Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Student Thesis No. 2409

Evaluation of a Methodology for Migration of the Database Layer to the Cloud based on a Research Case Study

Stephan Passow

Course of Study: Computer Science

Examiner: Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Supervisor: Steve Strauch
Karolina Vukojevic
Commenced: November 29, 2012
Completed: April 24, 2013

CR-Classification: C.2.4, D.2.11, H.2.1, H.3.4, H.4.2

Abstract

Cloud computing has the advantage of converting capital expenses into operational costs. This student thesis evaluates a phase driven methodology for migration of the database layer to the Cloud based on a research case study. In addition, a Cloud data migration tool is evaluated. We build an evaluation design for the methodology and the tool. Based on the methodology and with the help of the Cloud data migration tool, an application from the e-Science field is migrated to Amazon Web Services. During the migration project we collect data and document all shortcomings concerning the methodology and the tool. This evaluation data is afterwards analysed, to see how well the methodology and the tool support the migration. The results of the analysis lead to improvement suggestions on the methodology and the Cloud data migration tool.

Contents

List of Figures

List of Tables

Listings

1 Introduction

From an economic point of view Cloud computing has the clear advantage of converting capital expenses into operational costs [14]. There are two scenarios how to take advantage of Cloud computing. The first one takes an existing application and moves it to the Cloud. The second one uses Cloud technologies from the beginning to design and build the application.

In order to create flexible and reusable applications, they are built using a n-tier architecture. In such a n-tier architecture application developers can modify a specific layer or they can add a new layer in between existing layers to add new functionality. They do not have to modify the whole application. The most used n-tier architecture is the three layer architecture which comprises a presentation layer, a business logic layer and a data layer. The presentation layer encapsulates the application-users interactions, the business layer encapsulates the business logic and the data layer encapsulates data storage. The data layer is in turn subdivided into the Data Access Layer (DAL) and Database Layer (DBL). The DAL is responsible for the data access functionality, while the DBL realises data persistence and data manipulation [46].

Right now there is only little support for migrating data into the Cloud and between Cloud services. Bachmann developed a methodology for the migration of the DBL to the Cloud in his diploma thesis [15]. The methodology covers state of the art data migration techniques, different Cloud hosting solutions and Cloud data patterns. This work presents the evaluation of the developed methodology based on a research case study.

1.1 Motivating Scenario

In this section we describe the scenario, which is used for evaluating the methodology for migration of the DBL to the Cloud.

In the scientific domain there are several requirements for applications. For example the processing of huge amounts of data, the need for scalability due to computing-intensive tasks and the reproducibility of the results [41]. Leymann et al. identified distributed systems as a good choice for the practical use in scientific applications, because they fulfill a lot of the requirements [41].

Scientists follow a trial-and-error approach, when they create experiments. Sometimes they need to repeat parts of the experiments with new parameters in order to gain better results. Furthermore, they need to adapt running workflows so as to append new activities. Therefore, the classic workflow technology that has existed in the business world for a long time, cannot be used because of its rigidness. In order to overcome this obstacle, scientists from the

Stuttgart Research Center for Simulation Technology and Cluster of Excellence "Simulation Technology"[1] have developed the so called *Model-as-you-go* approach [43]. This concept combines the modelling, execution, monitoring and adaptation life cycle phases of workflows to fit the needs of scientists. The *Model-as-you-go* approach is implemented in the SimTech prototype. Right now the SimTech prototype consists of different components that are running locally on one machine. The components use Web service technology and message queuing to communicate. This simplifies the distribution of the prototype along different machines. A more detailed description of the SimTech prototype can be found in Section 2.3 .

In this work parts of the prototype are migrated to the Cloud temporarily. The scientific workflow for the simulation is run in the Cloud to benefit from *pay-per-use* and *rapid scalability*. Only the resources that are used during the execution of the workflow have to be paied. After the simulation the results are migrated back to the local machine. We assume that changes of the simulation workflow have no impact on the data required for the simulation. More precisely, there is no need for additional data migration in case of changes of the simulation workflow during runtime in the Cloud.

1.2 Scope of Work

This student thesis has the goal to evaluate the methodology for migration of the database layer to the Cloud developed by Bachmann [15]. The evaluation is based on a research case study which originates from the Stuttgart Research Centre for Simulation Technology and Cluster of Excellence "Simulation Technology". We focus on the migration scenario *Cloud burst*, where application data are moved to the Cloud for off-loading of peak loads [45]. The scientist prepares the simulation on-premise. To be more exact, he models the workflow using the SimTech Workflow Management System (SWMS) Prototype on a local machine. In the rest of the student thesis we call the SWMS Prototype only SimTech prototype. The simulation is run in the Cloud to draw advantages like *rapid scalability* and *pay-per-use*. After the simulation the components are migrated back manually. Instead of migrating the DBL of the SimTech prototype to the Cloud only, also parts of the business logic are migrated. Without the migration of the business logic the data for the simulation has to be transferred from the local machine to the Cloud storage solution. Since data in scientific simulations can be enormous, the transfer over the network becomes the bottleneck for the simulation. Especially with respect to latency and network errors.

Other migration scenarios are not taken into consideration.

Furthermore, Bachmann developed a tool that supports the migration of the data layer to the Cloud (see Section 2.6). The focus for the evaluation of the tool is on the suitability for use.

[1] Stuttgart Research Centre for Simulation Technology and Cluster of Excellence "Simulation Technology": http://www.simtech.uni-stuttgart.de

1.3 Research Design

In order to get a better understanding of the structure of the thesis and the contents of the different chapters, we give a short overview of our research design. These are the main components:

- **Migration methodology and migration tool of Bachmann:** These are the objects of interest.
- **Metrics:** In order to be able to collect data for the evaluation we need metrics.
- **SimTech prototype:** This is the object to which the migration methodology and tool is applied.
- **Web services:** In order to migrate the prototype we need real services. We use services from AWS.

