bones [39].

2.2 Functionality

The aim of the SimTech SWfMS is to provide a tool for scientists to easily create, manage, and execute simulation workflows [19].

The creation, or modeling aspect of simulation workflows is based on the existing BPEL language. To support the requirements of scientists and simulation workflows, extensions to the BPEL language have been developed. For example, data references allow data to be passed by reference instead of value, which avoids decreasing performance and overloading of the workflow management system by the often huge amounts of data that scientific workflows have to process [also see 45]. Other extensions include simulation workflows with shared context, to package common context information of multiple simulation workflows, and data streams, for example, to integrate sensor data [32].

For the execution side, a framework has been developed to extend existing BPEL workflow management systems. These extensions include a service bus for simulation workflows, which invokes services that implement workflow activities. Here, flexibility mechanisms like late binding and rebinding are important, as well as support for legacy simulation software. Therefore, another extension is a generic wrapper for simulation frameworks and applications that enables existing legacy simulation applications to run as Web Services. Also in development is the Simulation Data Management System (SIMPL), which provides unified access methods for arbitrary external data [38].

To make workflows suitable for simulations, two key areas are also extended, namely flexibility and human users. Flexibility is a key requirement for simulation workflows, which differ from business workflows in that they are often developed in a “model as you go” approach. Therefore, simulation workflows need to support suspending, adapting, and resuming existing workflows. Additionally, concurrent workflow evolution (supporting several instances of a model), iteration and re-execution of workflow logic, as well as breakpoints are supported by the SimTech SWfMS [19].

Human user involvement in simulation workflows, for example, for decision making or data manipulation and workflow repair, is also important for simulation workflows. Therefore, human tasks can be added to workflows and a simulation task manager handles these task during execution, for example, by providing a user interface for manual input or by notifying a user that his action is required [23].

2.3 Architecture

The SimTech SWfMS consists of three parts. Part one is SimTech Workflow Modeling & Monitoring Tool (SimTech Modeler), which is based on Eclipse JEE[9]. Figure 2.2 shows the SimTech Modeler user interface.

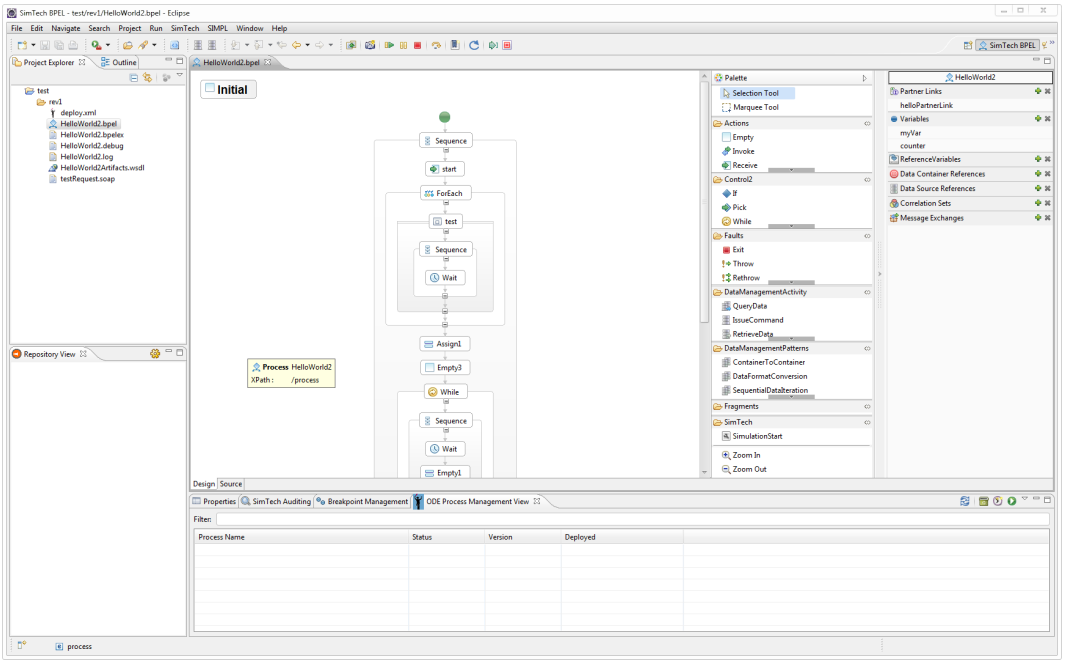

Figure 2.2: The SimTech Modeler user interface.

Part two is an application server, in this case Apache Tomcat[10], which runs various components that execute the workflows and corresponding simulations, as well as some management tasks. Part three is the SimTech Auditing Application, which makes persistent copies of events generated during workflow execution. These three parts communicate via Web Services and through a messaging middleware, Apache ActiveMQ[11].

The bottom half of Figure 2.3 shows that functionality of Eclipse is extended by five components to form the SimTech Modeler. The central component is the SimTech BPEL Designer, which enables the SimTech Modeler to model and manage simulation workflows. These workflows can be divided in smaller parts, so called fragments, which can be reused in other workflows. The SimTech Fragment Extraction component handles the extraction of such fragments. Fragments can be stored in a central repository, which allows them to

[9] http://www.eclipse.org/downloads/moreinfo/jee.php
[10] http://tomcat.apache.org/
[11] http://activemq.apache.org/

be easily shared. This Repository, called Fragmento[12], is one of the four applications running on the application server, as shown in the top left of Figure 2.3. Communication with Fragmento is handled by the FragmentoRCP component inside SimTech Modeler. This component provides the functionality to upload, download, and manage fragments, by calling a Web Service provided by Fragmento. Fragmento, in turn, stores and retrieves fragments in a PostgreSQL[13] database.

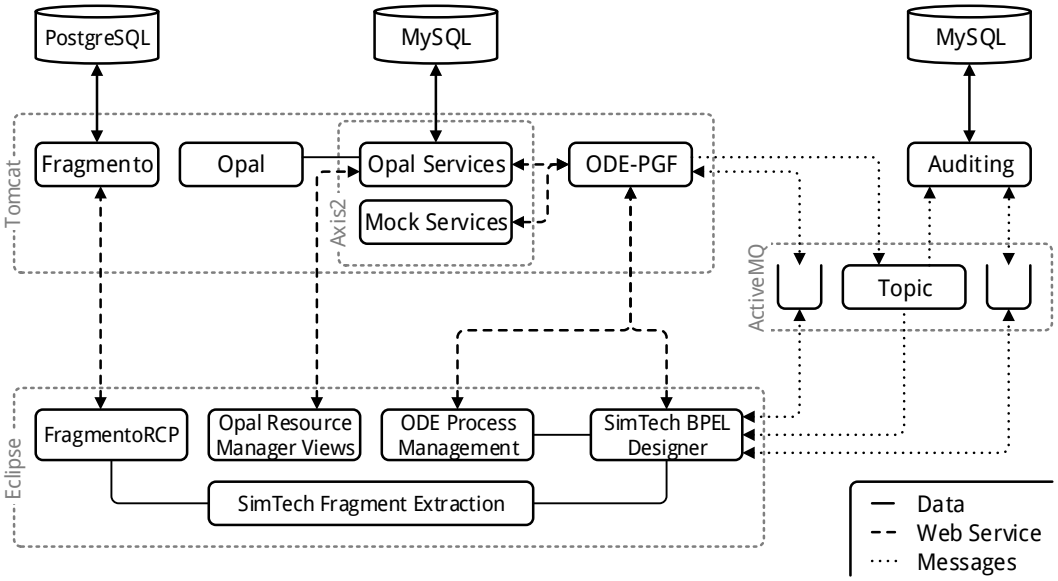

Figure 2.3: Architecture of the SimTech Workflow Management System.

Once a workflow is ready for execution, it can be deployed on a workflow engine. The workflow engine, in this case, is the ODE Pluggable Framework (ODE-PGF)[14] component running on the application server. To deploy, undeploy, and otherwise manage a workflow on the workflow engine, the SimTech BPEL Designer calls a Web Service provided by ODE-PGF. Additionally, the SimTech BPEL Designer uses information from the ODE Process Management component in the SimTech Modeler to handle duplicate workflow process instances. Ode Process Management also uses the Web Service provided by ODE-PGF to manage workflows on the workflow engine.

During the execution of a workflow on the workflow engine, ODE-PGF reads messages from a queue on the messaging middleware. The SimTech BPEL Designer can put messages in this queue to access various functions of ODE-PGF, such as setting variable values or breakpoints. ODE-PGF also emits events under a topic published on the messaging middleware, as shown on the right side of Figure 2.3. Other components can subscribe to this

[12] http://www.iaas.uni-stuttgart.de/forschung/projects/fragmento/start.htm
[13] http://www.postgresql.org/
[14] http://www.iaas.uni-stuttgart.de/forschung/projects/ODE-PGF/

topic and act upon these events. In this case, the SimTech BPEL Designer, as well as the SimTech Auditing Application subscribe to these events.

The SimTech Auditing Application is a Java application that is provided as *.jar* file. It persistently saves each event that is published by ODE-PGF on this topic in a MySQL[15] database. SimTech Auditing also communicates with the SimTech Modeler using a queue on the messaging middleware. When a process instance is opened in the SimTech Modeler, data saved by Auditing on this instance is shared via this queue. This allows a user to switch between multiple process instances inside the SimTech Modeler.

Part of the execution of simulation workflows on the workflow engine is the invocation of external service during particular workflow steps. Two such services are hosted by Apache Axis2[16], a Web Service engine running on the application server, as seen in the top middle of Figure 2.3. The Mock Services offer functionality for testing purposes. The Opal Services enables communication with the Opal simulation software, which is a solid state body simulation software that is based on workflows. The Opal Services allows workflows to incorporate Opal simulations and can additionally store and retrieve simulation data in a MySQL database. The Opal Resource Manger Views in the SimTech Modeler also use the Opal Web Services to retrieve information from Opal.

[15] http://www.mysql.com/
[16] http://axis.apache.org/axis2/java/core/

3 Cloud Computing

Cloud computing is a combination of technologies and business models that enables a more flexible use of IT resources compared to traditional IT. In the following sections definitions and essential characteristics of cloud computing will be examined in more detail.

3.1 Definition

Vaquero et al. studied over 20 definitions and concluded that a minimum definition would contain scalability, pay-per-use utility model, and virtualization [44]. They then propose the following definition:

> Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs[1]. [44]

The National Institute of Standards and Technology (NIST) also proposes a definition:

> Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [30]

It further proposes that "this cloud model is composed of five essential characteristics, three service models, and four deployment models" [30], which will be examined in more detail in the following sections.

[1] Service Level Agreement (SLA): "An agreement that sets the expectations between the service provider and the customer and describes the products or services to be delivered, the single point of contact for end-user problems and the metrics by which the effectiveness of the process is monitored and approved." [40]

3.2 Actors

Vaquero et al. distinguish between three different actors involved in cloud computing. First, there are the infrastructure providers, who offer the infrastructure necessary for cloud computing as a service to others. Service providers use these offerings to create their own software services in the cloud, while reducing costs and increasing flexibility. They, in turn, offer their software services to service users through the internet [44]. Behrendt et al. also defines three very similar roles, but call them cloud service provider, cloud service creator, and cloud service consumer [6].

These roles are not mutual exclusive, i.e., an infrastructure provider might also act as a service provider and create and sell services on his own infrastructure. A service provider that uses the services of an infrastructure provider can also be seen as a service user from the perspective of the infrastructure provider. An infrastructure or service provider could also consume his own services as a service user.

3.3 Essential Characteristics

In their definition of cloud computing, NIST identifies five essential characteristics: On-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [30]. On-demand self-service allows service providers and service users to provision (i.e., to setup, configure, manage, shutdown, etc. [25]) cloud resources (e.g., server time, storage, bandwidth, etc.) as needed, even automatically, without human interaction with the infrastructure and/or service providers [30]. Broad network access means that cloud services are accessible over networks using standard mechanisms, thus allowing service users to use the services with many different devices, such as mobile phones, tables, laptops, and workstation [30].

Resource pooling (or resource sharing [44]) describes the multi-tenant model used by infrastructure providers, where multiple service providers (and thus also service users) share resources in a dynamic way. This usually is non-transparent to the service providers, who have no control over the physical location of the resources and perceive them as a single dedicated entity [30, 44]. Infrastructure providers do however sometimes give the option to choose between locations at a high level, for example Amazon Web Service (AWS), who offers different regions around the world (e.g., in the US, EU, Asia, etc.), and inside these regions different availability zones, which are separated from each other to insulate against failure [3, pp. 114-115].

Rapid elasticity enables service providers to add or remove cloud resources automatically in response to various situations, such as increase or decrease in demand, seasonal traffic spikes, failures and outages, etc [30]. It also facilitates one-time usage scenarios, where cloud resources can be used for a particular task that is not reoccurring. Here, rapid elasticity

removes the need for long time planning, since a virtually unlimited amount of resources can be used at any time [4]. Measured service means that cloud systems can control and optimize resources based on measurements, such as used bandwidth, processor load, active users, etc. This also provides transparency about actual usage to both infrastructure and service providers, and service users [30].

3.4 Cloud Service Models

The increasing number of services offered in the cloud are collected under the umbrella of Anything as a Service (XaaS) or Everything as a Service (EaaS). Of these services, the three most widely known are Infrastructure as a Service, Platform at a Service, and Software as a Service.

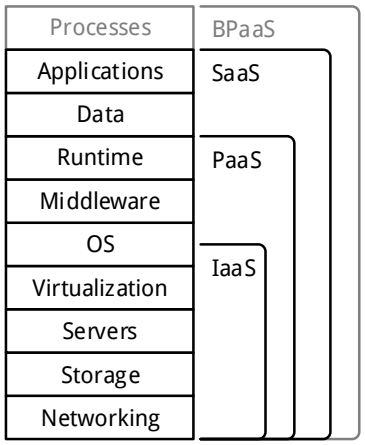

Figure 3.1: Cloud service models landscape [based on 14].

Infrastructure as a Service (IaaS) provides service users with the hardware infrastructure for cloud services. Here, the service and infrastructure providers only control the network, storage, servers, and virtualization environment, as can be seen in Figure 3.1. The service users can install, run, and manage arbitrary operating systems and software using this infrastructure [30]. Lenk et al. further divide IaaS into the resource set and infrastructure services. The resource set includes both physical and virtual resources, which both provide management functionality for higher level services to allow them to automate setup, scaling, tear-down, and fail-over of these resources. Infrastructure services encompass basic computational, storage, and networking infrastructure services, as well as higher infrastructure services [27].

Figure 3.1 shows that Platform as a Service (PaaS) offers the service users a cloud infrastructure, encompassing network, servers, operating systems, storage, and pre-installed mid-

dleware or runtime systems, which are controlled by the service and infrastructure providers. The service users can build on this platform, using the tools provided by the service provider (or by others), and can control, to some degree, the deployed applications and configuration settings of the environment [30]. Lenk et al. further split PaaS into programming environments and execution environments, where the former offers tools and framework to enable and support development for a particular platform, while the latter then allows for the execution of the code on this platform [27].

Software as a Service (SaaS) is in essence software that is accessible over the internet via web browsers or a program interface. Except for limited user configurable application settings, the service users do not manage or control any part of the SaaS offering. This is entirely the domain of the service and infrastructure providers, as shown in Figure 3.1 [30]. According to Lenk et al., SaaS can be further divided into applications and application services. Application services are mainly used as high level building blocks for applications [27].

In additions to these three main service models, literature describes many other service models. Business-Process-as-a-Service (BPaaS) is one example that is also shown in Figure 3.1, which is a layer on top of SaaS. Behrendt et al. describe it as "any business process (horizontal or vertical) delivered through the Cloud service model [...]" [6]. In this case, the provider of the BPaaS offering is responsible for the business functions related to the service. As an example, Behrendt et al. give software testing, where the testing process would be provided by a BPaaS provider, who would be responsible for the testing staff [6].

Lenk et al. mention Humans as a Service (HuaaS), also a layer on top of SaaS, where crowds of people are used to provide services that require human intelligence. An example is Amazon Mechanical Turk[2] [27]. Dawoud, Takouna, and Meinel add two more service models, Storage as a Service and Components as a Service (CaaS). Storage as a Service, on the same level as PaaS, provides storage and databases. CaaS, sitting in between PaaS and SaaS, offers Service Oriented Architecture (SOA) via Web Service (WS) standards [11]. Youseff, Butrico, and Silva add Data-Storage as a Service (DaaS) and Communication as a Service besides IaaS [47].