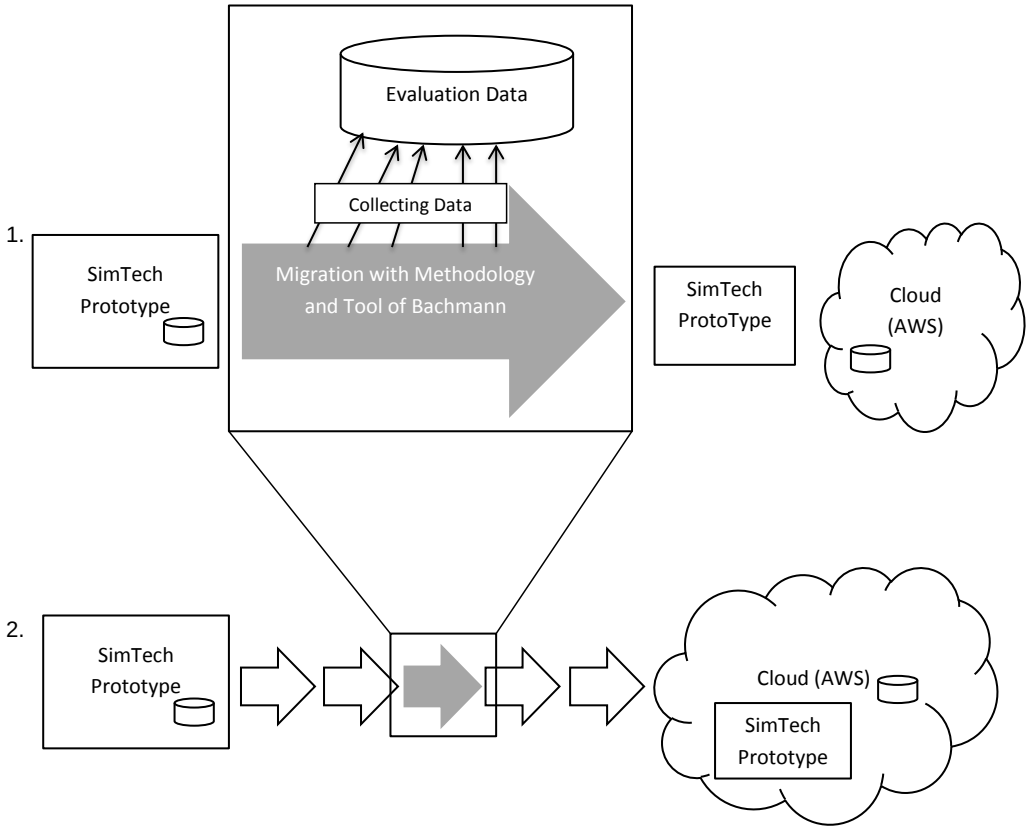

Figure 1.1: Overview Research Design

The evaluation is done by using a case study as research method. We apply the methodology and the tool of Bachmann in a typical scenario. The big picture of our research design is shown in Figure 1.1. In the first step we apply the migration methodology of Bachmann to the DBL of the SimTech prototype. We use a Cloud hosting solution of AWS. During the

migration process we collect data for the evaluation, which is analysed later. In the second step we migrate the whole SimTech prototype. Since the migration methodology of Bachmann focuses on the migration of the DBL, there is need for a holistic approach. AWS also delivers such a holistic migration methodology. The AWS strategy is phase driven. We combine both migration methodologies in such a way, that we replace the data migration phase of the AWS model with the migration methodology of Bachmann. This helps us to evaluate, how well Bachmanns methodology fits into an integrated Cloud migration scenario. Furthermore, we gain knowledge about the ability of the methodology and tool to support the adaptation of the architecture of the whole application.

1.4 Outline

In order to accomplish the given goals and to draft future tasks on the topic the work is structured as followed:

- **Chapter 2 – Background:** This chapter gives an overview about Cloud computing and the SimTech prototype with its architecture. Moreover, the Cloud data migration tool and the migration methodology of Bachmann are described. Additionally, the AWS technologies used for the migration and the AWS migration approach are illustrated.

- **Chapter 3 – Related Work:** An overview of existing approaches and state of research in the fields of software processes and tool evaluation are presented. Furthermore, a general view of evaluation is given and different types of data in evaluation are discussed. Additionally, we position our research to each related work.

- **Chapter 4 – Evaluation Design:** This chapter is based strongly on the lessons learned of the chapter related work. We describe the different components of the evaluation design like focus of the evaluation or the kind of data which should be collected. Beyond, there is argumentation for each of our decisions for the evaluation design.

- **Chapter 5 – Evaluation Data Collection and Processing:** In this chapter we perform the actual migration to the Cloud. On the basis of the chapter evaluation design, we collect the data for the evaluation. Furthermore, we process parts of the data.

- **Chapter 6 – Discussion and Lessons Learned:** Challenges that have emerged during the evaluation are explained. Finally, the collected data is analysed. Improvement suggestions on the methodology and on the Cloud data migration tool of Bachmann are discussed.

- **Chapter 7 – Outcome and Future Work:** The last chapter summarizes the outcomes of this work. Additionally, further tasks related to this student thesis are presented.

1.5 Definitions and Conventions

This section contains a list of abbreviations used in this student thesis.

AWS	Amazon Web Services
AMI	Amazon Machine Image
BPEL	Web Services Business Process Execution Language 2.0
CSI	Continual Service Improvement
CMMI	Capability Maturity Model Integration
DAL	Data Access Layer
DB	Database
DBL	Database Layer
DeGEval	Gesellschaft für Evaluation
EC2	Elastic Compute Cloud
GUI	Graphical User Interface
IaaS	Infrastructure-as-a-Service
IEC	International Electronical Commission
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
ITSM	IT Service Management
NIST	National Institute of Standards and Technology
ODE	Orchestration Director Engine
PaaS	Platform-as-a-Service
PGF	Plugable Framework
RDS	Relational Database Service
SaaS	Software-as-a-Service
SDLC	Systems Development Life Cycle
SEI	Software Engineering Institute
SWMS	SimTech Workflow Management System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

WSDL Web Services Description Language

2 Background

2.1 Application Layers

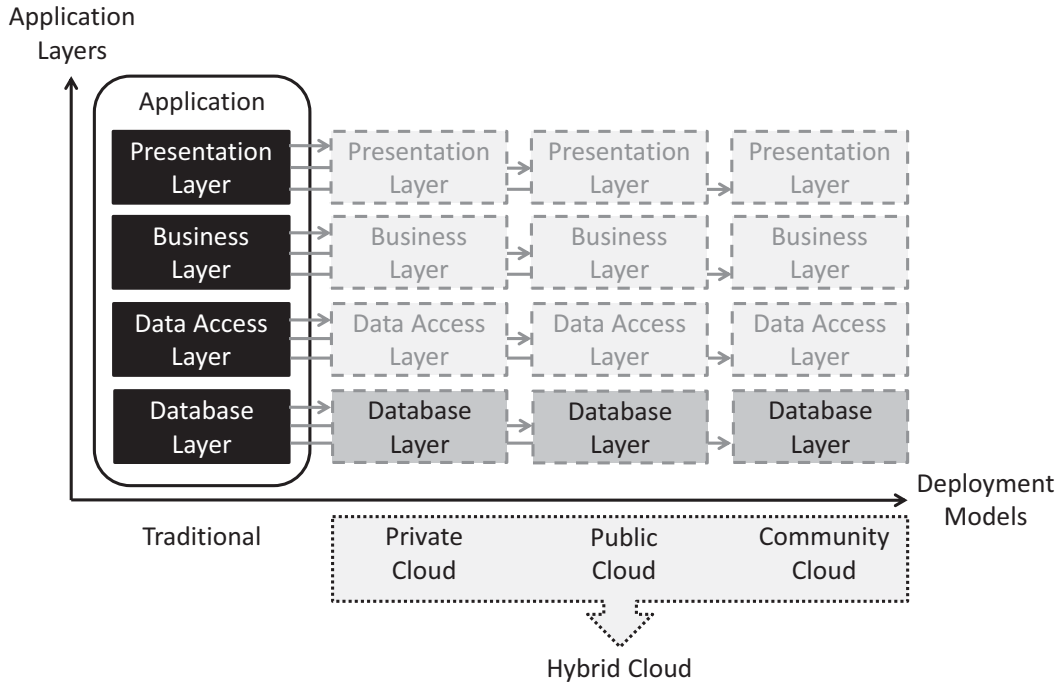

Figure 2.1: Overview of Cloud Deployment Models and Application Layers [46]

Applications usually consist of three layers:

- The *presentation layer* is on top and treats the interaction with the user.
- The *business layer* encapsulates the business logic.
- The *data layer* stores data. It can be divided in the DAL and DBL, where the DAL encapsulates the data access functionality. Data persistence and data manipulation are encapsulated in the DBL.

If the data layer is subdivided in DAL and DBL the application consists of four layers. Figure 2.1 presents such an architecture.

2.2 Cloud Computing

The National Institute of Standards and Technology (NIST) developed a definition for Cloud computing, which we use in this student thesis [32]. The main characteristics of Cloud computing are *on-demand self-service*, *broad network access*, *resource pooling*, *rapid elasticity* and *measured service*.