3.5 Cloud Deployment Models

Deployment models describe, how a cloud infrastructure is used. NIST distinguishes between four different deployment models in cloud computing: Private cloud, community cloud, public cloud, and hybrid cloud [30]. They differ in how exclusive the service users have access to the resources provided by the cloud [29].

A private cloud is a cloud infrastructure that can only be used by a single organization. However, ownership, management, operation, and location of the cloud infrastructure are not necessarily exclusive to this organization [30]. For example, the cloud infrastructure

[2] https://www.mturk.com/mturk/

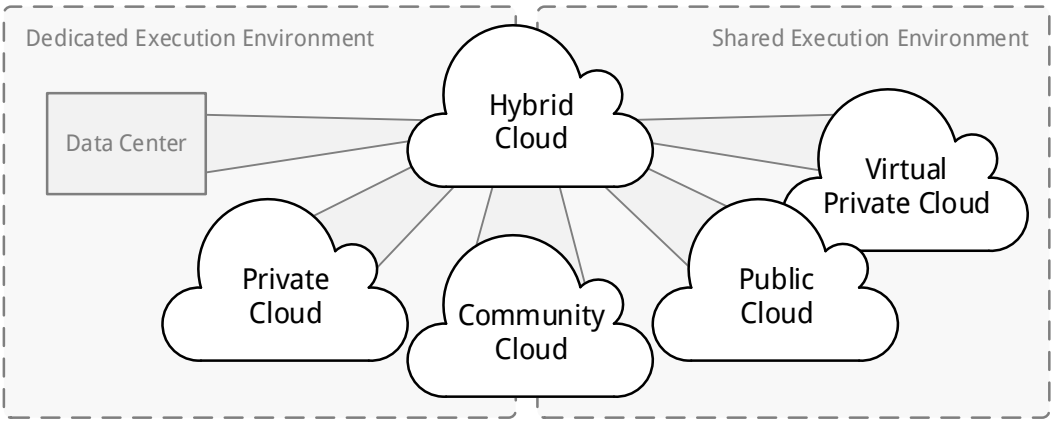

Figure 3.2: Cloud deployment models [based on 14].

could be owned, managed, and operated by a third party, and the location of the physical infrastructure could be on premise, or off premise (i.e., on-site private cloud, or outsourced private cloud) [29]. Private clouds offer a dedicated environment similar to traditional data centers. Limited access and often close proximity to users can affect the elasticity of private clouds, if the user group is to small or to not diverse enough [14]

A community cloud is very similar to a private cloud, but instead of only being used by a single organization, it is used by a community with shared concerns, e.g., mission objective, policies, etc. Ownership, management, operation, and location is not exclusive to this community [30]. The cloud infrastructure may be managed by a third party, and the location could be on, or off premise (i.e., on-site community cloud, or outsourced community cloud) [29].

A public cloud can be used by the general public. It is owned, managed, and operated by any combination of businesses, and academic or government organizations, and exist on the premise of the cloud provider [30]. A large and diverse user group leads to better economies of scale compared to other cloud types, which also makes public clouds cheaper. Parts of the public cloud can be separated to only allow access to certain users, creating a virtual private cloud. Compared to a real private cloud, the separation is only virtual, not physical [14].

A hybrid cloud is a combination of two or more of the aforementioned deployment models, as well as traditional data centers, in which these deployment models remain unique, but are able to interchange data or applications [30]. This type of cloud can be interesting for businesses, where different parts of their infrastructure require different types of security or privacy. A hybrid cloud allows them to use a private cloud for critical infrastructure, augment it, if necessary, with public cloud services, and integrate legacy applications from traditional data centers [14].

3.6 Cloud Offerings

Fehling et al. describe three different types of services offered in the cloud, namely cloud compute services, cloud storage services, and cloud communication services [16, also see 17, 15].

Cloud compute services execute work in the cloud, i.e., they offer real and virtual hardware resources that can be used to run any software. These services are based on an elastic infrastructure, which allows resources to be provisioned dynamically, depending on the current workload. Furthermore, cloud compute service can be subdivided based on availability. Low availability computing nodes provide cheaper cloud compute services when availability is not a critical factor. This is often accomplished by using a large number of commodity servers. High availability computing nodes, on the other hand, offer cloud compute services where high availability and performance is important. This can be achieved with dedicated hardware that offers redundant internal components and self-healing capabilities [16].

Cloud storage services provide several different options to store data in the cloud. Relational data stores are used to store data elements and their relations, which allows to query the database for a wide selection of different subset of this data. Blob storage can store large data elements in a folder hierarchy. Data can later be retrieved with a unique name using traditional technologies such as FTP or HTTP. Block storage allows servers to access a high available storage like a local hard drive. This allows these servers to fail and simply be replaces, without loosing any data saved in block storage. NoSQL Storage offers a way to store data that is highly scalable and flexible to changes of the data structure. Compared to relational data stores it has very limited querying capabilities, which allows it to be widely distributed but also adds additional complexity to applications that use NoSQL Storage. Cloud storage service also have to handle varying degrees of consistency, i.e., how to keep data that is replicated multiple times for fault tolerance synchronized. Strict consistency, i.e., all replicas are always up to date, offers lower availability, while eventual consistency trades in consistency for increased availability and performance [16].

Cloud communication services exchange information between different cloud and non-cloud resources. Message-oriented middleware enables remote communication via messages for loosely coupled components. It acts as an intermediary that handles the conversion between different languages, formats and technologies, as well as message addressing and routing. Cloud communication services also provide ways to do reliable messaging, i.e., ensure that messages are not lost and can be recovered after a failure. Additionally they provide facilities to send messages exactly once (exactly-once delivery) or at least once (at-least-once delivery) [16].

3.7 Cloud Application Architecture

Fehling et al. describe multiple architectural patterns for cloud applications. They divide these patterns in four groups: Basic architectural patterns, elasticity patterns, availability patterns, and multi-tenancy patterns [16, also see 17, 15].

Basic architectural patterns describe the fundamental structure of cloud applications. In a cloud environment, a composite application, i.e., an application that is composed out of independent components, is desirable. This makes the application flexible, since components can easily be changed, and functionality can be scaled-out individually or even offered by different providers. For this to work, loose coupling, i.e., that component make few assumptions about each other, is important. Because of the distributed nature of cloud computing, components should also be stateless. Instead of internal state, components rely on external state in persistent storage, so if a component fails or is added or removed, it can recover its state at any time. Finally, components must handle duplicate messages if the message middleware does not guarantee exactly-once delivery. For this, idempotent components should be used, i.e., components that can handle multiple identical messages safely [16].

Elasticity patterns describe how cloud applications can adjust their size automatically, depending on the current workload. Elastic Components can be added or removed automatically, utilizing the system monitoring and provisioning functionality of the elastic infrastructure. Elastic load balancers can use the number of requests to determine the amount of required resources and automatically provision them. An elastic queue can be used to adjust the number of component based on the amount of requests it handles. Finally, map reduce can be used to break up large data sets returned by NoSQL storage solutions into smaller sets. These smaller sets can be distributed to multiple compute nodes that now can execute a complex query on them. The results of of these queries can then be merged to obtain the end result [16].

Availability patterns specify how different kinds of availability can be achieved in a cloud environment, where the availability of single component often is not high. One solution is to use redundant nodes that communicate with reliable messaging over message oriented middleware, coupled with a watchdog, a monitor component that replaces failed nodes. To update compute nodes in a high availability environment, update transitioning is used. An additional compute node with updated software is started and can be tested in parallel with older nodes. Once proper functionality is verified, workload can be fully transitioned to the updated node and the old node can be shut down [16].

Multi-tenancy patterns describe how cloud resources can be shared by multiple users. If individual configuration of a component is not required, a single instance component can be used for all tenants, thereby utilizing resources more efficiently. If individual configuration is required, a single configurable instance component might be used. Here, a single instance component is provided for all tenant, which adjusts its functionality based on individual configurations. Sometimes, component sharing is unfeasible, for example, because

of law restriction, performance problems, or greatly differing configuration requirements. In this case, multi-instance components are deployed, providing each tenant with his own individual component, but at the cost of lower efficiency and hence, higher costs [16].

4 Migration

As technology continuously develops, older systems sometimes have to be migrated into a newer environment. While often involving high costs and effort, such a migration also provides possibilities to improve the existing systems. The following sections provide a brief overview of migration and what it entails.

4.1 Overview

In nature, groups of animals and people regularly move to different areas that offer improved conditions to them. This process is called migration [31]. Similarly, software has to be moved to new environments once in a while to be able to profit from new technological developments. This allows the developers to extend its service life during the evolution phase of the software life cycle, before it becomes technically or economically too difficult to continue to support the software [35]. Thus, migration describes the transfer of a software system to a new environment or a new design, without changing its functionality. After a successful migration, new technological possibilities can then be used to further extend the software system [2].

Gimnich and Winter further divide migration into four subcategories. Hardware migration is the move from one underlying hardware environment to another. Runtime migration describes changing the underlying system software, such as operating systems or databases. Fundamental changes in the systems structure are an architecture migration, for example, the change from a monolithic system to a multi-tier architecture. Finally, a migration of the development environment changes, for example, the programming language [21].

4.2 Approach

Over the years numerous migration process models have been specified in the literature. Brodie and Stonebraker describe two processes, namely Cold Turkey and Chicken Little. Cold Turkey involves a complete rewrite of the system to be migrated, whereas Chicken Little migrates the system in small incremental steps [7]. Wu et al. add the Butterfly Methodology, where legacy and target system don't need to be operated in parallel during migration [46]. Battaglia, Savoia, and Favaro developed the Renaissance Method, which is also a phased

approach to software migration [5]. Ackermann created a reference process for software migration from these and other process models [1, also see 2]. What follows is a closer description of this process, which provides a general overview of software migration activities.

The reference process model describes the essential activities involved in a migration and divides them in supporting and migration specific core areas. Supporting core areas include project management, configuration and change management, development environments, etc. Migration specific core areas are requirement analysis, legacy analysis, target or bridge design, strategy selection, implementation, quality control, and handover [2].

Starting with the requirement analysis, all requirements for a successful migration are collected. Additionally, functional requirements relevant to the migration are collected during the legacy analysis phase. Also during this phase, reuse of existing components is considered and a reusable, decomposable structure is created. Based on the results of the legacy analysis, a migration strategy is selected. This strategy determines for every component if it should be migrated either by conversion, wrapping, or redevelopment. This also determines a strategy for the handover during the last phase, which could be done incremental or in one big step ("Big-Bang") [2].

Next, a target design is created, which describes the target system in its new target environment. When a parallel operation of legacy and target system is necessary during development, a bridge design that describes this temporary architecture is also necessary. The actual transformation of the legacy system to the target system is done during the implementation phase. It includes the development of transformation tools, wrappers, gateways, and total redevelopment. The result of the implementation phase should be a target system in the target environment that is functionally equivalent to the legacy system. This is tested during the quality control phase, for example, with regression tests, before the final results are handed over. These steps do not have to be executed in a strictly linear order. They often influence each other and have to be revisited to reach a satisfactory end result [2].

4.3 Cloud Migration

Cloud migration is not necessarily different from any other software migration. Depending on the changes made during the migration it fits nicely into one or more of the subcategories established by Gimnich and Winter. There are however certain aspects that need special attention when migrating into a cloud environment, for example licensing and compliance.

Armbrust et al. identified software licensing as one of the ten biggest obstacles for adoption and growth of cloud computing. The reason for this is that software licenses commonly restrict the computers on which the software can run and are not designed for geographically distributed usage [4, 10]. Dalheimer and Pfreundt present a license management framework (GenLM) that allows software companies to provide on-demand licensing, which better suits the cloud model [10]. It is, however, in the hands of the individual software companies to offer

licensing options that fit the cloud computing approach [4, p. 19]. Therefore, when migrating an existing software system to the cloud, one has to make sure that the license of every component allows this, especially when using commercial software.

Armbrust et al. also mention compliance with national laws as an obstacle to cloud migration [4, pp. 15-16]. For example, many nations require that certain data, e.g. health care data, must be kept within national boundaries for privacy and security reasons. If software that uses such data is to be migrated into a cloud environment, precautions have to be made to meet such regulations. While some cloud provider, like Amazon, support a more granular control of the geographic location of data and compute instances, this is not necessarily the case for all cloud providers. Control also might not be granular enough to meet the required standards in particular situations. Legal requirements therefore could hinder a cloud migration and have to be considered.

It can also be quite challenging to find a cloud provider among the many available options. Li et al. found that offered services vary widely in performance and cost. They present CloudCMP, a system that aims to help potential customers to decide on a provider by comparing the performance and cost of cloud providers through various metrics [28]. Khajeh-Hosseini et al. also describe a cost modeling tool to help decision making during migration into public IaaS clouds[1]. It uses models of application, data, and infrastructure requirements, as well as expected computational resource usage patterns to calculate cost estimates for various public IaaS clouds, deployment options, and usage scenarios [24].

[1] Now part of RightScale, Inc.: `http://www.planforcloud.com/`

5 Assessing the Migratability of the SimTech Prototype

This chapter assesses which component of the SimTech SWfMS can and should be migrated to the cloud. Different cloud providers and service models are examined. Then, the components are grouped based on their requirements.

5.1 Which Components Should be Migrated to the Cloud?

The underlying architecture of the SimTech SWfMS is already a composite architecture. All components are loosely coupled by using Web Services or ActiveMQ as messaging middleware. This meets the desired architecture pattern for cloud applications described in Section 3.7, so from an architectural standpoint, a cloud migration is therefore possible.

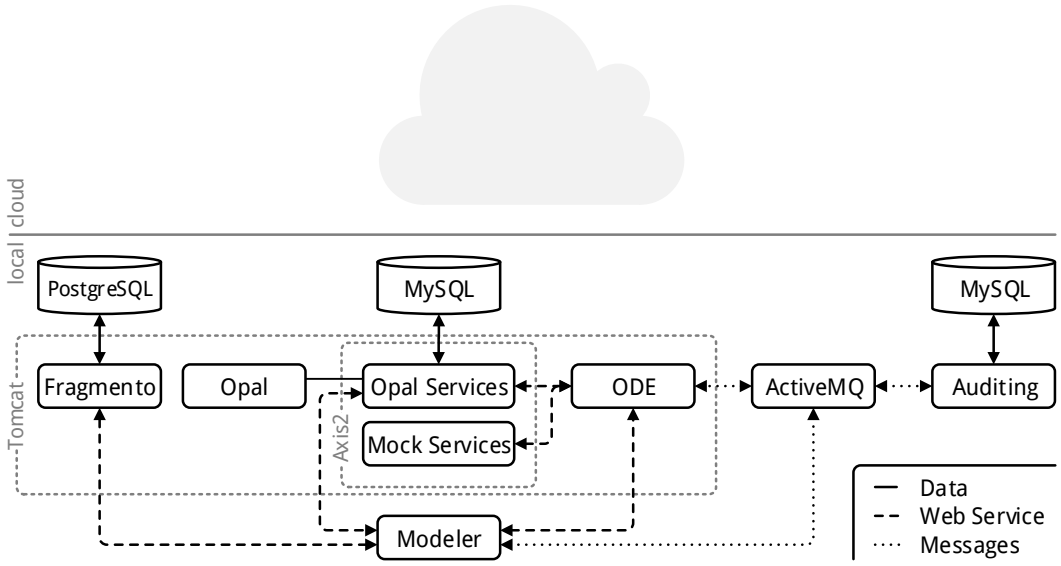

Figure 5.1: Components and their communication dependencies.

There are also no commercial components used that could be difficult to migrate due to licensing issues. Table 5.1 shows that most components are licensed as open source. Opal, Opal Services, Mock Services, and Auditing are developed at the University of Stuttgart but are not publicly available.

Component	License	Open Source
ActiveMQ	Apache License, Version 2.0[2]	✓
Auditing	-	✗
Eclipse	Eclipse Public License, Version 1.0 (EPL-1.0)[1]	✓
Fragmento	Apache License, Version 2.0[2]	✓
Mock Services	-	✗
MySQL	GPL[3] (depending on usage and distribution[4])	✓
ODE	Apache License, Version 2.0[2]	✓
Opal	-	✗
Opal Services	-	✗
PostgreSQL	The PostgreSQL License (PostgreSQL)[5]	✓

Table 5.1: Component licenses.

Now, each component is examined in closer detail to see, where it would be technically possible to migrate the component to the cloud.

5.1.1 SimTech Modeler

As can be seen in Figure 5.2, the SimTech Modeler communicates with Fragmento, the Opal Services, as well as ODE over Web Service calls and returns. Additionally, the SimTech Modeler communicates with ActiveMQ via Messages. A migration into the cloud would therefore be possible from a communication standpoint.

However, installing SimTech Modeler on a server in the cloud and accessing it remotely without modification is not something it was designed for, because it is based on Eclipse. While the Eclipse Foundation is also developing an open source platform for cloud based development, called Orion [33], it is currently aimed towards web development and does not support standard Eclipse plugins. Eclipse can be run headless, but only to compile existing projects. The Eclipse Web Interface project aims to make Eclipse commands invokable via a web browser, but development seems to have stopped [13]. The Remote Application Platform (RAP), developed by the Eclipse Foundation, provides functionality to run Java applications and Eclipse plugins in a web browser [36]. A proof of concept shows the BPEL

[1] http://opensource.org/licenses/eclipse-1.0.php
[2] http://opensource.org/licenses/Apache-2.0
[3] http://opensource.org/licenses/GPL-3.0
[4] http://www.mysql.com/about/legal/licensing/index.html
[5] http://opensource.org/licenses/postgresql

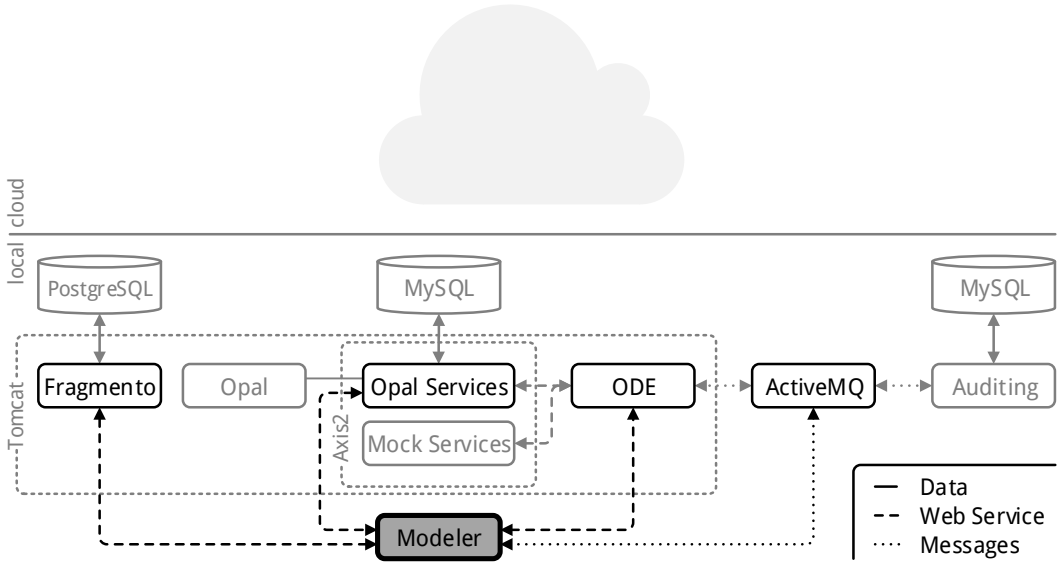

Figure 5.2: Dependencies of the SimTech Modeler.

designer running as RAP application [41], which could make RAP a viable option, if enough development effort is made.

Another option would be to replace the SimTech Modeler with an existing cloud BPEL editor. This is also problematic, because the SimTech Prototype uses a heavily modified BPEL editor. No existing editor offers this modified functionality.

The last option is to develop a custom SaaS solution that provides a workflow editor that is able to create compatible workflows, as well as manage and execute them on the existing back-end. While certainly possible with modern web standards, such as HTML5 and JavaScript, this entails a substantial development effort. For the reasons expressed here, the best choice for the remainder of this thesis is to not migrate the SimTech Modeler into the cloud.

5.1.2 Fragmento

Figure 5.3 shows that Fragmento has only two dependencies. On the bottom, it communicates with its counterpart, FragmentoRCP, inside the SimTech Modeler. Here, Web Service calls and returns are used exclusively. On the top, Fragmento saves all data in a PostgreSQL database. Other than that, Fragmento is separate from all other components, which makes it easily detachable.

There are several reasons that suggest to move Fragmento into the cloud. Since it is used to store and retrieve workflow fragments, using Fragmento to share such fragments

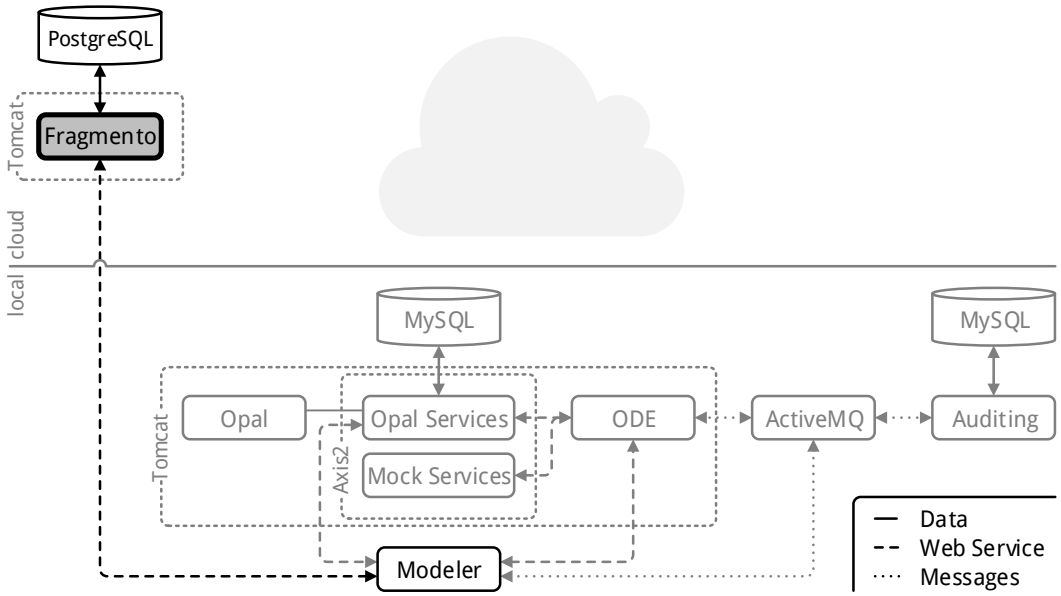

Figure 5.3: Dependencies of the Fragmento component.

between users is a possible usage scenario, which is enhanced by bringing Fragmento into the cloud. Users no longer have to manage local installations of Fragmento and in the cloud it can be automatically scaled with changing demand. In this case, it also makes sense to separate Fragmento from other components, such as Opal, ODE, or Axis2. The Fragmento cloud instance could then be run non-stop to facilitate collaboration at any time. All other components however are only needed when actually executing a workflow and can therefore be shut down when not used.

As shown earlier, the SimTech Modeler cannot be moved into the cloud or easily be replaced, so migrating it together with Fragmento is not an option at this time. Migrating only Fragmento to the cloud and keeping PostgreSQL local would mean that the data communication between Fragmento and PostgreSQL would leave the cloud, which is expensive and slower then cloud internal data transfer. It would also negate the shareability benefits mentioned earlier, since the data would no longer be available in the cloud. Therefore it makes sense to migrate Fragmento and PostgreSQL together.

5.1.3 Opal

Opal only communicates with the Opal Web Service, which in turn handles data storage into a MySQL database and Web Service communication with Eclipse and ODE, as can be seen in Figure 5.4. The Opal Web Service is tightly coupled to Opal, since it calls the Opal executables directly. Opal and the Opal Web Service can therefore not be separated and must migrate

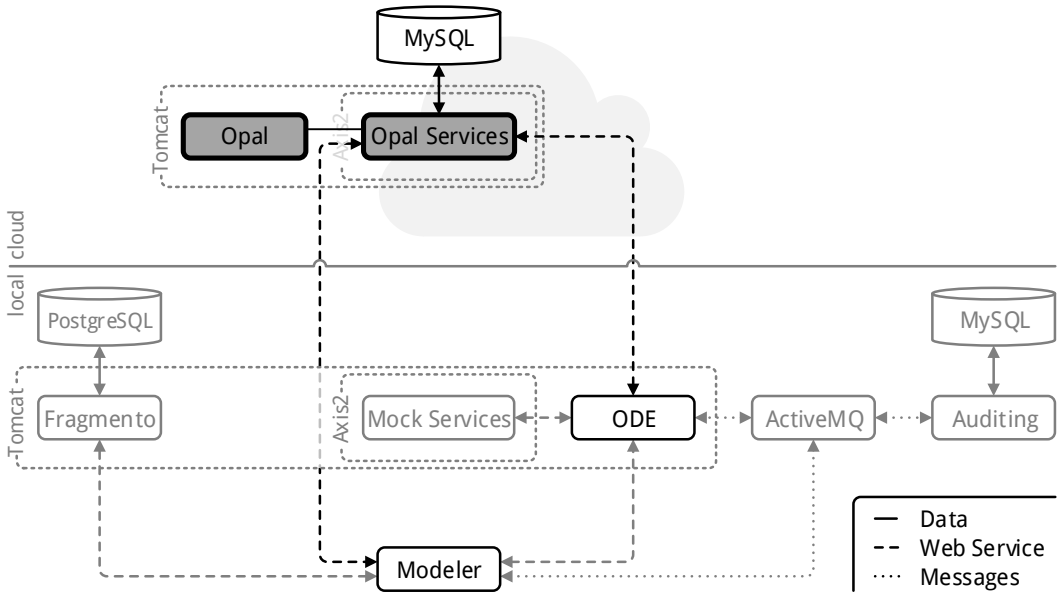

Figure 5.4: Dependencies of the Opal component.

together. The MySQL database should also stay close to the Opal Web Service to avoid increased costs and latency. Separation from the SimTech Modeler and ODE is however possible, since communication with these components is via Web Service calls and returns only. Migrating Opal into the cloud is therefore possible, if done together with the Opal Web Service and MySQL.

When migrating Opal to the cloud, it also makes sense to separate it from other components like, for example, ODE. Then it would be possible to run multiple Opal instances in parallel and scaling them with demand. Opal instances also only have to run, if a simulation needs to be executed by a workflow, and can be shut down otherwise, saving resources.

5.1.4 Mock Services

The Mock Services, as displayed in Figure 5.5 can be easily moved to the cloud. Since they act as a dummy for testing, they don't have any other dependencies besides the communication with ODE.

5.1.5 ODE

Figure 5.6 shows that ODE communicates with the SimTech Modeler, Opal Services, and Mock Services via Web Service Calls and returns, and with ActiveMQ via messages. It is therefore possible to separate ODE from the other components and migrate it into the cloud.

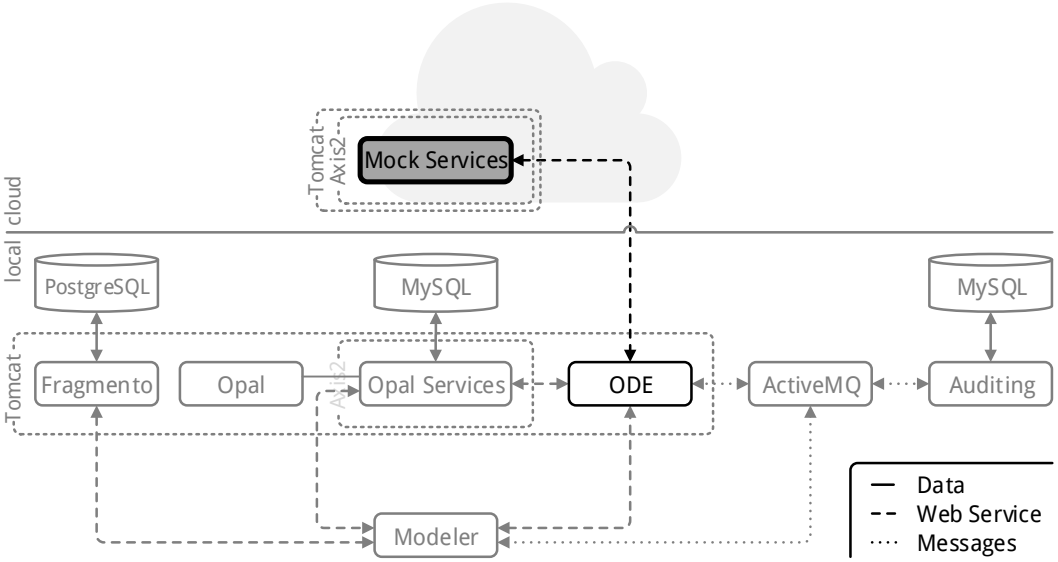

Figure 5.5: Dependencies of the Mock Services component.

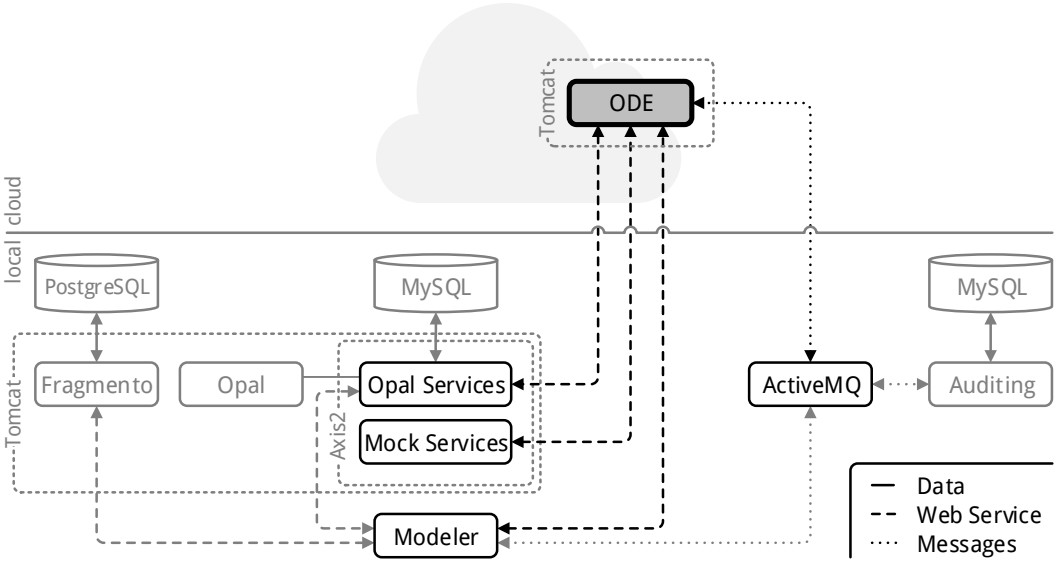

Figure 5.6: Dependencies of the ODE component.

It is also advisable to keep ODE separate from other components, especially the Web Services. If they would be running together with ODE in one Tomcat instance, a crashed or highly resource intensive Web Service could have an impact on ODE and therefore the execution of the whole workflow. Loosely coupling the Web Services to ODE would possibly allow the workflow engine to recover failures and resource intensive applications would not impact the whole workflow.

5.1.6 ActiveMQ

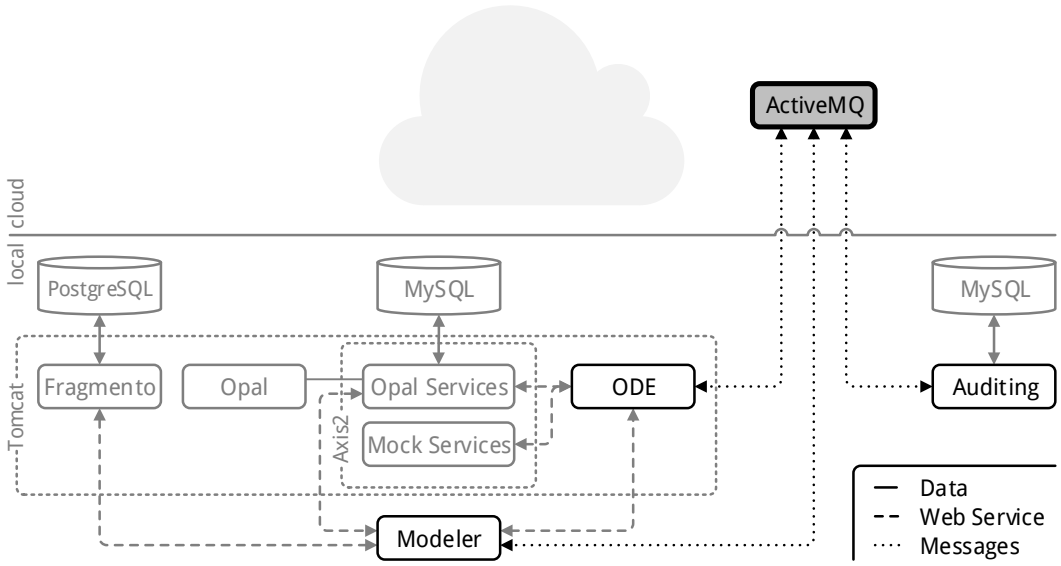

Figure 5.7: Dependencies of the ActiveMQ component.