Cloud computing consists of three service models:

- **Infrastructure-as-a-Service (IaaS):** This service provides the user with computing power, network resources and data storage.
- **Platform-as-a-Service (PaaS):** This service provides the user with programming languages and libraries to build own applications.
- **Software-as-a-Service (SaaS):** The capability provided to the user are the applications running in the infrastructure of the provider.

Applications can be deployed in four different deployment models: *private Cloud*, *community Cloud*, *public Cloud* and *hybrid Cloud*. A taxonomy of Cloud data hosting solutions can be seen in Figure 2.2. Figure 2.1 additionally shows the different possibilities of outsourcing layers to the Cloud.

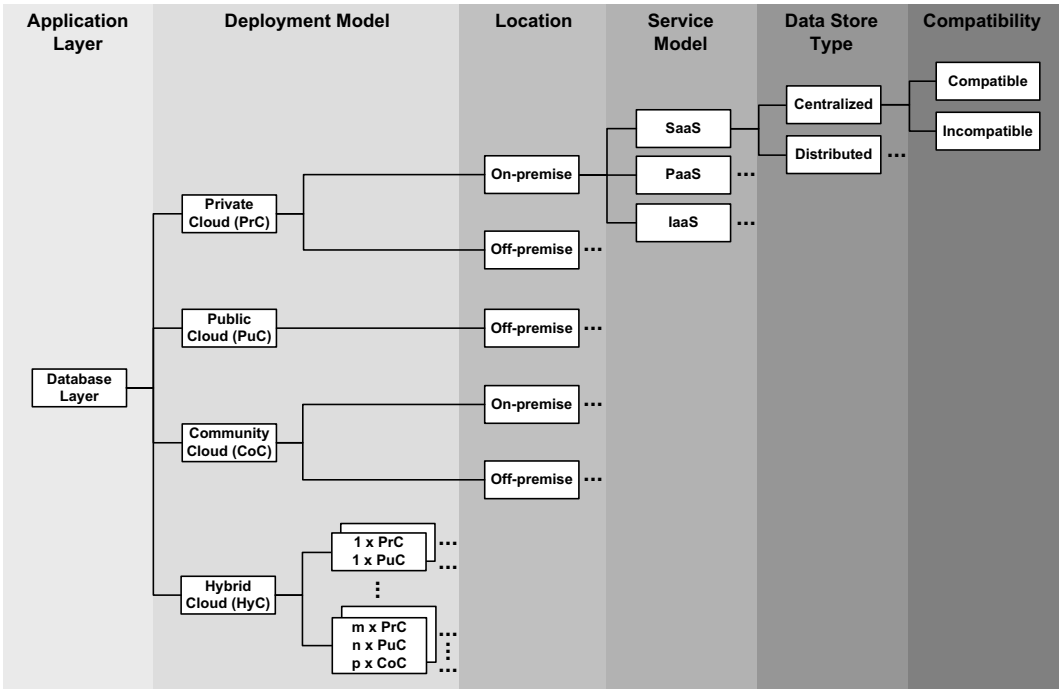

Figure 2.2: Taxonomy of Cloud Data Hosting Solutions [46]

2.3 SimTech Prototype

The SimTech prototype is a central element for the case study. We migrate parts of the prototype to the Cloud and such being the case we present the significant components of the prototype in this section.

2.3.1 Scientific Workflows

The SimTech prototype supports scientific workflows and therefore, we give a short overview of scientific workflows in the following. Leymann et. al identified in [40] three main reasons why workflows are interesting for scientists: "(1) simulations often consist of manual steps that can be automated with the help of workflows; (2) former monolithic (legacy) scientific applications can be executed on multiple machines in a distributed manner; or (3) new simulations/calculations can be created in a graphical manner by modelling workflows." In the motivating scenario we have described the obstacles from classic business workflows in the scientific domain. A solution for those obstacles is the *Model-as-you-go* principle, which was described in the motivating scenario as well.

2.3.2 Architecture

In this section we provide a high-level description of the SimTech prototype.

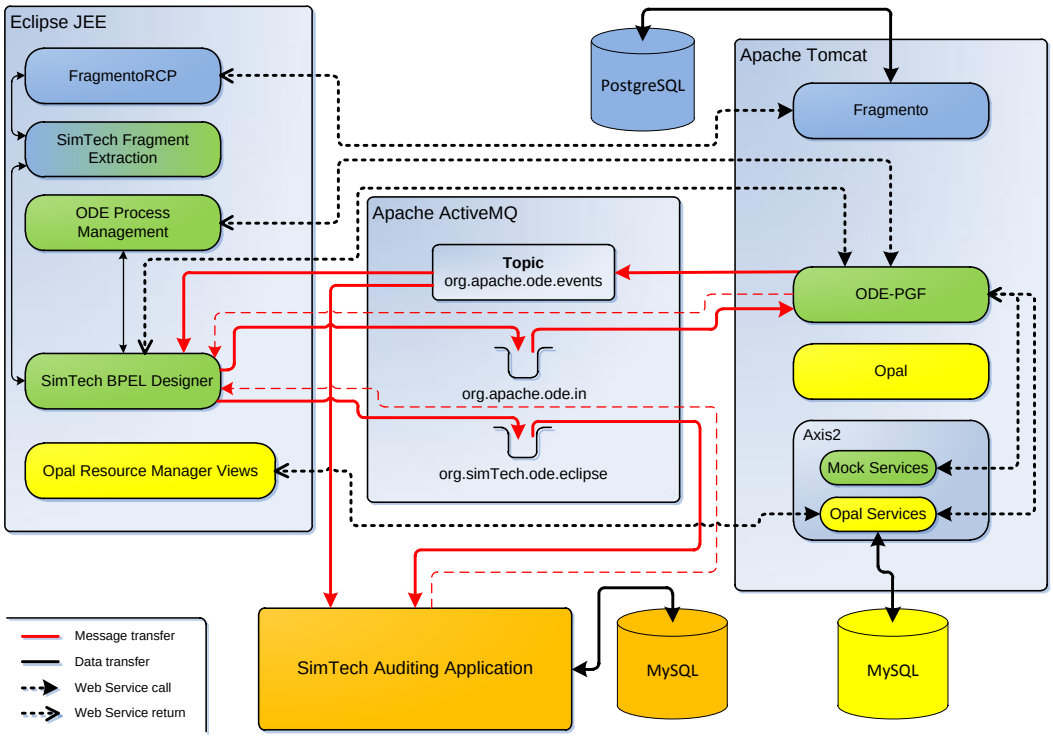

Figure 2.3: Architecture of the SimTech Prototype [25]

The SimTech prototype consists of three main components:

- **Eclipse JEE with the SimTech extensions:** This component uses Web Services Business Process Execution Language 2.0 (BPEL) as workflow language and is based on the Eclipse BPEL Designer as modeling tool [51]. We call this component **Mayflower** (**M**odel-**a**s-**y**ou-go Work**flow** Develop**er**) [42].
- **Apache Tomcat Server:** This component contains an extended Apache Orchestration Director Engine (ODE) as a workflow engine, which we call ODE-Plugable Framework (PGF) in the following [49]. Fragmento is an advanced process fragment library. Its goal is to enable reuse of process fragments in order to speed up and ease the development of process-based applications [36]. Apache Axis2 is a Web services / SOAP / Web Services Description Language (WSDL) engine [50]. Opal is a workflow based solid state simulation application in which structural changes of metallic solids over large periods of time can be simulated [26].
- **SimTech Auditing Application:** It stores execution events for workflow instances published by the engine [42].