Figure 5.7 shows that ActiveMQ communicates with the SimTech Modeler, ODE and the SimTech Auditing application via messages. Since ActiveMQ is a message oriented middleware that enables communication between loosely coupled services, it's naturally easy to separate it from these service. Moving ActiveMQ into the cloud would therefore be possible.

5.1.7 Auditing

As can be seen in Figure 5.8, the SimTech Auditing application communicates with ActiveMQ via messaging and uses a MySQL database to store data. Auditing therefore can be easily separated from ActiveMQ, but should be migrated together with its database, to avoid higher cost and latency, as described earlier.

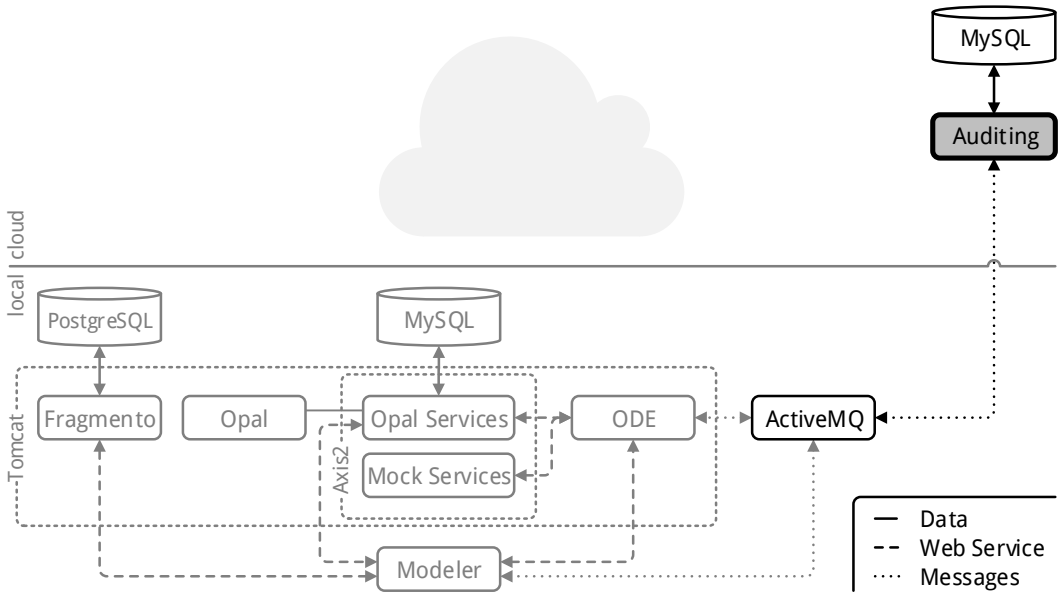

Figure 5.8: Dependencies of the Auditing component.

There are however some problems that make a cloud migration of the Auditing application difficult. Auditing is distributed as a simple Java desktop application, packaged as *.jar* file. It is written in such a way that it requires a Graphical User Interface (GUI) to work as intended. If the GUI can't be displayed, for example, if the X-Server on a Unix system is not present, Auditing will fail to start. Since servers are commonly run without window systems, for security and performance reasons, Auditing will not be able to run without either modifying Auditing or the server.

Modifying the server to run with a GUI would allow the execution of the Auditing application. Doing so would however decrease performance and security of the server. Since using Auditing without a GUI is not possible, the GUI must also be accessible remotely, for example via X-Forwarding or Virtual Network Computing (VNC). This introduces an additional decrease in performance and security.

Modifying Auditing to run without a GUI would make a new control mechanism necessary, since users still have to configure and manage the application. This could be done, for example, with a Web Service that ties Auditing into the the SimTech Modeler GUI or a web interface that allow control from a web browser. This however requires some development effort. For these reasons, Auditing will remain local for the remainder of this thesis.

5.1.8 Conclusion

The SimTech Prototype uses loosely coupled open source components, which makes cloud migration a possibility from an architectural and a licensing standpoint. On further examination of the components it becomes evident that all but two of them can be easily migrated to the cloud. The SimTech Modeler would have to be replace or heavily modified and SimTech Auditing would have to be converted into a Web Service. For both components, this would require some effort, so for this thesis they will remain local. This does however not change the usefulness of migrating the other components to a cloud environment.

5.2 Choosing the Right Service Model

The previous section established that Fragmento, Opal, Opal Services, Mock Services, ODE, ActiveMQ, MySQL, and PostgreSQL can be migrated into the cloud, while the SimTech Modeler and Auditing can't, at least without significant effort. This section will look at the different service models available and which of them is appropriate for which component.

In general, a PaaS offering is preferable to IaaS. It allows fine grained scalability and billing based on actual usage, while reducing management to the application and data. However, existing PaaS offerings might not support all requirements of the components. In those cases, using IaaS will become necessary. Then, management of the OS, middleware, and runtime layers, as well as scalability, will increase cost and complexity.

5.2.1 Databases

Table 5.2 shows an overview of the requirements as well as how well a selection of cloud providers meets these requirements with their PaaS offerings. Support for databases as PaaS is in general good. The first row shows that MySQL is supported by every cloud provider as a PaaS solution, except by Windows Azure[14], which only offers SQL Server databases. Support for PostgreSQL is not quite so strong, but still good, as can be seen in row two. Modifying Fragmento to use MySQL instead of PostgreSQL could be considered to practically eliminate database support as a selection criteria for a cloud provider. It is, however, not strictly necessary.

5.2.2 Application Servers

An Application Server is required to run Fragmento, Opal, and ODE, as well as Axis2, which hosts the Web Services. Apache Tomcat, which is currently used in the local installation, is supported by some cloud providers as a PaaS solution, as shown in row three and four of Table 5.2. There are other application servers available that possibly could support the previously mentioned components. CloudBees[7] and OpenShift[12] offer JBoss[15], and Google

App Engine[9] and Jelastic[11] support Jetty[16]. Jelastic also offers GlassFish[17]. But, since support for these application servers is similarly spotty, making the effort to change application servers doesn't seem beneficial.

		Cloud Provider								
		Amazon Web Services[6]	CloudBees[7]	Cloud Foundry[8]	Google App Egnine[9]	Heroku[10]	Jelastic[11]	OpenShift[12]	Rackspace[13]	Windows Azure[14]
Databases	MySQL	✓	✓	✓	✓	✓	✓	✓	✓	✗
	PostgreSQL	✗	✓	✓	✗	✓	✓	✓	✗	✗
Application Servers	Tomcat6	✓	✓	✓	✗	✗	✓	✓	✗	✗
	Tomcat7	✓	✗	✗	✗	✓	✓	✗	✗	✗
	other	✗	✓[15]	✗	✓[16]	✗	✓[16,17]	✓[15]	✗	✗
Messaging	ActiveMQ	✗	✗	✗	✗	✗	✗	✗	✗	✗
	JMS compl.	✗	✗	✗	✗	✗	✗	✗	✗	✗
	other	✓[18]	✗	✓[19]	✗	✓[19,20]	✗	✗	✗	✗

Table 5.2: Provider PaaS fulfillment of requirements.

5.2.3 Messaging Middleware

The SimTech Prototype uses ActiveMQ as messaging middleware. Unfortunately, row five of Table 5.2 shows that ActiveMQ as PaaS is not offered by any of the cloud providers considered here. An alternative messaging middleware that is Java Message Service (JMS)[21] compliant is equally non-existent as PaaS solution, as seen in row six. There are some cloud

[6] `http://aws.amazon.com/`
[7] `http://www.cloudbees.com/`
[8] `http://www.cloudfoundry.com/`
[9] `https://developers.google.com/appengine/`
[10] `https://www.heroku.com/`
[11] `http://jelastic.com/`
[12] `https://www.openshift.com/`
[13] `http://www.rackspace.com/`
[14] `http://www.windowsazure.com/`
[15] JBoss `http://www.jboss.org/`
[16] Jetty `http://www.eclipse.org/jetty/`
[17] GlassFish `https://glassfish.java.net/`
[18] AWS SQS `http://aws.amazon.com/sqs/`, AWS SNS `http://aws.amazon.com/sns/`
[19] RabbitMQ `http://www.rabbitmq.com/`
[20] IronMQ `http://www.iron.io/mq`
[21] `http://www.oracle.com/technetwork/java/index-jsp-142945.html`

providers that support other messaging middleware with their PaaS offerings, as evident from row seven. AWS[6] offers Amazon Simple Queue Service (Amazon SQS) and Amazon Simple Notification Service (Amazon SNS)[18]. A possible scenario to be considered would be to use the Nevado JMS driver[22], a connector that allows JMS conform messaging over Amazon SQS and SNS. Nevado JMS currently supports 83% of the JMS 1.1 specification. Another scenario to consider would be to replace ActiveMQ with a different messaging middleware that supports queues and topics and is available as PaaS. Both Heroku and Cloud Foundry[8] offer RabbitMQ[19] as PaaS solution. Heroku also offers IronMQ[20]. Because of the complexity involved in these possible solutions, for this thesis, ActiveMQ will be provided from IaaS.

5.2.4 Conclusion

A PaaS only cloud environment for the SimTech Prototype is not possible at the moment without modifications. PaaS solutions for the databases and application servers can be found, but PaaS support for JMS compliant messaging middleware does not exist. To reach a PaaS only environment, ActiveMQ would have to be replaced. Regarding cloud providers, Google App Engine, Rackspace, and Windows Azure don't seem to be good choices for this migration. Overall, all components except ActiveMQ could be migrated to PaaS solutions.

5.3 Component Grouping

The two previous section have established that Fragmento, Opal, Opal Services, Mock Services, ODE, MySQL, PostgreSQL, and ActiveMQ can, at this stage, be migrated to the cloud. All components, except ActiveMQ, are available as PaaS offerings at some cloud providers. Now, component grouping is examined further.

5.3.1 Grouping Considerations

The results of Li et al. suggest that there is no all-round best cloud provider. Instead, some providers excel in fast storage service, while others provide more cost-effective virtual instances or better wide-area latency. They see future research potential in developing a meta-cloud that combines these strengths [28]. The benefits of different cloud providers could already be used by grouping components into storage intensive, computation intensive, and latency sensitive groups, and selecting the best provider for each group.

Also considered should be the difference between intra-datacenter and inter-datacenter communication. Li et al. find intra-datacenter communication to be two to three times faster and also less expensive (often free) then inter-datacenter communication. Also, throughput variation for inter-datacenter communication is higher. Additionally, intra-datacenter la-

[22] `http://nevado.skyscreamer.org/`

tency is very low, while inter-datacenter latency corresponds with geographical distance [28]. Therefore, components that exchange lots of data or latency sensitive messages should be placed in the same datacenter. Overall, to minimize latency, all component should be geographically close to most of the users.

Differences in usage patterns and scaling requirements between components should also be considered. Some components might be used all the time, for example Fragmento. Others might only be used at certain times, for example ODE or Opal when a simulation workflow is executed. Resource extensive components such as Opal might require frequent scaling, while less resource extensive components won't. Additionally resource extensive components might impede performance of other components, if they are hosted together. Therefore, components should be divided so that they don't affect each others performance and don't prevent other components from being scaled or shut down. In conclusion, the following component groups seem sensible.

5.3.2 Suggested Component Groups

Fragmento and PostgreSQL should be hosted together at a cloud provider who offers good data storage performance, since Fragmento's purpose mainly is storing and retrieving data. Hosting both in the same datacenter would offer high data throughput between Fragmento and PostgreSQL at low or no costs. Since Fragmento does not communicate with any other components that should move to the cloud, hosting them together would not provide any benefits, provided latency is not unreasonably low (unless the SimTech Modeler or an equivalent solution is hosted in the cloud in the future). Fragmento also is the only component where it would make sense to run it continuously, which is possible in this scenario.

Opal, Opal Services, and MySQL should be grouped together. Since the Opal Services call Opal directly, they cannot be separated and have to be on one VM. MySQL should stay close to the Opal Service to keep latency and costs low, as mentioned earlier. Since this combination can require both high computing power and fast storage, a suitable cloud provider should be selected. Separating this group from other services, such as the Mock Services, and from the workflow engine ODE, has the advantage that they won't affect each others performance. Opal could also be scaled independently of the other components if necessary and the whole group could be shut down if not needed in a simulation workflow.

Mock Services (and other services added in the future) should be separated from other component for the same reasons as Opal. While the Mock Service don't require high computing power, it seams reasonable that other web services that will be part of simulation workflows, will have higher requirements, which makes separation advantageous. This also allows all the services to be scaled up, or down, or to be shut down individually. It may however be beneficial to keep all web services at one provider and one data center to keep latency low.

ODE should be separated from all other components. This way, the workflow engine wouldn't be affected by other high performance components. Additionally, if other components would crash, ODE could try to keep the workflow going by starting new instances. ODE could also be shut down if no simulation workflows are executed. A cloud provider with high computing power and close to the web services should be chosen.

ActiveMQ should be separated from all other components mainly for scalability reasons. It would also be beneficial to keep ODE and Auditing close to minimize latency.

Auditing and MySQL are comparable to Fragmento in that they mainly deal with storing and retrieving data. If Auditing would be modified so that it could be hosted in the cloud, a cloud provider with fast storage services should be selected. As already mentioned earlier, MySQL should be kept close to Auditing to maximize throughput and minimize costs. Since it communicates only with ActiveMQ, it would be beneficial to keep it relatively close to minimize latency.

5.3.3 Conclusion

Grouping components by storage, computation, latency, and scaling requirements creates five groups in the cloud. The final architecture for the SimTech Prototype in a cloud environment, based on the results of the previous sections, is shown in Figure 5.9.

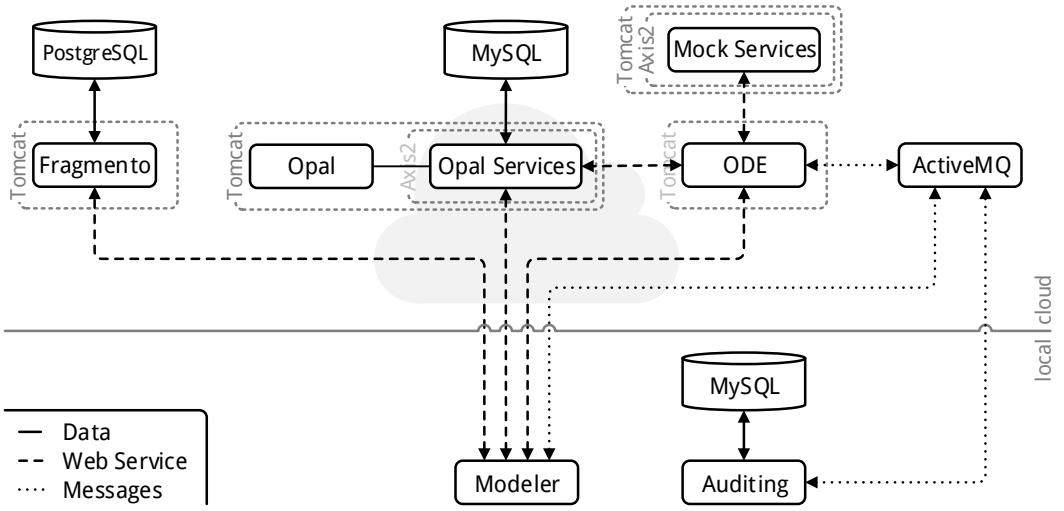

Figure 5.9: Component groups in the cloud.

6 Migrating the SimTech SWfMS to a Cloud Infrastructure

This chapter describes all steps necessary to migrate the SimTech Workflow Management System to a cloud infrastructure. For this prototypical migration, Amazon's cloud offerings will be used. Before the actual migration can start, some changes have to be made to components of the SimTech prototype, which are described in the following section.

6.1 Necessary Changes to the Software

The SimTech SWfMS has to be changed in two places to make a successful cloud migration possible. Both changes are necessary, because in the SimTech Auditing application, as well as in ODE-PGF, IP addresses are hard-coded to a value of *127.0.0.1* (or localhost). This address works only, if all components are hosted on the same physical or virtual machine, which is not always the case. The following sections show the necessary changes to make these IP addresses configurable.