The architecture of the SimTech prototype is shown in Figure 2.3. The three components are loosely coupled and can therefore be distributed easier [25]. They are integrated via Web services and messaging. The messaging server (Apache Active MQ) connects the Mayflower instances with the ODE-PGF. The ODE-PGF provides a topic, which any number of Mayflower instances can subscribe to.

A more detailed description of the whole prototype can be found in [25, 42]. Furthermore, there exist several demonstration videos of the SimTech prototype online. They can be found here: `http://www.iaas.uni-stuttgart.de/institut/ehemalige/sonntag/videos_de.html`.

2.4 Components for Migration

In this section we have a closer look at the components of the SimTech prototype that are migrated to the Cloud.

As mentioned before, the Opal services are a workflow based solid state simulation application in which structural changes of metallic solids over large periods of time can be simulated [26]. Fist of all, we provide an overview of the different kinds of data that are required for a simulation. In Figure 2.4 the Graphical User Interface (GUI) for entering simulation parameters and a simplified outline of the file system structure of the Tomcat installation is shown. The real structure and the names of the folders differ from this outline. The Opal services require an energy configuration for the simulation. The user specifies the file path in the GUI. The path is relative to the installation directory of Tomcat and points to one energy configuration file (see Figure 2.4 red arrow marked with 1). In addition a file that defines the structure of the crystal lattice is required for a simulation (see Figure 2.4 red arrow marked with 2). At the beginning of a simulation a new folder in the file system is

created. The folder name is specified in the input field mgmtCtx (see Figure 2.4 blue arrow). In the next step the lattice file is copied in the newly created directory for the simulation. Then the *opal.in* file is generated. It is dependent on the selected energy configuration and other simulation parameters that can be entered in the GUI. After this, the Opal manager starts the BPEL process for the simulation. During the simulation several application data are created. The folder *Simulation Data* contains intermediate results of the algorithm, whereas the folder *Analysis of Snapshots* contains all files created during analysis of snapshots. The folder *Visualization* contains all files generated in the context of the visualization.

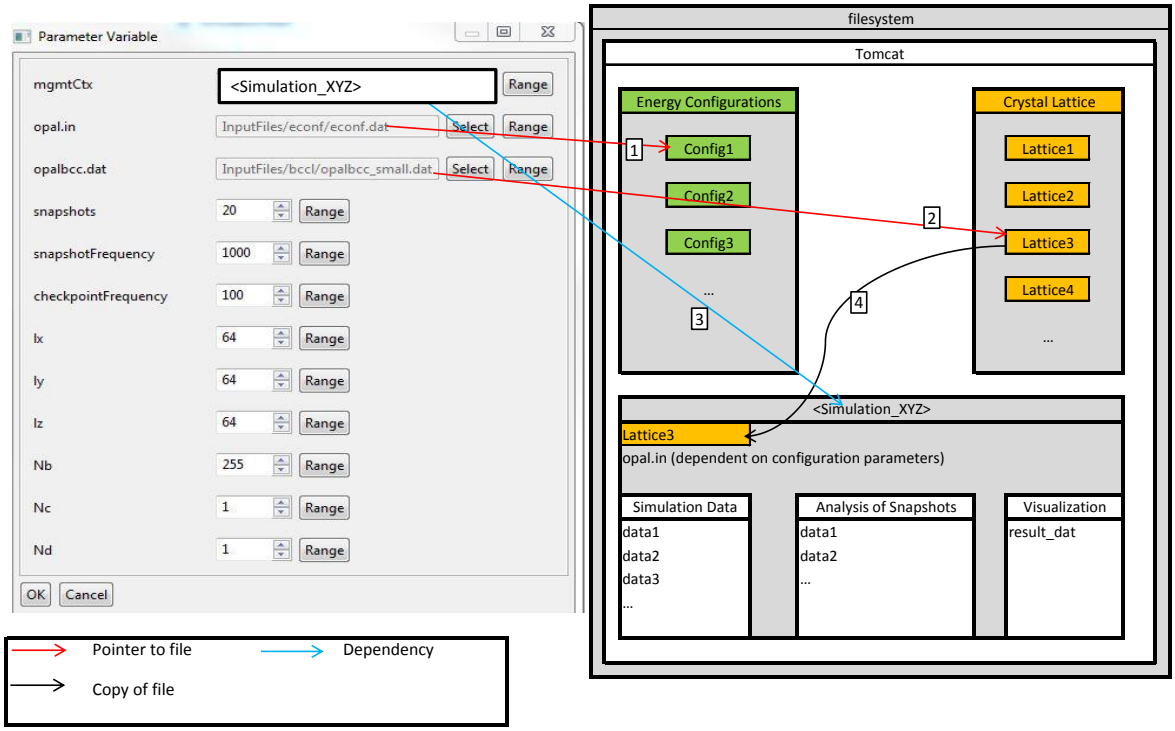

Figure 2.4: Overview Opal Simulation Data

After the description of the GUI and main artefacts of the Opal services, we focus now on the MySQL database, which is migrated to the Cloud. In order to develop a deeper understanding of the data and the database queries, we describe both the resource manager and the Opal manager. The information in the following two sections is taken from [27] and the related source code.

Resource Manager

The Opal services are very CPU-intensive. Hence, there is need for a coordination instance, which regulates the quantity of parallel simulations. By reason of modularization the coordination logic is stored in a single component, the resource manager, and is not embedded in the simulation workflows. During a simulation different kinds of data are created. This data is collected by the simulation workflow and e.g. passed to the Web services for the

visualization. BPEL is not able to directly access data storage of an operating system, but only able to interact with Web services. The resource manager also provides a Web service interface for reading and writing operations on the data storage. In avoidance of losing all registered servers, Web services and the actual endpoints after a restart of the resource manager, this information needs to be persisted in the MySQL database. The resource manager does not query the database often during a simulation, since he stores the data mainly for backup at the beginning of a simulation. Typical SQL queries are shown in Listing 2.1.

```
1  INSERT INTO rm_servers (server_url, server_cores, server_isactive);
2
3  SELECT * FROM rm_managementcontexts;
4
5  SELECT ctx_id FROM rm_managementcontexts WHERE ctx_id = X;
6
7  DELETE FROM rm_managementcontexts WHERE ctx_id = Y;
8
9  SELECT sep.server_id, sep.service_name, sep.sep_epr, sep.sep_req_cores, sep.
       sep_isactive
10 FROM rm_serviceendpoints as sep, rm_servers as s
11 WHERE sep.server_id = s.server_id AND server_url= Z;
```

Listing 2.1: Typical SQL Queries of the Resource Manager

Opal Manager

The Opal manager connects the user with the ODE. In order to ease the creation of simulations, energy configurations and initial crystal lattice should be saved in a single point, which is accessible by all users. This information is stored in the data storage of the operating system. Therefore, the Opal manager needs to interact with the resource manager to access this information. Later on this configuration parameters can be reused by others. Furthermore, the Opal manager creates entries in the MySQL database for every simulation. The current status of the simulation and all starting parameters are stored in the MySQL database. The SQL queries got the same complexity like the ones of the resource manager (see Listing 2.1).

2.5 Migration Methodology of Bachmann

In his diploma thesis Bachmann identified ten migration scenarios [15]. We focus on the scenario **Cloud burst!** (**Cloud burst!**) because it fits best for the case study. Bachmann based his methodology on the methodology of T. Laszewski et al. which can be found in [31]. A phase of the migration methodology of Bachmann consists of a description of the main

objective of the phase, a condition for occurrence, an exit condition and a description of the detailed steps [15].

In the following we give a short summary of the description and the most important steps for each phase.

2.5.1 Assessment

In the assessment phase the information for the project-management is collected, e.g. cost assessment.

The following are the most important steps in this phase:

1. Identification of drivers for migration [31]
2. Inventory of current environment [31]
3. Identification of requirements for the target database [31]
4. Identification of migration tools and hardware [31]
5. Identification of the migration scenario with its specific characteristics [15]
6. Project management [31]

2.5.2 Analysis and Design

The main goal of the analysis and design phase is to identify and describe the implementation details on the target database.

The following are the most important steps in this phase:

1. Selection of the target database and registration at Cloud service providers to verify the requirements [15].
2. Planning of the data cleansing [24]
3. Planning of changes on the different data types [31]
4. Identification of effects and planning of changes on higher levels of the application [31]
5. Planning of changes in the system landscape [31]

2.5.3 Migration

The realization of the plans created in the analyse and design phase is the main goal of the migration phase.

The following are the most important steps in this phase [31]:

1. Migration of the data types
2. Adaptation of the higher levels
3. Adaptation of adjacent systems

2.5.4 Testing

After the migration phase and the adjustments the success of the migration has to be evaluated [15]. The focus of the testing is on the functionality.

The following are the most important steps in this phase [15]:

1. Test of the system
2. Adjustment of the system (if test cases fail)
3. Test of the adjacent systems
4. Adjustment of the adjacent systems (if test cases fail)

2.5.5 Optimization

After the functional testing in the testing phase, the system has to be checked for poor performance.

The following are the most important steps in this phase:

1. Installation of the adopted and migrated system on the target infrastructure and simulation of operation [15]
2. Optimization of the system, e.g. hardware sizing or index optimization [31]
3. Test of the optimizations [15]

2.5.6 Deployment

After the functional testing (testing phase) and the non-functional testing (optimization phase) the whole system can go live in the deployment phase [31].

The following are the most important steps in this phase [31]:

1. Hardware configuration (storage, network, cluster, recovery)
2. Software installation and configuration (schema, users, rights, groups, connection)
3. Data import and index creation
4. Test of the backup and recovery scripts and processes

2.5.7 Post-Production Support

In order to troubleshoot any issues after the deployment it is common to support the users with personnel who where involved in the migration process. For instance there can emerge problems with the new environment (functional and non-functional problems). [31]

2.6 Cloud Data Migration Tool

The Cloud data migration tool is based on questionnaires. It helps to identify the migration scenario with its characteristics, the Cloud data solution and required Cloud data patterns [15]. Moreover, the tool limits the number of Cloud hosting solutions and identifies possible conflicts for the migration [15]. Another important feature of the tool is the providing of suggestions and help with respect to the adaptation of the application architecture according to the choice of the Cloud data hosting solution. Furthermore, it provides information about potential corresponding shortcomings.

Analyses, specification and design of the Cloud data migration tool are described in [15]. Here we only give a short overview of the questionnaires, the tool is based on.

- **Scenario Identification:** In the case study we only focus on the scenario Cloud burst, but other migration projects might need other scenarios like outsourcing of the DBL or using NoSQL, BLOB data storage. In addition, a specific migration scenario can consist of a combination of different scenarios.
- **Cloud Data Hosting Solution Identification:** In this part the user can choose e.g. the deployment model or service model.
- **Description of the Source System:** In order to detect conflicts between source and target system, the tool requires information about the source system.

- **Identification of Cloud Data Hosting Solutions:** After the user has answered the questions in the previous sections, the tool provides a list of matching Cloud data stores. If a Cloud data store lacks required functionalities, this might be compensated with Cloud data patterns. The tool also provides the Cloud data patterns if needed.

2.7 Amazon Web Services

There exist several Cloud providers on the market. We choose AWS as Cloud hosting solution, because they have an education program. Educators, academic researchers and students can apply to obtain free usage credits for their projects [10]. Furthermore, AWS is supported by the migration tool of Bachmann. According to our inquiry, AWS in general is well documented. Therefore, we hope to find helpful information, when we have some issues with the migration.

2.7.1 Migration Methodology

In Section 1.3 we gave on overview of our research design. In the second step we want to fit in the migration methodology of Bachmann into the migration methodology of AWS. Therefore, we present the basic concept of the AWS approach in the following.

The AWS paper addresses business users. It should help them to define a migration strategy for their company. This includes steps, techniques and methodologies for moving existing enterprise applications to the AWS Cloud. [52]

The phase driven approach is shown in Figure 2.5.

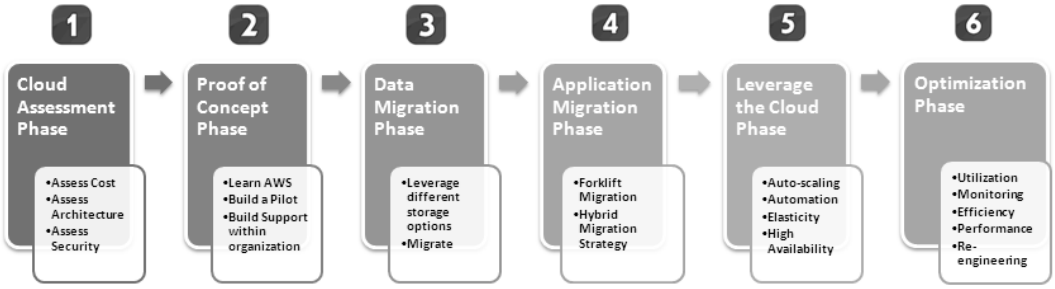

Figure 2.5: The Phase Driven Approach to Cloud Migration from AWS [52]

It consists of six phases that are outlined in short [52]:

- **Cloud Assessment Phase**: Costs, architecture and security are assessed.
- **Proof of Concept Phase**: A pilot is built in order to validate the technology.
- **Data Migration Phase**: A storage solution is selected and the data is moved to the Cloud.

- **Application Migration Phase**: A migration strategy is selected. An Amazon Machine Image (AMI) is created for each component [7].
- **Leverage the Cloud Phase**: Elasticity and Systems Development Life Cycle (SDLC) are automated.
- **Optimization Phase**: Efficiency is improved and usage based on demand is optimized.

A more detailed explanation of the phases can be seen in Figure 2.6.

Phases	Benefits
Cloud Assessment • Financial Assessment (TCO calculation) • Security and Compliance Assessment • Technical Assessment (Classify application types) • Identify tools that can be reused and the tools that need to be built • Migrate licensed products • Create a plan and measure success	Business case for migration (Lower TCO, faster time to market, higher flexibility & agility, scalability + elasticity) Identify gaps between your current traditional legacy architecture and next - generation cloud architecture
Proof of Concept • Get your feet wet with AWS • Build a pilot and validate the technology • Test existing software in the cloud	Build confidence with various AWS services Mitigate risk by validating critical pieces of your proposed architecture
Moving your Data • Understand different storage options in the AWS cloud • Migrate fileservers to Amazon S3 • Migrate commercial RDBMS to EC2 + EBS • Migrate MySQL to Amazon RDS	Redundancy, Durable Storage, Elastic Scalable Storage Automated Management Backup
Moving your Apps • Forklift migration strategy • Hybrid migration strategy • Build “cloud-aware” layers of code as needed • Create AMIs for each component	Future-proof scaled-out service-oriented elastic architecture
Leveraging the Cloud • Leverage other AWS services • Automate elasticity and SDLC • Harden security • Create dashboard to manage AWS resources • Leverage multiple availability zones	Reduction in CapEx in IT Flexibility and agility Automation and improved productivity Higher Availability (HA)
Optimization • Optimize usage based on demand • Improve efficiency • Implement advanced monitoring and telemetry • Re-engineer your application • Decompose your relational databases	Increased utilization and transformational impact in OpEx Better visibility through advanced monitoring and telemetry

Figure 2.6: Phases of the Migration from AWS [52]

A guide that explains the phases in depth can be found in [52].