6.1.1 SimTech Auditing Application

The SimTech Auditing Application has a functioning preference menu, which allows users to change the settings for connecting to the MySQL database, as well as ActiveMQ. However, the IP address set in this preference menu (and the preference files) is never read during the execution of the application. Instead, a hard-coded value of *tcp://localhost:61616* is used, as can be seen in line 41 in Listing 6.1.

JMSCommunication.java

```
35 public class JMSCommunication {
   ...
41   private String url = "tcp://localhost:61616";
42   ...
```

Listing 6.1: Excerpt of the original source code of SimTech Auditing.

The necessary changes to *JMSCommunication.java* are shown in Listing 6.2. All additional packages that are needed to read the configuration value are imported in line 3-5. Then, at the beginning of the *startup* function, the preferences are loaded in line 125-126 and the ActiveMQ URL is read in line 127, before the already existing line 128 creates a new ActiveMQ connection factory using this URL. The hard-coded URL at the beginning of the class declaration (now line 45) is left as fallback.

JMSCommunication.java

```
1   package org.simtech.workflow.ode.auditing.communication;
2
3   import org.simtech.workflow.ode.auditing.ui.Preferences;
4   import org.simtech.workflow.ode.auditing.ui.PropertyConstants;
5   import java.util.Properties;
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
119 public void startup() throws Exception {
120   if (!initialized) {
121     try {
122       /**
123        * Engine Out (subscribe)
124        */
125       Properties properties = Preferences
126         .getProperties(PropertyConstants.AUTO_SELECTION);
127       url = properties.getProperty(PropertyConstants.ACTIVEMQ_URL);
128       connectionFactory = new ActiveMQConnectionFactory(url);
129       ...
```

Listing 6.2: Excerpt of the modified source code of SimTech Auditing.

After the changes are made, SimTech Auditing has to be recompiled.

```
C:\...\org.simtech.workflow.ode.auditing\>ant -f buildAndPack.xml
```

6.1.2 ODE-PGF

ODE-PGF currently uses a hard-coded URL to communicate with ActiveMQ, as can be seen in Listing 6.3, line 4.

IConstants.java

```
1 package org.apache.ode.bpel.extensions.comm;
2
3 public interface IConstants {
4   public static String URL = "tcp://localhost:61616";
5   public static String ENGINE_OUT = "org.apache.ode.events";
6   public static String ENGINE_IN = "org.apache.ode.in";
7 }
```

Listing 6.3: Original source code of ODE-PGF with hard-coded ActiveMQ URL.

This value should be configurable in an external file to allow the URL to be easily adjusted. There is already an existing file, *ode-axis2.properties*, which can be used to store the URL. A property holding the ActiveMQ URL is added to the end of the existing properties file, as shown in Listing 6.4, line 87.

ode-axis2.properties

```
86 ...
87 activemq_url = tcp://127.0.0.1:61616
```

Listing 6.4: Addition made to the existing properties file.

This property can now be read during run-time by ODE-PGF. First, additional packages needed to read the properties file are imported at the beginning of *Communication.java*, which is shown in Listing 6.5, line 3-6. Then, the code that reads the URL from the properties is added at the beginning of the *Communication* method, shown in line 69-83, before the already existing code that creates a new ActiveMQ connection factory using this URL, in line 85.

A new URL attribute is added to the *Communication* class in line 63. The old URL attribute in *IConstants.java* is removed. Next, in line 69, a new properties object is created to load the properties file into. Then, in line 70, the location of the configuration directory is read from a system property. If that fails, lines 72-74 will try to construct the path to the configuration directory manually, using the directory out of which Tomcat7 was started. Lines 77-83 will then try to load the properties file using this path, and, if successful, read the ActiveMQ URL from it.

Communication.java

```
1  package org.apache.ode.bpel.extensions.comm;
2
3  import java.io.File;
4  import java.io.FileInputStream;
5  import java.io.IOException;
6  import java.util.Properties;
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
63   private String url = "tcp://localhost:61616";
64
65   // @stmz: startup the connections
66   private Communication() {
67     initialized = false;
68
69     Properties prop = new Properties();
70     String confDir = System.getProperty("org.apache.ode.configDir");
71     if (confDir == null) {
72       confDir = System.getProperty("user.dir") + File.separator + "webapps" +
73         File.separator + "ode" + File.separator + "WEB-INF" +
74         File.separator + "conf";
75     }
76
77     try {
78       prop.load(new FileInputStream(confDir + File.separator +
79                                     "ode-axis2.properties"));
80       url = prop.getProperty("activemq_url");
81     } catch (IOException ex) {
82       ex.printStackTrace();
83     }
84
85     factory = new ActiveMQConnectionFactory(url);
86     ...
```

Listing 6.5: Excerpt of the modified source code of ODE-PGF.

After the changes are made, ODE-PGF has to be recompiled.

```
C:\...\ode-PGF\\>mvn clean install -DskipTests=true
```

6.2 Amazon Web Services

Amazon offers IaaS, as well as PaaS cloud solutions, which will both be used during this migration. First, a migration to IaaS is carried out using Amazon Elastic Compute Cloud (Amazon EC2)[1]. EC2 allows users to easily start VMs in Amazon's cloud, which can then be customized by installing additional software.

An account for AWS has to be set up, by following the instructions during the registration process at `http://aws.amazon.com/` After the registration process is complete, login to the AWS Management Console is available at `https://console.aws.amazon.com/`. All of the following instructions will be carried out in the EC2 dashboard (`https://console.aws.amazon.com/ec2/v2/home`).

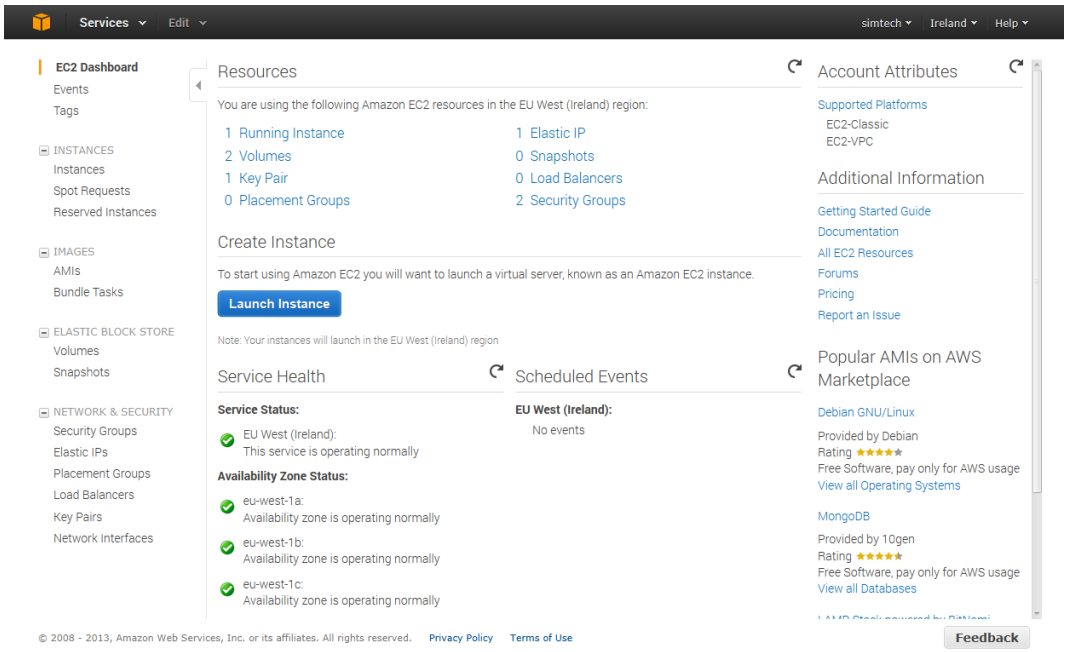

Figure 6.1: AWS EC2 management console.

6.2.1 Starting EC2 Instances

Before starting any cloud services, the region where they should be started in must be selected. In the upper right-hand corner, the region *EU (Ireland)* should be selected, since it is the closest one. Then, new EC2 instances can be launched by navigating in the left side navigation menu to the *Instances* tab and clicking on the *Launch Instance* button. Using the

[1] `http://aws.amazon.com/ec2/`

Quick Launch Wizard, the instance name can be set and a new key pair can be created and downloaded. *Ubuntu Server 12.04.1 LTS 64 bit* is selected from the launch configuration list before clicking *Continue*.

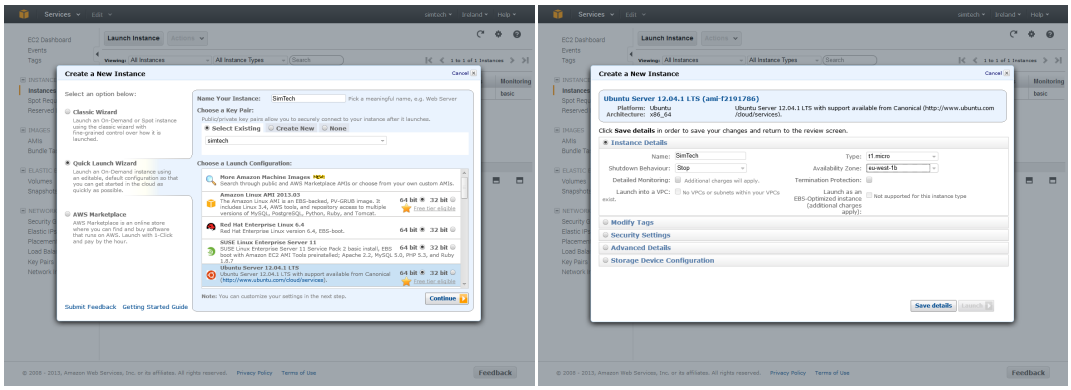

Figure 6.2: AWS EC2 quick launch wizard.

In the next screen, under *Edit Details*, an *Availability Zone* should be selected from the list (either *eu-west-1a*, *eu-west-1b*, or *eu-west-1c*). All future instances should be launched in the same availability zone, so that Elastic Block Storage (EBS) volumes can be shared between these instances. The Quick Launch Wizard is finished by clicking on *Save details* and then on *Launch*. There should now be a running EC2 instance in the instance list.

6.2.2 Elastic IPs

Each EC2 instance can be assigned an elastic IP address, so that it is always available under the same IP address. Elastic IPs can be assigned in the navigation menu on the left, under *NETWORK & SECURITY*, *Elastic IPs*. By clicking on the *Allocate New Address* button at the top and, selecting *EC2* from the *EIP used in* drop down list, and then clicking on *Yes, allocate*, an elastic IP can be allocated. There should now be a new IP address in the elastic IP address list. This IP address can be associated to an EC2 instance by right-clicking on it, selecting *associate*, and then selecting the EC2 instance from the *Instance* drop down list. In the EC2 instance list, the EC2 instance should now show this IP address in the *Elastic IP* field.

Amazon does restrict the number of elastic IP addresses to five per account. An increase can be requested by filling out an online form[2], but Amazon recommends to only use elastic IPs for front facing components such as load balancers and to use private IP addresses for internal communication. For the SimTech Prototype, five elastic IPs should be sufficient at this time. They can be used for ActiveMQ, Fragmento, ODE-PGF, the Opal Services, and the Mock Services, while communication with MySQL and PostgreSQL, which is only cloud

[2] `http://aws.amazon.com/de/contact-us/eip_limit_request/`

internal, can be done with private IP addresses. However, if any other component or new service is added in the future, five elastic IPs might not be enough.

6.2.3 Security Groups and Ports

Each EC2 instance is assigned to one or more security groups. A security group acts like a simple firewall that blocks connections, but can be configured to allow certain IP ranges to connect on certain ports. To which security group a certain instance belongs can be seen in the instance's description under the *Instances* menu. Security Groups can be changed in the *NETWORK & SECURITY* section. There, each security group can be configured by adding or removing rules under the *Inbound* tab. Changes to a security group will apply instantly to all EC2 instance assigned to this security group, after clicking on the *Apply Rule Changes* button.

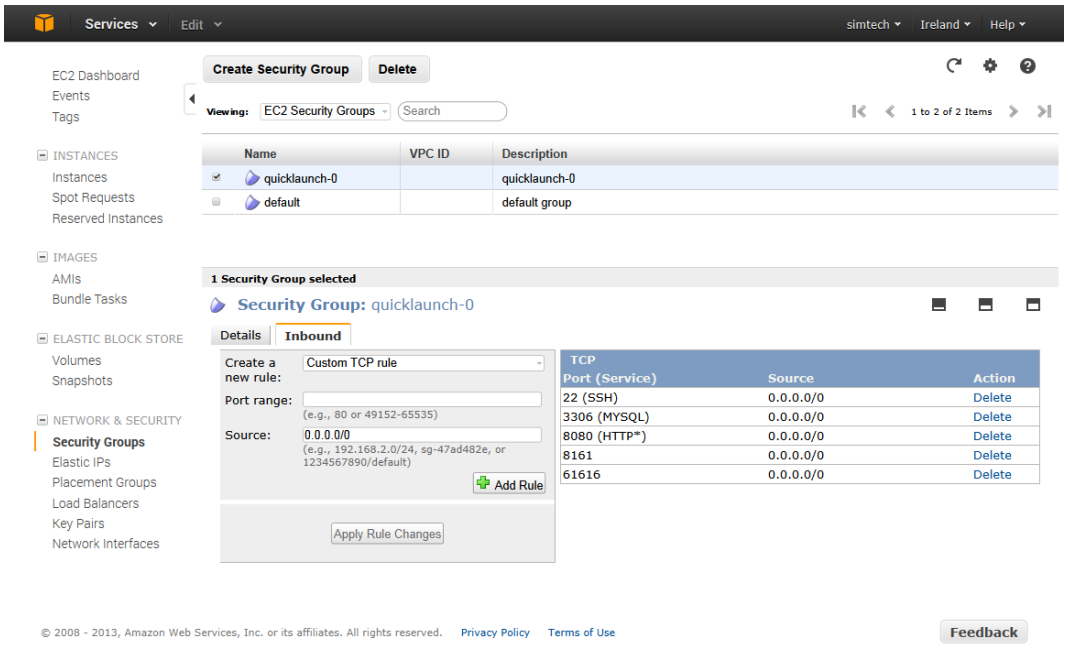

Figure 6.3: The security groups settings panel.

6.2.4 Elastic Block Storage

Elastic Block Storage is used by EC2 instances as a persistent storage medium, comparable to a physical hard drive. When a new EC2 instance is created, a new EBS volume is automatically created for it and should be visible in the EBS volumes list under *Elastic Block Store*, *Volumes*. Like a physical hard drive, EBS volumes can only be attached to one EC2 instance at a time.

An additional EBS volume can be created and used like an external hard drive to share files between EC2 instances. A new EBS volume can be created by clicking on the *Create Volume* button on the top. In the new window, the *Standard* volume type should be selected. The size should be able to accommodate all installation files, so *1 GiB* is sufficient. The Availability Zone has to be the same in which the EC2 instances where launched, since EBS volumes can't be attached to EC2 instances in other availability zones. The newly created volume has to be attached to an EC2 instance by right-clicking on the volume and selecting *Attach Volume*. The volume also has to be mounted inside the EC2 instance, which is explained in Subsection 6.3.3.

6.3 Preparation

The following sections describe preparation steps that have to be carried out once before beginning with the actual setup or once per EC2 instance during the setup. Setup instructions will refer to this section where appropriate.

6.3.1 Collecting Files

Since all files provided by SimTech are hosted on a server that is only reachable from the university network or via Virtual Private Network (VPN), EC2 instances can't access them directly. There are three options to make these files accessible to the EC2 instances. Option one is to install a VPN client on the EC2 instances and download the files through this client. Option two is to download the files to a computer that already has access to the university network and upload them to the EC2 instances from there. Option three is to host these files on a public server and secure the access with a password or similar methods. The second approach is used here.

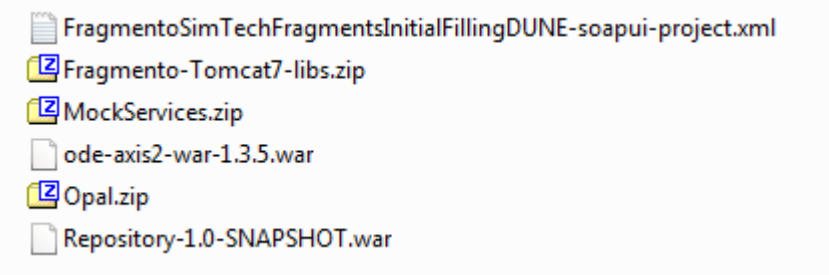

Figure 6.4: Files provided by SimTech.

The following files have to be downloaded: ODE-PGF[3] (modified as described in Subsection 6.1.2), Fragmento[4], the Fragmento Tomcat7 libraries[5], the initial filling file for Fragmento[6], the SimTech Test Web Services[7], and Opal[8], as shown in Figure 6.4.

6.3.2 SSH

Secure Shell (SSH) will be used to connect to the EC2 instances. The username for EC2 instances created from the Ubuntu AMIs is *ubuntu*. The key pair downloaded during the creation of the first EC2 instance is used instead of a password. The IP address is the elastic IP address or private IP address assigned to the server. If the PuTTY[9] SSH client is used under Windows, the key file has to be converted with *PuTTYGen* before it can be used.

6.3.3 Mounting EBS Volumes

After connecting to an EC2 instance via SSH, EBS volumes can be mounted to use them like a physical hard drive. The following command first creates a directory to mount the volume to. Then, the volume is mounted to this folder and the owner and group is changed to the *ubuntu* user.

```
$ sudo mkdir /mnt/files
$ sudo mount /dev/xvdf /mnt/files
$ sudo chown ubuntu:ubuntu /mnt/files
```

These steps have to be carried out on every EC2 instance that does not install all software via *apt-get*. To be able to mount this EBS volume to another EC2 instance, it first has to be unmounted with the *umount* command. Now, it can be detached from the current EC2 instance and reattached to another instance on the EBS management page, as described in Subsection 6.2.4.

```
$ sudo umount /dev/xvdf
```

[3] http://kaaroda.informatik.uni-stuttgart.de:8080/job/ODE-PGF/lastStableBuild/artifact/axis2-war/target/ode-axis2-war-1.3.5.war
[4] http://kaaroda.informatik.uni-stuttgart.de:8080/job/Fragmento/lastStableBuild/artifact/target/Repository-1.0-SNAPSHOT.war
[5] http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Fragmento/Fragmento-Tomcat7-libs.zip
[6] http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Fragmento/FragmentoSimTechFragmentsInitialFillingDUNE-soapui-project.xml
[7] http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Services/MockServices.zip
[8] http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Opal/Opal.zip
[9] http://www.chiark.greenend.org.uk/~sgtatham/putty/

6.3.4 Preparing Files

When the EBS volume is mounted, it can be filled with the installation files. This has to be done only once, since the volume can then be mounted to every new EC2 instance during installation. To upload the SimTech files downloaded earlier, Secure Copy (SCP), which is part of SSH installation, is used. On Windows, when using PuTTY, the following command uploads all files stored at *C:\files* to the */mnt/files* directory on the EC2 instance with the elastic IP *54.217.245.31* and user name *ubuntu*, using the PuTTY key file at *C:\simtech.ppk*.

```
C:\>pscp -i C:\simtech.ppk C.\files\* ubuntu@54.217.245.31:/mnt/files
```

After the upload has finished, the remaining files needed for the installation can be downloaded to the */mnt/files* directory using the following commands over SSH:

```
$ cd /mnt/files
$ wget http://apache.imsam.info/activemq/apache-activemq/5.7.0/apache-activem
      q-5.7.0-bin.tar.gz
$ wget http://apache.lehtivihrea.org//axis/axis2/java/core/1.5.6/axis2-1.5.6-
      war.zip
$ wget http://downloads.sourceforge.net/project/soapui/soapui/4.5.1/soapui-4.
      5.1-linux-bin.zip
$ wget http://www.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.
      1.24.tar.gz/from/http://cdn.mysql.com/ -O mysql-connector-java-5.1.24.
      tar.gz
```

All files for the installation should now be saved on the EBS volume that was mounted to */mnt/files*. This can be checked with the *ls* command.

```
$ ls /mnt/files
apache-activemq-5.7.0-bin.tar.gz
axis2-1.5.6-war.zip
FragmentoSimTechFragmentsInitialFillingDUNE-soapui-project.xml
Fragmento-Tomcat7-libs.zip
MockServices.zip
mysql-connector-java-5.1.24.tar.gz
ode-axis2-war-1.3.5.war
Opal.zip
Repository-1.0-SNAPSHOT.war
soapui-4.5.1-linux-bin.zip
```

6.3.5 Setup Preparation

Before the installation can be started, a few precautions have to be made. The apt-get cache should be updated on every EC2 instance, otherwise apt-get might not find packages that are installed later. The Java and unzip packages also have to be installed, if they are needed during installation or execution.

```
$ sudo apt-get update
$ sudo apt-get install openjdk-6-jdk
$ sudo apt-get install unzip
```

6.3.6 Swap Files

Depending on the memory available to the EC2 instance (for example on a micro instance), there might not be enough free RAM for ActiveMQ to start. Also, if Fragmento is installed together with other component on one EC2 instance, there might not be enough free memory to execute the initial filling with SoapUI. In these cases, a swap file needs to be set up, so that these tasks can be executed successfully.

```
$ sudo dd if=/dev/zero of=/var/swapfile bs=1M count=2048 &&
> sudo chmod 600 /var/swapfile &&
> sudo mkswap /var/swapfile &&
> echo /var/swapfile none swap defaults 0 0 } sudo tee -a /etc/fstab &&
> sudo swapon -a
```

6.4 IaaS Setup Instructions

The following sections describe the actual installation process of every component of the SimTech SWfMS. During this installation, IaaS, i.e. EC2 instances, are used exclusively. Instructions for PaaS solutions offered by Amazon can be found in Section 6.6.

Two setups using only EC2 instances were tested. Figure 6.5 shows the first setup, where all cloud components are run together on one singe EC2 instance. Figure 6.6 shows the second setup, where every cloud component was placed in a separate EC2 instance.

6.4.1 ActiveMQ

On the server running ActiveMQ, Java has to be installed as explained in Subsection 6.3.5. Then, the */opt/activemq/* directory is created and the content of the *apache-activemq-5.7.0-bin.tar.gz* is extracted into it. Next, ActiveMQ is added as a service and configured to start on boot. Finally, the default configuration file is build.

```
$ sudo mkdir /opt/activemq
$ sudo tar zxvf /mnt/files/apache-activemq-5.7.0-bin.tar.gz \
> -C /opt/activemq --strip-components=1

$ sudo ln -sf /opt/activemq/bin/activemq /etc/init.d/
$ sudo update-rc.d activemq defaults

$ sudo /etc/init.d/activemq setup /etc/default/activemq
```

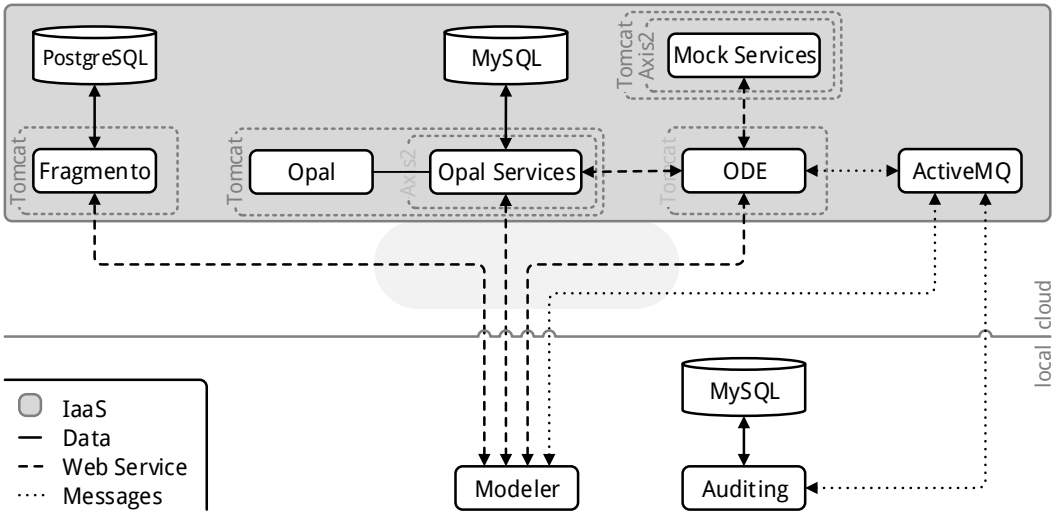

Figure 6.5: Architecture using one single EC2 instance.

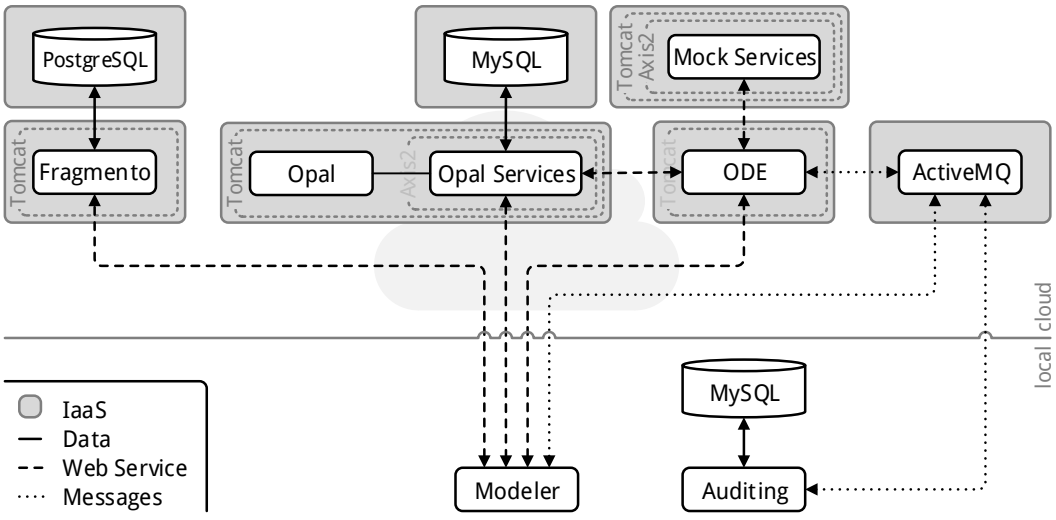

Figure 6.6: Architecture using seven EC2 instance.

ActiveMQ is now installed and the service can be started. Depending on the EC2 instance type selected (for example on a micro instance), there might be not enough memory available for ActiveMQ to start. To be able to start ActiveMQ, a swap file needs to be set up as described in Subsection 6.3.6

```
$ sudo service activemq start
```

ActiveMQ is now running, but can't talk to component outside the EC2 instance, because the right ports are not open. Ports have to be opened as described in Subsection 6.2.3. Rules for the port range *8161* (the admin panel), and for the port range *61616* (the ActiveMQ broker) have to be added. The ActiveMQ admin panel should now be available under `http://[ELASTIC_IP]:8161/admin` (the default login is admin/admin).

6.4.2 MySQL

MySQL can be installed via apt-get.

```
$ sudo apt-get install mysql-server-5.5
```

During the installation, MySQL will ask for a new password for the root user. The installation can be tested by connecting to the MySQL server as root user using this password. Once logged in, the anonymous users that MySQL creates by default have to be removed (or given a password). Otherwise, they will prevent the users created by Opal from accessing the database.

```
$ mysql -u root -p password
mysql> DELETE FROM mysql.user WHERE user='';
mysql> FLUSH PRIVILEGES;
mysql> \q
```

By default, MySQL will only accept connections from the same physical or virtual machine. The MySQL configuration file has to be changed to accept remote connection.

```
$ sudo nano /etc/mysql/my.cnf
```

In the configuration file, the *bind-address* parameter has to be set to the IP address of the server that will connect to the database remotely. Since the server hosting the Opal Services is the only server that needs access to this database, its elastic IP could be used here. If, however, in the future, more than one server needs access to this database (for example if SimTech Auditing is also moved to the cloud), *bind-address* has to be set to *0.0.0.0* to accept any remote connection. A modified *bind-address* parameter is shown in line 47 of Listing 6.6.

my.cnf

```
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address    = 0.0.0.0
...
```

Listing 6.6: Excerpt of the modified MySQL configuration file *my.cnf*.

To improve security in this case, a firewall should be used to block all IPs other than those of the servers who need access to the database. Then, MySQL has to be restarted to make the change active.

```
$ sudo service mysql restart
```

The database initialization required for Opal can also be carried out now. Unzip has to be installed, as described in Subsection 6.3.5. The Opal archive has to be extracted before the database initialization script can be executed. The requested password is the root password specified during the installation.

```
$ sudo unzip /mnt/files/Opal.zip -d /tmp
$ cd /tmp/Opal/MySQL
$ sudo bash init_database.sh
```

Finally, port *3306* has to be opened in the security group that the server running MySQL belongs to, as described in Subsection 6.2.3.

6.4.3 PostgreSQL

PostgreSQL can be installed via apt-get.

```
$ sudo apt-get install postgresql
```

Similar to MySQL, PostgreSQL only accepts local connection by default. To change this, two files have to be edited. Add the end of *pg_hba.conf*, the line *host all all 0.0.0.0/0 trust* has to be added. In *postgresql.conf*, the parameter *listen_adresses* has to be uncommented and set to '*' (to accept all remote connections), or a list of IPs (to accept only connections from those IPs).

```
$ sudo nano /etc/postgresql/9.1/main/pg_hba.conf
$ sudo nano /etc/postgresql/9.1/main/postgresql.conf
```

Listing 6.7 shows the modified *pg_hba.conf* file with the additional line added in line 88.

pg_hba.conf

```
87 # TYPE  DATABASE        USER            ADDRESS                 METHOD
88 host    all             all             0.0.0.0/0               trust
89 ...
```

Listing 6.7: Excerpt of the modified PostgreSQL configuration file *pg_hba.conf*.

Listing 6.7 shows the modified *postgresql.conf* file, where in line 59, *listen_addresses* was uncommented and set to *'*'*.

postgresql.conf

```
57 # - Connection Settings -
58
59 listen_addresses = '*'                 # what IP address(es) to listen on;
60                                         # comma-separated list of addresses;
61                                         # defaults to 'localhost', '*' = all
62                                         # (change requires restart)
63 ...
```

Listing 6.8: Excerpt of the modified PostgreSQL configuration file *postgresql.conf*.

After these changes, PostgreSQL has to be restarted.

```
$ sudo service postgresql restart
```

Next, the PostgreSQL database has to be set up for Fragmento. First, a password for the default postgres user has to be specified. Then, a new database with the name *repository* is created and the owner is set to the user *postgres*.

```
$ sudo -u postgres psql postgres
postgres=# \password postgres
postgres=# \q

$ sudo su - postgres
$ createdb repository -O postgres
$ exit
```

Finally, port *5432* has to be opened in the security group that the server running PostgreSQL belongs to, as described in Subsection 6.2.3.