2.7.2 Amazon Database Services

So as to migrate the DBL of the SimTech prototype to the Cloud, we need a real Cloud data hosting solution. AWS offers four different products:

- **Amazon RDS**: This is a Web service for setting up, operating and scaling a relational database in the Cloud [9].

- **Amazon DynamoDB**: This is a fully-managed, high performance, NoSQL database service that is easy to operate and scale [3].
- **Amazon ElastiCache**: This is a Web service that offers deployment, operation and scaling of an in-memory cache in the Cloud [6].
- **Amazon Redshift**: This is a fully managed, petabyte-scale data warehouse service in the Cloud [8].

The MySQL database of the Opal services is a relational database and therefore, we use Amazon RDS. It supports the full features and capabilities of a relational database. According to Amazon we need to do the following tasks, when we want to use Amazon RDS [9]:

- Launching a Database (DB) Instance, selecting the DB Engine (MySQL, Oracle or SQL Server), License Type, DB Instance class and storage capacity that meets our needs best.
- Connecting to the DB Instance using our favourite database tool or programming language. Since we have direct access to the native database engine, most tools designed for theses engines should work unmodified with Amazon RDS.
- Via Amazon CloudWatch metrics, we can monitor compute and storage resource utilization of our DB instance. With a few clicks, we can scale the compute and storage resources.

2.7.3 Amazon Computing Services

In the second step of an iteration we want to migrate the whole Opal services. Therefore, we also need computing capacity. Amazon provides four different products:

- **Amazon Elastic Compute Cloud (EC2)**: This is a Web service that provides resizable compute capacity in the Cloud [4].
- **Amazon Elastic MapReduce**: This is a Web service that enables researches, data analysts and developers to easily and cost-effectively process vast amounts of data [5].
- **Auto Scaling**: This is a Web service that allows to automatically scale the Amazon EC2 capacity up or down according to predefined conditions [11].
- **Elastic Load Balancing**: This is a Web service that automatically distributes incoming application traffic across multiple Amazon EC2 instances [12].

We use a case study for the evaluation. In the context of the student thesis the Amazon EC2 instance fits our needs best, because it provides basic computing power. If the SimTech prototype is used on huge amounts of data, we recommend to add Amazon Elastic MapReduce. In addition, we propose also the auto scaling service, when there is unpredictable use. In a real world scenario, elastic load balancing can be useful, if greater fault tolerance in the application is required. Amazon EC2 presents a true virtual computing environment. According to Amazon we need to perform the following tasks, in order to use Amazon EC2 [4]:

- Selecting a pre-configured, templated AMI. We also can create an AMI containing our applications, libraries, data and associated configuration settings.
- Configuring security and network access on our Amazon EC2 instance.
- Choosing instance types, then starting, terminating and monitoring as many instances of our AMI as needed.
- Determining whether we want to run in multiple locations, utilize static IP endpoints or attach persistent block storage to our instances.

3 Related Work

This chapter provides an overview of existing approaches and state of the art in the domain of the thesis. Moreover, we position our work towards the state of the art.

The *Gesellschaft für Evaluation (DeGEval)* defines evaluation as the systematic study of the utility or value of objects [17, 23]. In addition the achieved results, implications or recommendations have to be comprehensible and based on empirical gained qualitative or quantitative data [17]. Additionally evaluations should feature four basic attributes: utility, feasibility, propriety and accuracy [17]. Again Tergan et al. define evaluation as the systematic, target-oriented collection, analysis and assessment of data for the purpose of quality assurance and quality control [48]. Wottawa defines evaluation as the collecting and combining of data with a weighted set of scales that should lead to a comparative or numerical assessment [54]. We picked three definitions that differ from a large variety of existing definitions of evaluation [17]. There exist many definitions, because there is need for evaluation in several fields of application [17]. We did not find an evaluation method that fits perfectly for our purpose. Therefore, we take a stand on other evaluation methods in related areas. With the presentation of existing approaches and lessons learned concerning evaluation, this chapter builds a foundation for the design of our evaluation method, which is presented in Chapter 4.

There exist different empirical investigation methods. They are presented in Section 3.1. A major part of any evaluation is collecting data [21]. Ways of collecting and classifying data are presented in Section 3.2. As we evaluate not only the methodology for migration of the DBL to the Cloud, but also the Cloud data migration tool, we focus on works in the fields of software processes evaluation and tool evaluation. Existing approaches on evaluating software processes can be found in Section 3.3. Strategies for evaluating tools are shown in Section 3.4. Both objects of interest are directly designed for human beings. Therefore, we focus on human-centred evaluation methods.

3.1 Empirical Investigation Methods

Although the title of this student thesis determines case study as the selected method, we want to present an overview of the different empirical investigation techniques. Kitchenham et al. categorize empirical investigation methods in *case study*, *formal experiment* and *survey*. A study is called *formal experiment*, when it involves appropriate levels of replication. Moreover, the experimental subjects and objects have to be chosen at random within the experimental design. A *formal experiment* always requires an experimental hypothesis that is to be evaluated.

Its scale is small and therefore, it is hard to scale up from the laboratory to a real project. In contrast to that a study is called a *case study*, if it focuses on a single project. The advantage of case studies is the applicability to real-world objects. A study is called a *survey*, when it covers many teams and many projects. [30]

The scope of the work is wide-ranging and therefore, a *formal experiment* is not applicable. In our case a *survey* could be designed with many user groups, which evaluate different migration scenarios with different methodologies against the migration methodology of Bachmann. Furthermore, they use different Cloud hosting solutions. In the context of a student thesis a *survey* of this extent is not possible. The *case study*, where we evaluate a typical scenario on our own, fits best.

3.2 Data in Evaluation

On a higher level data can be classified in *qualitative data* and *quantitative data*. *Qualitative data* is expressed in words and it is useful to get a richer understanding of processes and tools [37]. *Quantitative data* is numerical and it is useful for measuring a particular aspect of a process or tool [37]. Nevertheless, *qualitative data* is usually verbally interpreted to increase the understanding of the numerics [38]. In quantitative research authors begin with a hypothesis, which is tested on a large number of cases, using accepted statistical measures [38]. In contrast, a qualitative research starts with a single case, because e.g. of its convenience or interest [38]. Shull et al. are of the opinion that evolving processes or tools need both types of data: Measuring the effectiveness is usually quantitative and feasibility or fit to the environment may be qualitative [37]. Since we do not have a single hypothesis to evaluate, but a *case study*, we consider collecting *qualitative data* as the best approach for our evaluation. Techniques for collecting data in qualitative research are questionnaires, interviews and documentary materials [33]. Since we evaluate the *case study* on our own, we use documentary materials. Additionally we measure the time spent in the different phases of the migration scenario. Time is assigned to *quantitative data*. Finally, we use both types of data, but with a clear focus on *qualitative data*.