6.4.4 Tomcat7

Tomcat7 can be installed via apt-get.

```
$ sudo apt-get install tomcat7
```

Inbound rule for the port range *8080* have to be added to the appropriate security groups, as described in Subsection 6.2.3. The Tomcat7 test page should then be available at `http://[ELASTIC_IP]:8080` For now, the Tomcat7 service should be stopped again.

```
sudo service tomcat7 stop
```

6.4.5 Axis2

Unzip has to be installed on this server, as described in Subsection 6.3.5. Then, the *axis2-1.5.6-war.zip* archive has to be extracted into the Tomcat7 *webapps* directory.

```
$ sudo unzip /mnt/files/axis2-1.5.6-war.zip -d /var/lib/tomcat7/webapps
```

The Tomcat7 service can be started again to verify that axis is deployed properly. The Axis2 admin page should be reachable under `http://[ELASTIC_IP]:8080/axis2`. Then, Tomcat7 should be stopped again.

```
$ sudo service tomcat7 start
$ sudo service tomcat7 stop
```

6.4.6 ODE-PGF

On the server running ODE-PGF, Tomcat7 has to be installed as explained in Subsection 6.4.4. Then, *ode-axis2-war-1.3.5.war* should be copied into the Tomcat7 *webapps* folder. The Tomcat7 service can be started to test the deployment. ODE should be reachable under `http://[ELASTIC_IP]:8080/ode`.

```
$ sudo cp /mnt/files/ode-axis2-war-1.3.5.war /var/lib/tomcat7/webapps/ode.war
$ sudo service tomcat7 start
```

During this deployment, the configuration file *ode-axis2.properties* was extracted to */var/lib/tomcat7/webapps/ode/WEB-INF/conf*. The *activemq_url* parameter in this configuration file has to be set to the appropriate ActiveMQ IP address. Then, Tomcat7 has to be restarted to apply the changes.

```
$ sudo nano /var/lib/tomcat7/webapps/ode/WEB-INF/conf/ode-axis2.properties
$ sudo service tomcat7 restart
```

6.4.7 Fragmento

On the server running Fragmento, Tomcat has to be installed as explained in Subsection 6.4.4. Also, Java and unzip have to be installed, as described in Subsection 6.3.5. For Fragmento, additional libraries have to be extracted to Tomcat7. Then, *Repository-1.0-SNAPSHOT.war* is copied into the Tomcat7 *webapps* directory. Then, two of the libraries that were just copied have to be removed, or Tomcat7 will fail to start.

```
$ sudo mkdir /var/lib/tomcat7/lib
$ sudo unzip /mnt/files/Fragmento-Tomcat7-libs.zip -d /var/lib/tomcat7/lib

$ sudo cp /mnt/files/Repository-1.0-SNAPSHOT.war \
> /var/lib/tomcat7/webapps/Repository.war

$ sudo rm /var/lib/tomcat7/lib/naming-factory.jar
$ sudo rm /var/lib/tomcat7/lib/naming-resources.jar
```

Now, he Tomcat7 service can be started again. Fragmento should now be reachable under `http://[ELASTIC_IP]:8080/Repository/start.htm`.

```
$ sudo service tomcat7 start
```

The connection settings for the PostgreSQL database also have to be adjusted. They are saved in */var/lib/tomcat7/webapps/Repository/META-INF/context.xml*. The username, password and URL have to be set to the appropriate values of the already installed PostgeSQL database. Then, Tomcat7 has to be restarted to apply the changes.

```
sudo nano /var/lib/tomcat7/webapps/Repository/META-INF/context.xml
sudo service tomcat7 restart
```

6.4.8 SoapUI

SoapUI has to be installed on the same server as Fragmento. It is installed by extracting the *soapui-4.5.1-linux-bin.zip* to */usr/local*.

```
$ sudo unzip /mnt/files/soapui-4.5.1-linux-bin.zip -d /usr/local/
```

Now, the initial filling of Fragmento using SoapUI is executed. If it fails because not enough memory could be allocated, a swap file has to be created as described in Subsection 6.3.6, be-

fore trying again. The artifact overview at `http://[ELASTIC_IP]:8080/Repository/artefact_overview.htm` should now list the artifacts.

```
$ bash /usr/local/soapui-4.5.1/bin/testrunner.sh \
> -s"Fragment Initial Filling" \
> -c"Publish SimTech Fragment Bundles" \
> /mnt/files/FragmentoSimTechFragmentsInitialFillingDUNE-soapui-project.xml
```

6.4.9 SimTech Test Services

On the server running the SimTech Test Services, Tomcat7 and Axis2 have to be installed as explained in Subsection 6.4.4 and Subsection 6.4.5. Also, unzip has to be installed, as described in Subsection 6.3.5. Then, the *MockServices.zip* archive can be extracted into the Tomcat7 *webapps* directory. Then, Tomcat7 has to be started again.

```
$ sudo unzip /mnt/files/MockServices.zip \
> -d /var/lib/tomcat7/webapps/axis2/WEB-INF/services
$ sudo service tomcat7 start
```

6.4.10 Opal

On the server running Opal, Tomcat7 and Axis2 have to be installed as explained in Subsection 6.4.4 and Subsection 6.4.5. Also, unzip has to be installed, as described in Subsection 6.3.5. Then, *mysql-connector-java-5.1.24-bin.jar* is extracted from the *mysql-connector-java-5.1.24.tar.gz* archive to the Tomcat7 *webapps/axis2/WEB-INF/lib* directory. The database should have already been initialized during the MySQL installation, as described in Subsection 6.4.2. The *Opal.zip* archive is extracted to the */tmp* directory and the content of the */tmp/Tomcat/* directory is copied to */var/lib/tomcat7*. Now, Tomcat7 can be started again.

```
$ sudo tar zxvf /mnt/files/mysql-connector-java-5.1.24.tar.gz \
> -C /var/lib/tomcat7/webapps/axis2/WEB-INF/lib --wildcards --no-anchored \
> --strip-components=1 'mysql-connector-java-5.1.24-bin.jar'

$ sudo unzip /mnt/files/Opal.zip -d /tmp
$ sudo cp -a /tmp/Opal/Tomcat7/. /var/lib/tomcat7
$ sudo service tomcat7 start
```

During this deployment, the configuration files *opalmgr.properties*, *opalservice.properties*, and *resmgr.properties*, were extracted to the */var/lib/tomcat7/webapps/axis2/WEB-INF/classes* directory. Inside these files, the IP addresses for the MySQL database, as well as the ODE-PGF server have to be adjusted. The executable paths for Opal also need to be changed. Then, Tomcat7 has to be restarted to apply the changes.

```
$ sudo nano /var/lib/tomcat7/webapps/axis2/WEB-INF/classes/opalmgr.properties
$ sudo nano /var/lib/tomcat7/webapps/axis2/WEB-INF/classes/opalservice.properties
$ sudo nano /var/lib/tomcat7/webapps/axis2/WEB-INF/classes/resmgr.properties
$ sudo service tomcat7 restart
```

6.5 Connecting Local Components to the Cloud

Now that all components are migrated to the cloud, the SimTech Modeler and SimTech Auditing, the remaining two local components, can be connected to the cloud. First, the modified version of Auditing should be installed by extracting the archive to a directory on the hard drive. Then, Auditing should be started by running the following command inside this directory:

```
C:\SimTechAuditing>java -jar SimTechAuditing.jar
```

In the options dialog under *File*, *Preferences*, the settings for the MySQL database connection can be adjusted. Then, the ActiveMQ URL parameter has to be changed to the elastic IP of the EC2 instance that runs ActiveMQ. When the settings are saved, Auditing should be able to connect to ActiveMQ by clicking on the *File*, *Start listening* menu entry.

Eclipse and the SimTech Eclipse Plugins should be set up following the instructions in the SimTech installation manual. Once set up, some processes should be loaded to test the connection with, for example the processes contained in *TestProzesse.zip*[10]. Now, the connections to the various components in the cloud can be set up.

First, the connection to Fragmento is established. The URI parameter in the Fragmento service options dialog hast to be changed to contain the elastic IP address of the EC2 instance hosting Fragmento. By clicking on *Apply* and then on *Retrieve Repository*, the repository view should be filled with the artifacts from the Fragmento repository.

Next, the URLs for ActiveMQ and ODE-PGF have to be set in the preference dialog under *SimTech*, *Preferences*. The ActiveMQ URL parameter has to be changed to point to the EC2 instance hosting ActiveMQ. The Ode server URL has to be changed to the elastic IP address of the EC2 instance hosting ODE-PGF.

After applying the changes, a test process can be opened. Before it can be deployed in the cloud, its *.wsdl* files has to be adjusted. Inside this file, the service port URL has to be changed to the elastic IP address of the server hosting ODE-PGF. After saving the changed file, the process should now be able to be deployed in the cloud.

[10] http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Processes/TestProzesse.zip

6.6 PaaS Setup Instructions

Amazon offers PaaS solutions for databases, as well as application servers, as shown in Section 5.2. Amazon Relational Database Service (Amazon RDS)[11] offers MySQL, Oracle, and Microsoft SQL Server as PaaS. AWS Elastic Beanstalk[12] offers flexible, auto-scaling application server environments for Java, .Net, PHP, Python, Ruby, and Node.js applications. Figure 6.7 shows the architecture of the SimTech prototype in the Amazon cloud, if these PaaS services are used. The MySQL database can be replaced with a RDS instance, and Fragmento, Opal, ODE and the Mock Services can be run with Elastic Beanstalk. PostgreSQL and ActiveMQ remain on EC2 instances.

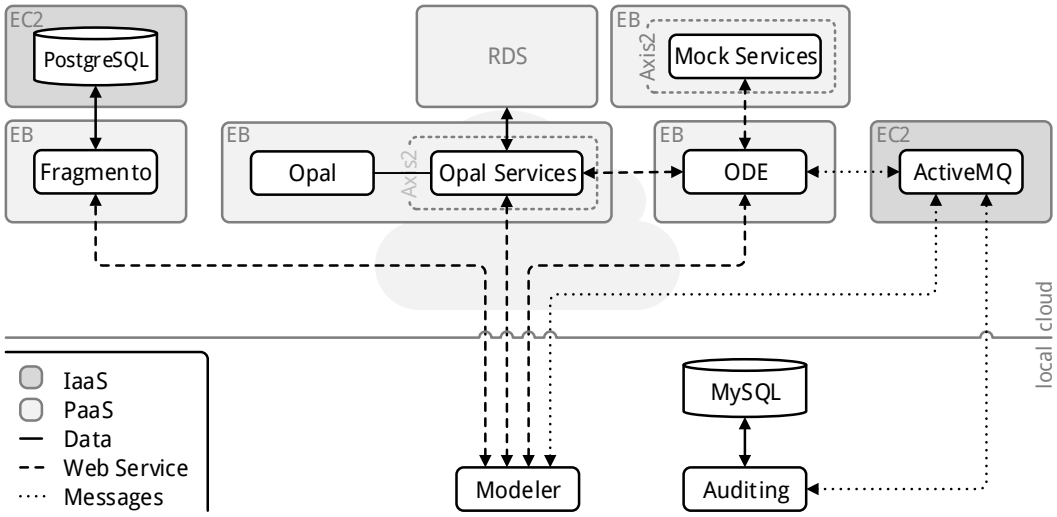

Figure 6.7: Architecture including Amazon PaaS offerings where possible.

6.6.1 Amazon RDS

Amazon RDS management is done in a separate management dashboard, which is available at `https://console.aws.amazon.com/rds/`. As with EC2 instances, the region in the upper right-hand corner should be set to *Ireland* before adding any database instances. Now, new database instances can be added by clicking the *Launch DB Instance* button at the top of the *DB Instances* screen.

The *Launch DB Instance Wizard* will open, which allows for various settings to be changed. MySQL should be selected as database engine during the first step. In the second step, different MySQL version and instance classes can be selected, and storage space has to

[11] `http://aws.amazon.com/rds/`
[12] `http://aws.amazon.com/elasticbeanstalk/`

be allocated. The database instance is also given a name (*DB Instance Identifier*) and the master user and his password are set. In step three, the availability zone can be selected, which should be the same that was used for the EC2 instances. *Database Name* should be left blank, so that no database will be created initially. In step four, backup settings can be adjusted as needed.

After the new database instance has started, its security group settings need to be adjusted to allow remote access. In the *DB Security Groups* window, the security group assigned to this database instance should be selected to show its details at the bottom. In the *Description* tab, two types of authorized connections can be added. The *CIDR/IP* option allows to authorize connections from certain IP addresses. The *EC2 Security Group* option authorizes all EC2 instances belonging to the selected security group to connect to the database instance. The security group to which the Opal server belongs should be selected and authorized by clicking on the *Add* button.

The Opal server should now be able to access this database instance. The database for Opal can now be initialized from this server via SSH. However, the *init_database.sh* script has to be modified, since it assumes a local MySQL database.

```
$ sudo nano /tmp/Opal/MySQL/init_database.sh
```

Listing 6.9 shows a modified version of the *init_database.sh* script. In line two, a prompt was added to ask the user for the MySQL host address. The prompt is filled with the default value *127.0.0.1*, which can be changed by the user to point to a remote MySQL server, for example, with the *Endpoint* address of a RDS instance, which can be found in its description. Each *mysql* call is then modified to include the host option (*-h*), which uses the value supplied by the user. The user option (*-u*) also has to be adjusted, if the master user name chosen during the RDS setup is not *root*.

After saving the file, the Opal database can now be initialized as before.

```
$ sudo bash init_database.sh
```

The MySQL settings in *opalmgr.properties*, *opalservice.properties*, and *resmgr.properties* also have to be changed to contain the proper endpoint address, as described in Subsection 6.4.10.

6.6.2 AWS Elastic Beanstalk

The Elastic Beanstalk management console can be accessed at `https://console.aws.amazon.com/elasticbeanstalk`. As with EC2 instances, the region in the upper right-hand corner should be set to *Ireland* before creating a new application. A new application can be launched by clicking on *Create New Application* in the top right-hand corner (it might be necessary to skip the introduction screen that appears on the first login). In the *Applica-*

init_database.sh

```
read -e -p "Enter MySQL root password: " password
read -e -p "Enter MySQL host address: " -i "127.0.0.1" host
mysql -h $host -u root -p$password < opalmanager_dist_opal_bcclat.sql
mysql -h $host -u root -p$password < opalmanager_dist_opal_energies.sql
mysql -h $host -u root -p$password < opalmanager_dist_opal_simulations.sql
mysql -h $host -u root -p$password < opalmanager_dist_opal_procinstances.sql
mysql -h $host -u root -p$password < resmanager_dist_rm_managementcontexts.sql
mysql -h $host -u root -p$password < resmanager_dist_rm_servers.sql
mysql -h $host -u root -p$password < resmanager_dist_rm_serviceendpoints.sql
mysql -h $host -u root -p$password < resmanager_dist_rm_servicetickets.sql
mysql -h $host -u root -p$password < createUsers.sql
echo "Datenbank initalisiert :)"
```

Listing 6.9: Modified version of the Opal database initialization script *init_database.sh*.

tion Details screen, *64bit Amazon Linux running Tomcat7* is selected as *Container Type*, and a *.war* file is selected to be uploaded as *Application Source*. In the *Environment Details* tab, the environment name, URL, and description can be set. Additionally, a RDS instance can also be created for this environment, if required.

Further configuration details can be changed on the *Configuration Details* screen. For example, the name of an existing key pair could be added to the *Existing Key Pair* field, to use an already created key pair to access EC2 instances created by Elastic Beanstalk. If a database instance was enabled previously, its settings can also be changed here, similar to Subsection 6.6.1. After reviewing all settings and clicking on *Finish*, Elastic Beanstalk will launch the complete environment, which can take several minutes. When the environment is launched, the selected application should be accessible at the environment URL specified earlier.

There are some differences to be noted when accessing an application created with Elastic Beanstalk compared to IaaS approach. Elastic Beanstalk by default uses port 80 instead of 8080 to make the applications hosted on Tomcat available to the outside. Also, since Elastic Beanstalk launches the application in an auto scaling group, which has a load balancer in front of it, the application should no longer be accessed through the IP address of an EC2 instance, but instead, through the load balancer's Domain Name System (DNS) name. So, for example, when changing configuration files to access an ODE-PGF launched with Elastic Beanstalk, the right address would now be `http://[LOAD-BALANCER-DNS-NAME]:80/` (note: no *ode* at the end) instead of `http://[ELASTIC-IP]:8080/ode`.

The load balancer and its DNS name can be found in the EC2 dashboard under *NETWORK & SECURITY - Load Balancers*. All other components launched by Elastic Beanstalk can also be found in their respective dashboards. As during the IaaS chapter, the EC2 instances launched by Elastic Beanstalk can be accessed them same way via SSH.

6.7 Problems and Concerns

In this section, some problems and concerns are described. When using PaaS solutions there are some problems that need to be solved to be able to use auto scaling. There are also security concerns that need to be addressed, mainly when using IaaS.

6.7.1 PaaS and Automatic Scaling

One of the benefit of PaaS deployment is automatic scalability. Elastic Beanstalk provided this with the auto scaling feature. If a certain specified trigger level is reached, EC2 instances are added to or removed from the auto scaling group, which is automatically set up by Elastic Beanstalk. The new instances are created from the Amazon Machine Image (AMI) and the applications *.war* file, the same way that the very first instance was created. This means that all changes made to the first instance after it was created, e.g., the adjustment of IP addresses in configuration files, as described in Section 6.4, are not replicated on the instances created by auto scaling. However, these changes have to present on newly created instances in order to make them function as intended.

It is neither realistic, nor reasonable, that each time a new instance is created by auto scaling, someone will make these changes manually. There are however a few different options to make auto scaling work with various levels of effort involved. For one, all changes currently done after the installation could be done in the source code, so that they are already part of the *.war* files used to launch the applications. For settings that won't change during deployment, this may make sense. But for other settings, for example IP addresses, which could change during deployment, this would mean that the application would have to be repacked and redeployed from the modified *.war* file whenever some parameter changes.

The second option is to use custom AMIs. The default AMIs used by Elastic Beanstalk can be used to start a single EC2 instance. This instance can then be modified as described in Section 6.4 with all necessary components and configuration changes. From this instance, a new AMI can be created with all these modifications baked in. This custom AMI can then be used by Elastic Beanstalk to start all instances [42]. This option has the same disadvantages as the first one. Each time that something has to change on an instance, even if it is only one bit, a new AMI has to be created to replace the old one. Additionally, these AMIs have to be stored in the cloud to be accessible for Elastic Beanstalk, which increases costs.

Option three is to use Elastic Beanstalks custom environment properties. In the AWS management console or with the AWS toolkit for Eclipse, custom environment properties can be specified. These properties are passed to the individual EC2 instances, where they can then be read, for example in Java, with the *System.getProperty()* call [43]. This option has several disadvantages. The source code of most components would have to be modified to read in these environment variables, but this would only have to be done once. One instance can't use more than six custom environment variables, including a database connection string.

Also, this option can only deal with configuration values, while all other options can handle additional packages that have to be installed or other changes that might be necessary.

The fourth option is to use the Elastic Beanstalks customization option. If a configuration file with the extension *.config* is placed in a directory named *.ebextensions* inside a *.war* file, Elastic Beanstalk will execute this configuration file during deployment [8]. These configuration files are written in a YAML[13] syntax and can download and install or unpack packages and archives, create files, users, and groups, manage services, and execute commands [9]. For this option, *.war* files would have to be modified, but compared to the first and second option, these configuration files are more powerful and flexible. Particularly interesting is the option to create files from a remote source. This would allow configuration files that have to be changed often to be stored outside the *.war* files, for example in an Amazon S3 bucket. Changes could then be made to those files and Elastic Beanstalk could duplicate those changes on the individual instances during deployment.

Option five is to use a provisioning solution like Opscode Chef[14] or Puppet Labs Puppet[15]. Similar to option four, these solutions can create files, run commands, manage users, groups, and service, and more. Instead of embedding the configuration file inside the applications *.war* archive, these solutions use a client-server architecture to load configuration descriptions remotely during deployment. To be able to do so, a client component has to be installed on every instance during deployment. From all alternatives described here, these are the most powerful and flexible, but also require the most effort to set up.

6.7.2 Security

Security is also a concern, especially when using the IaaS approach. All the software is managed by the user, which includes any precautions necessary to make components secure. The user has to initially secure each component and also has to stay up to date on new security vulnerabilities that could appear after the initial deployment. This will increase the effort necessary when using IaaS, compared to PaaS, where security for the provided services is the responsibility of the service provider. During the deployment described in this chapter, no special security measurements have been made. A greater effort should be made to secure the application during an actual migration.

6.8 Possible Improvements

In this thesis, a manual approach was used to install the various components in the cloud. The effectiveness of the installation and management process can be greatly improved by

[13] http://www.yaml.org/
[14] http://www.opscode.com/chef/
[15] https://puppetlabs.com/

using the various tools provided by cloud providers and third parties. All these solution allow users to describe a desired architecture in some textual format. Depending on the solution, things that can be described are the resources that are needed, the software that should be installed on them, how resources and software should be configured, how components are connected, and how everything is managed. These descriptions are then interpreted and applied by the provision software.

Amazon offers AWS CloudFormation[16], but there are also many third party provisioning solutions that can automate the setup and management of cloud environments. In Subsection 6.7.1, Chef and Puppet were already mentioned. Other solutions are RightScale RightScripts[17], ControlTier [18], CFEngine [19], SmartFrog [20], and others. Amazon also recently introduced AWS OpsWorks[21], a solution that integrates Chef into Amazon's management environment, which simplifies the usage of Chef with Amazon's cloud resources.

[16] http://aws.amazon.com/cloudformation/
[17] http://support.rightscale.com/12-Guides/Dashboard_Users_Guide/Design/RightScripts
[18] http://doc36.controltier.org/wiki/Main_Page
[19] http://cfengine.com/
[20] http://wiki.smartfrog.org/
[21] http://aws.amazon.com/opsworks/

7 Summary and Conclusion

The SimTech SWfMS has been developed at the University Stuttgart as part of the Excellence Initiative funded by the German government. It allows scientist to create, manage, and execute simulation workflows, which describe task that are necessary to create a simulation and the order in which they should be executed. Chapter 2 explained the separate components of the system, which each handle a specific function. These components communicate with messages or Web Service calls, which makes it reasonable to migrate this system to a cloud environment.

Chapter 3 described cloud computing in more detail. Cloud services can be rapidly provisioned, are highly scalable, and are offered on a pay-per-use basis. This makes them ideal for applications with unpredictable or fluctuating usage patterns.

To take advantage of the benefits of cloud computing, applications have to be migrated into a cloud environment. While not necessarily different from any other software migration, there are some aspect that need special attention when migrating to the cloud, such as software licensing and compliance, which was mentioned in Chapter 4. In this case however, they are not an issue.

Chapter 5 showed that all components, except the SimTech Modeler and SimTech Auditing, can be migrated to the cloud with very little or no changes. Also, while comparing different cloud providers, it became evident that not all requirements of the SimTech Prototype are met with suitable PaaS offerings. While MySQL databases and Tomcat7 application servers are in general widely available, support for PostgreSQL is not as widespread, and ActiveMQ as PaaS does not exist.

The small changes to SimTech Auditing and ODE-PGF that were necessary to make the cloud migration possible were described in Chapter 6. Then, a prototypical migration to Amazon's cloud environment was carried out, testing both IaaS and PaaS solutions. EC2 instances were used for IaaS, while PaaS was tested with RDS for MySQL databases and Elastic Beanstalk as Tomcat7 application servers.

To fully utilize PaaS solutions, additional effort has to be made. Additional security measurements should also be taken when deploying in the cloud. The manual installation process that was described can also be improved by using provisioning software.

In conclusion, a migration to a cloud environment is possible. With very little effort, most components of the SimTech SWfMS can be migrated to a mixture of IaaS and PaaS solutions. After this initial migration, more time could be spent on refining already migrated components or migrating the rest. It would probably make sense to make some additional

effort to fully utilize PaaS, and to find a PaaS solution for ActiveMQ. Then, SimTech Auditing could be modified to be able to be migrated to a cloud environment. Moving the SimTech Modeler to the cloud, or providing a suitable replacement, will probably require the most effort. However, it is not absolutely necessary to move the SimTech Modeler to the cloud and it should definitely not hold back a cloud migration of the rest of the SimTech SWfMS.

Bibliography

[1] Ellen Ackermann. “Ein Referenz-Prozess der Software-Migration”. Diploma. University Koblenz-Landau, 2005. URL: http://www.uni-koblenz.de/~ist/documents/ackermann2005da.pdf.

[2] Ellen Ackermann, Rainer Gimnich, and Andreas Winter. “Ein Referenz-Prozess der Software-Migration (erweiterte Kurzfassung)”. In: *Softwaretechnik-Trends* 25.4 (2005), pp. 20–22. URL: http://www.uni-koblenz.de/~ist/documents/Ackermann2005ERD.pdf.

[3] *Amazon Elastic Compute Cloud User Guide*. Amazon Web Services, Inc. Dec. 2012. URL: http://awsdocs.s3.amazonaws.com/EC2/latest/ec2-ug.pdf.

[4] Michael Armbrust et al. *Above the Clouds: A Berkeley View of Cloud Computing*. Tech. rep. UC Berkeley, Feb. 2009. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf.

[5] Marco Battaglia, G. Savoia, and J. Favaro. “Renaissance: A Method to Migrate from Legacy to Immortal Software Systems”. In: Second Euromicro Conference on Software Maintenance and Reengineering. 1998, pp. 197–200.

[6] Michael Behrendt et al. *IBM Cloud Computing Reference Architecture 2.0*. Tech. rep. IBM, Inc., Feb. 2011.

[7] Michael L. Brodie and Michael Stonebraker. “DARWIN: On the Incremental Migration of Legacy Information Systems”. In: (Mar. 1993). URL: http://db.cs.berkeley.edu/papers/S2K-93-25.pdf.

[8] *Customizing and Configuring AWS Elastic Beanstalk Environments*. Amazon Web Services, Inc. URL: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-containers.html.

[9] *Customizing the Software on EC2 Instances Running Linux*. Amazon Web Services, Inc. URL: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-containers-ec2.html.

[10] Mathias Dalheimer and Franz-Josef Pfreundt. “GenLM: License Management for Grid and Cloud Computing Environments”. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. 2009, pp. 132–139. URL: http://www.enterprisegrids.fraunhofer.de/fhg/Images/GenLM-preprint_tcm401-148731.pdf.

[11] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel. "Infrastructure as a Service Security: Challenges and Solutions". In: *Informatics and Systems (INFOS), 2010 The 7th International Conference on*. Mar. 2010, pp. 1–8. URL: http://www.researchgate.net/publication/224136774_Infrastructure_as_a_service_security_Challenges_and_solutions/file/9fcfd50ead5ff22b8d.pdf.

[12] *Decisions on the Second Programme Phase of the Excellence Initiative*. German Research Foundation. URL: http://www.dfg.de/en/service/press/press_releases/2012/press_release_no_26/index.html.

[13] *Eclipse Web Interface*. The Eclipse Foundation. URL: http://wiki.eclipse.org/Eclipse_Web_Interface.

[14] Christoph Fehling and Frank Leymann. *Cloud Computing*. Version 5. Gabler Wirtschaftslexikon. URL: http://wirtschaftslexikon.gabler.de/Archiv/1020864/cloud-computing-v5.html.

[15] Christoph Fehling and Ralph Retter. *Cloud Computing Patterns*. URL: http://cloudcomputingpatterns.org/.

[16] Christoph Fehling et al. *A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based Application Architectures*. Tech. rep. University Stuttgart, Institute of Architecture of Application Systems, May 2011. URL: http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/fehling/TR-2011-05%20Patterns_for_Cloud_Computing.pdf.

[17] Christoph Fehling et al. "An Architectural Pattern Language of Cloud-based Applications". In: 18th Conference on Pattern Languages of Programs, PLoP. 2011. URL: http://www.hillside.net/plop/2011/papers/A-20-Fehling.pdf.

[18] Marco Finetti, Karin Friedsam, and Dr. Beate Konze-Thomas. *Excellence Initiative at a Glance*. Tech. rep. German Research Foundation, Apr. 2011. URL: http://www.dfg.de/download/pdf/dfg_im_profil/geschaeftsstelle/publikationen/exin_broschuere_1104_en.pdf.

[19] *Flexibility of Simulation Workflows*. SRC Simulation Technology. URL: http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_flexibility.php.

[20] Yolanda Gil et al. "Examining the Challenges of Scientific Workflows". In: *IEEE Computer* 40.12 (Dec. 2007), pp. 24–32. URL: http://eprints.soton.ac.uk/271187/1/computer07.pdf.

[21] Rainer Gimnich and Andreas Winter. "Workflows der Software-Migration". In: *Softwaretechnik-Trends* 25.2 (2005), pp. 22–24. URL: http://www.uni-koblenz.de/~ist/documents/Gimnich2005WDS.pdf.

[22] David Hollingsworth. *The Workflow Reference Model*. Tech. rep. Workflow Management Coalition, Jan. 1995. URL: http://www.wfmc.org/standards/docs/tc003v11.pdf.

[23] *Human Users in Simulation Workflows*. SRC Simulation Technology. URL: http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_humanusers.php.

[24] Ali Khajeh-Hosseini et al. "Decision Support Tools for Cloud Migration in the Enterprise". In: 2011 IEEE International Conference on Cloud Computing (CLOUD). 2011, pp. 541–548. URL: http://arxiv.org/ftp/arxiv/papers/1105/1105.0149.pdf.

[25] Johannes Kirschnick et al. "Towards an Architecture for the Automated Provisioning of Cloud Services". In: *IEEE Communications Magazine* 48.12 (Dec. 2010), pp. 124–131. URL: http://jmalcaraz.com/wp-content/uploads/papers/AlcarazCalero-2010-CommMag-Preprint.pdf.

[26] Heike Lehmann. *SimTech - Cluster of Excellence*. Tech. rep. SRC Simulation Technology, July 2009. URL: http://www.simtech.uni-stuttgart.de/downloads/files/SimTech_Broschuere.pdf.

[27] Alexander Lenk et al. "What's Inside the Cloud? An Architectural Map of the Cloud Landscape". In: ICSE Workshop on Software Engineering Challenges of Cloud Computing. 2009, pp. 23–31. URL: http://www.di.ufpe.br/~redis/intranet/bibliography/middleware/lenk-what-2009.pdf.

[28] Ang Li et al. "CloudCmp: Comparing Public Cloud Providers". In: 10th ACM SIGCOMM conference on Internet measurement. 2010, pp. 1–14. URL: http://ftp.cs.duke.edu/~xwy/publications/cloudcmp-imc10.pdf.

[29] Fang Liu et al. *NIST Cloud Computing Reference Architecture*. Tech. rep. National Institute of Standards and Technology, Sept. 2011. URL: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=909505.

[30] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing*. Tech. rep. National Institute of Standards and Technology, Sept. 2011. URL: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[31] *Migration*. Oxford University Press. URL: http://oxforddictionaries.com/definition/english/migration.

[32] *Modelling of Simulation Workflows*. SRC Simulation Technology. URL: http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_modeling.php.

[33] *Orion*. The Eclipse Foundation. URL: http://www.eclipse.org/orion/.

[34] *PN 8: Integrated data management, workflow and visualisation to enable an integrative systems science*. SRC Simulation Technology. URL: http://www.simtech.uni-stuttgart.de/forschung/pn/pn8/index.en.html.

[35] Václav T. Rajlich and Keith H. Bennett. "A Staged Model for the Software Life Cycle". In: *Computer* 33.7 (July 2000), pp. 66–71. URL: http://www.cs.wayne.edu/~vip/publications/Rajlich.IC.2000.StagedModel.pdf.

[36] *Remote Application Platform*. The Eclipse Foundation. URL: http://www.eclipse.org/rap/.

[37] *Research Areas*. SRC Simulation Technology. URL: http://www.simtech.uni-stuttgart.de/forschung/forschungsfelder/index.en.html.

[38] *Runtime for Simulation Workflows*. SRC Simulation Technology. URL: http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/project_execution.php.

[39] *Simulation Workflows*. SRC Simulation Technology. URL: http://www.iaas.uni-stuttgart.de/forschung/projects/simtech/sim-workflows.php.

[40] *SLA (Service-level Agreement)*. Gartner, Inc. URL: http://www.gartner.com/it-glossary/sla-service-level-agreement/.

[41] *SMILA/BPEL Designer - RAP Showcase*. The Eclipse Foundation. URL: http://wiki.eclipse.org/SMILA/BPEL_Designer#RAP_showcase.

[42] *Using Custom AMIs*. Amazon Web Services, Inc. URL: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.customenv.html.

[43] *Using Custom Environment Properties with AWS Elastic Beanstalk*. Amazon Web Services, Inc. URL: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_Java.managing.html.

[44] Luis M. Vaquero et al. "A Break in the Clouds: Towards a Cloud Definition". In: *ACM SIGCOMM Computer Communication Review* 39.1 (Jan. 2009), pp. 50–55. URL: http://www.sigcomm.org/sites/default/files/ccr/papers/2009/January/1496091-1496100.pdf.

[45] Matthias Wieland et al. "Towards Reference Passing in Web Service and Workflow-Based Applications". In: 13th IEEE International Enterprise Distributed Object Computing Conference, EDOC. 2009, pp. 109–118. URL: http://www.iaas.uni-stuttgart.de/institut/ehemalige/schumm/INPROC-2009-52%20-%20Towards%20Reference%20Passing%20in%20Web%20Service%20and%20Workflow-based%20Applications%20-%20Authors%20Postprint.pdf.

[46] Bing Wu et al. "The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems". In: (1997), pp. 200–205. URL: http://www.tara.tcd.ie/bitstream/2262/27040/1/The%20Butterfly%20Methodology%20a%20gateway-free%20approach%20for%20migrating%20legacy%20information%20systems.pdf.

[47] Lamia Youseff, Maria Butrico, and Dilma Da Silva. "Toward a Unified Ontology of Cloud Computing". In: Grid Computing Environments Workshop. 2008, pp. 1–10. URL: http://dosen.narotama.ac.id/wp-content/uploads/2012/01/Toward-a-Unified-Ontology-of-Cloud-Computing.pdf.

All links were last visited on June 4, 2013.

Declaration of Authorship

I hereby certify that the student thesis entitled

Migrating the SimTech Workflow Management System to a Cloud Infrastructure

is entirely my own work except where otherwise indicated. Passages and ideas from other sources have been clearly indicated. To date, neither this student thesis nor essential parts thereof were subject of an examination procedure. Until now, I don't have published this student thesis or parts thereof. The electronic copy is identical to the submitted copy.

Stuttgart, June 4, 2013, ...

(Lukas Reinfurt)