3.3 Evaluation of Processes

Bachmann described several state of the art migration scenarios in his diploma thesis. He finally choose to base his methodology on the methodology of Laszweski [31]. Laszweski again based his methodology on the traditional waterfall software development methodology [31]. Therefore, we have strong believe that the methodology developed by Bachmann with its phases assessment, analysis and design etc. builds a convenient basement that is not to be changed. The waterfall model has been around since 1970 and several problems occurred when it was used in practice [39, 35]. The main points of criticism are the inflexibility to changes of stakeholder requirements and the late occurrence of problems since they appear in the testing phase only [35]. Sommerville outlines the need of iteration between the phases of

the model in practical use [39]. Cloud computing is relatively new and we are of the opinion that that there is uncertainty about requirements and goals at the beginning of a Cloud migration project. As a result we analyse the methodology for the need of more iteration and feedback loops.
Shull et al. differentiate global lessons and specific lessons when it comes to evaluation of software processes. Global lessons affect the entire process. The main focus is on validating the overall focus and direction of a process. Specific lessons are more a fine-tuning of the individual steps to increase effectiveness. [37]
We also undertake a fine-tuning of the different phases since we agree with the general phases of the waterfall model. Possible results of the fine-tuning might be providing more information for an intermediate step or adding more intermediate steps if needed.

3.3.1 Standards

There exist several standards for evaluation or improvement of processes in a broader sense. Based on our literature research we present two well known standards in the following.
The *Capability Maturity Model Integration (CMMI)* models of the Software Engineering Institute (SEI) are a collection of best practices that help organizations to improve their processes [19, 47]. CMMI aims on minimizing expenditures of the software development process. Furthermore, it improves predictability of costs, expenditures and time while it improves efficiency of allocation of resources. An CMMI assessment starts with a positioning of the process with its strengths and weaknesses. This builds the basis for defining measures to expand the strong points and to eliminate the weaknesses [22]. CMMI is a very complex framework and therefore, we can not use it in our evaluation. Firstly, we would need to spend more time with CMMI in order to be able to apply it correctly. Secondly, we detected an approach that fits our needs better.
Information Technology Infrastructure Library (ITIL) is a collection of best practices for IT Service Management (ITSM). According to their own declaration "ITIL is the most widely accepted approach to IT service management in the world." [13]
ITIL provides recommendations for service providers on the provision of quality IT services and on the processes and other capabilities needed to support them [44]. ITIL is a modular system. ITIL Continual Service Improvement (CSI) aims on alignment of IT services with changing business needs by identifying and implementing improvements to IT services that support business processes [44]. According to [44] CSI uses a seven-step improvement process, which is shown in Figure 3.1. We can reuse the general structure of the ITIL improvement process. In order to evaluate the methodology and the tool, we also need to define a strategy (phase 1). In our case this strategy is determined as a research case study due to the title of the student thesis. Moreover, we also have to define what to measure and how to gather the data (phase 2). In a live system there is always a trade-off between logging of data and slowing down the system. Since we do not have a productive system, which is used by others, there are no restrictions regarding the point in time or amount of data gathering (phase 3). In addition, most of the data in our research is collected manually. Phase 4 focuses on processing of data. In Section 3.2 we outlined our focus on qualitative data for our evaluation. We develop a standardized template for collecting the data. This helps us to

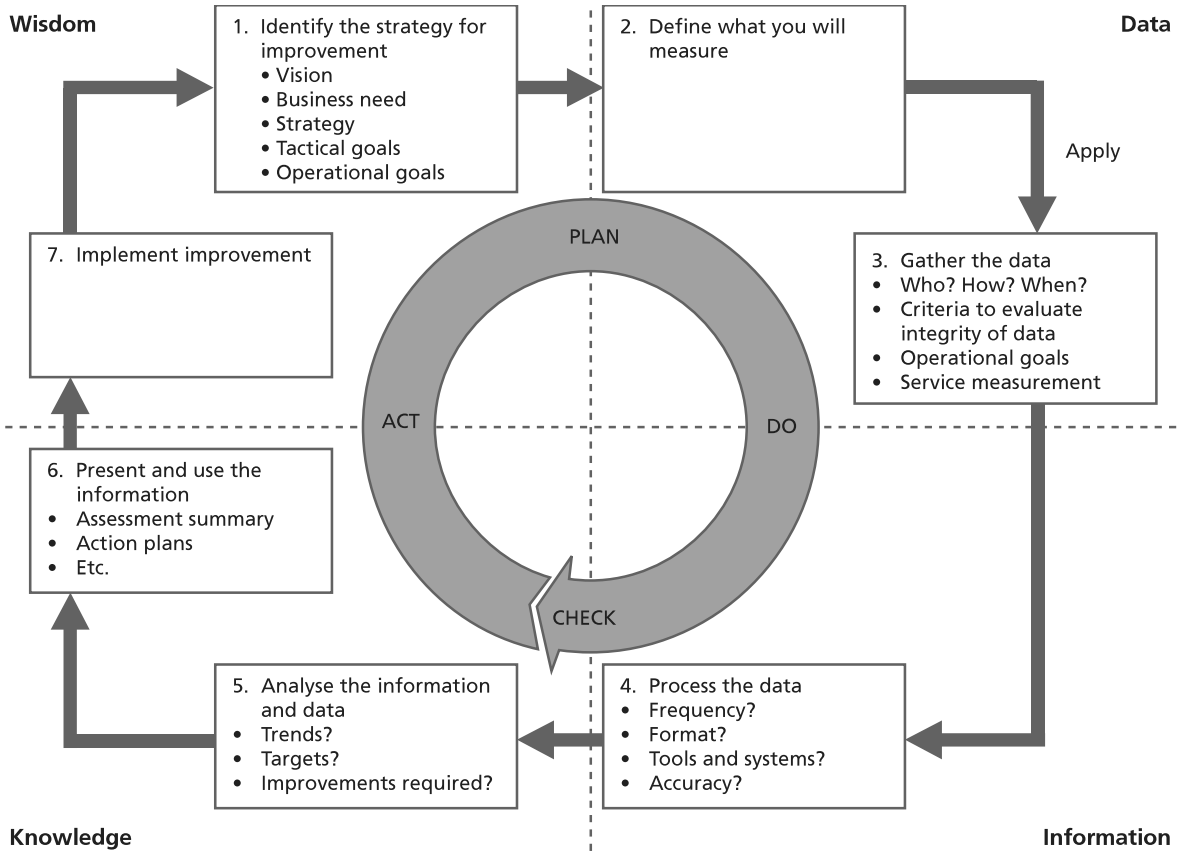

Figure 3.1: ITIL Seven-Step Improvement Process [44]

minimize the time of processing the data. After the migration we also need to analyse the information and data (phase 5). In contrast to phase 6 of the ITIL improvement process we do not have to communicate our results to stakeholders. Certainly, we have to write down the lessons learned. In the end we have to adopt the methodology and the tool (phase 7).

ITIL CSI always needs to be adapted to the specific environment [44]. Even if our research is more technical and the case study is not based in an organization, we can reuse the seven-step improvement process with the adaptations described above.

3.3.2 Metrics

In order to evaluate software processes we need metrics. Galin wrote a book about software quality assurance [20]. He defined metrics for software processes in order to measure their quality. Process metrics can be classified in four categories:

- *Software process quality metrics*
- *Software process timetable metrics*

- *Error removal effectiveness metrics*
- *Software process productivity metrics*

Software process quality metrics involve two measures: software volume and errors counted. Furthermore, errors in the development process can be weighted [20]. The errors are are put in context with the total lines of code in order to calculate the metrics [20]. Since we evaluate a methodology as well as the tool, we do not program lots of software. We use this kind of metric with a little adaptation. All errors during the migration process are documented.
Software process timetable metrics are based on milestones. One method calculates the average delay in completion of milestones [20]. The other approach puts the milestones completed in time in context with the total number of milestones [20]. Because we do not have a project plan, where we defined milestones, we would have to adapt this metrics. We lack experience in migration projects and therefore, we can not estimate the time needed. This makes it impossible to use this kind of metrics.
Error removal effectiveness metrics belong to the support phase. After a period of 6 or 12 months of regular operation this metric can be calculated [20]. The migration methodology of Bachmann also contains a support phase. Since we do not have a productive system in the end of the evaluation, we consider this phase less important. Therefore, we do not use this kind of metric.
Software process productivity metrics aim on human resource productivity in a project as well as on software reuse, because software reuse affects productivity substantially [20]. The human resource productivity can be calculated with 'total working hours invested' divided by 'lines of code' [20]. We build everything from the scratch and cannot reuse any fragments for the migration process. Since we do not program software, we do not write a lot of code. We would have to adopt this metric, but we do not find useful measures to base this metric on. Hence, we do not use this kind of metric.

3.4 Evaluation of Tools

There exist many definitions of software quality. A good overview over software quality models can be found in [2, 16]. We chose to use the ISO/IEC 9126 quality standard developed by the International Organization for Standardization (ISO) and the International Electronical Commission (IEC) [28]. We are aware of the fact that the ISO/ICE 25010[1] is the more extensive successor of the ISO/ICE 9126. ISO/ICE 25000 also addresses compatibility, security, etc. [29]. The ISO/IEC fits our needs, because it contains the software quality characteristics, which we consider most important for the migration tool. The standard consists of four parts:

- Part 1: Quality model
- Part 2: External Metrics
- Part 3: Internal Metrics

[1]This a series of standards on Software Engineering. It is called SQuaRE (Software product Quality Requirements and Evaluation)

- Part 4: Quality in use metrics

The ISO/ICE 9126-1 quality model is presented in Figure 3.2. The grey components are general *characteristics* of software. They are subdivided into *sub-characteristics* (white components), which are decomposed into *attributes*. The *attributes* are not part of the standard, because they differ from each project. Attributes that can be measured during the development process of the software are referred to as internal [18]. The external behaviour can be measured in the testing phase, whereas the quality-in-use metrics are measured from the user's perspective [18].
The main focus of the student thesis is the evaluation of the Cloud data migration methodology. The tool supports the user when he is migrating the DBL to the Cloud (see Section 2.6). Since we can not evaluate all quality characteristics due to lack of time, we focus on the evaluation of the functionality with respect to suitability. Moreover, we consider usability as another important quality aspect in the context of this work. Therefore, we evaluate the tool with focus on understandability and operability, too.

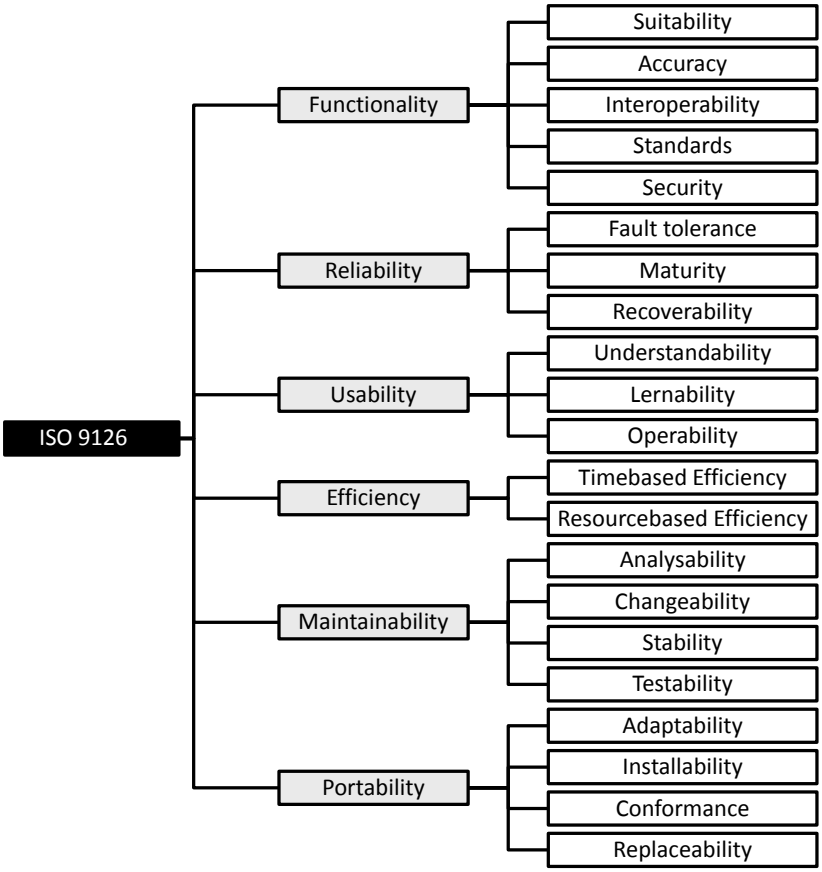

Figure 3.2: The ISO 9126 Quality Model Attribute Tree [1]

4 Evaluation Design

This chapter presents the design of the evaluation, which is based on the previous Chapter 3. We have already positioned our work towards the work of other authors in the domain of the thesis, but now we want to fit those pieces together. We reuse the seven-step improvement process of ITIL, because it is a framework that comprises all necessary activities for our evaluation. Nevertheless, we have to adapt it to our environment and put the process steps in concrete terms. In order to achieve a neat arrangement, we accomplish a consolidation of the process steps, that are linked closely together. The mapping of the process steps to the chapters of this student thesis is shown in Table 4.1. Additionally, we assign the different sections to the process steps.

ITIL step	Chapter	Section(s)
1. Identifiy the strategy for improvement	4	4.1
2. Define what you will measure	4	4.2, 4.3, 4.4, 4.5
3. Gather the data	5	5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.2.1, 5.2.2, 5.2.3, 5.2.4
4. Process the data	5	5.1.5, 5.2.5
5. Analyse the information and data	6	6.1, 6.2, 6.3, 6.4
6. Present and use the information	6	6.1, 6.2, 6.3, 6.4
7. Implement improvement	7	-

Table 4.1: Mapping of ITIL Steps to our Research

As mentioned before, we use *case study* as evaluation method (see Section 3.1). Therefore, we need a migration project, in which we use the methodology and tool of Bachmann. The migration strategy and the components that are migrated, are presented in Section 4.1. Section 4.2 points out the important aspects for the evaluation. Section 4.3 incorporates the concrete approach for collecting evaluation data. In Section 4.4 the structure for the data gathering is illustrated. Section 4.5 commands a view on the processing and analysis of the collected data.

4.1 Components

Initially, only the DBL should be migrated to the Cloud. In our case of application this leads to several shortcomings and therefore, we also migrate the business logic. The simulation workflow needs data for the simulation. Without the migration of the business logic the

data for the simulation has to be transferred from the local machine (hosted traditionally on-premise) to the Cloud (off-premise). Since data in scientific simulations can be enormous, the transfer over the network becomes the bottleneck for the simulation. Especially with respect to latency and network errors. Since expenses of the migration scenario can not be estimated, the migration process is executed in two stages. The first iteration includes the migration of the *Opal services* (see Section 2.4). If there is still enough time for the second iteration, we also migrate the *auditing application*. An iteration consists of two steps (see Section 1.3):

1. Using the migration method and the tool of Bachmann for migrating the DBL
2. Migrating the business logic and the DBL of a whole component to the Cloud using the phase driven methodology of AWS in combination with the methodology and tool of Bachmann

Both migration steps are realized with Cloud services from AWS (see Section 2.7).

4.2 Focus

The following requirements are derived from the formulation of the student thesis and the supervisor of the student thesis.
One key aspect for the evaluation is the quality of the technical migration of the DBL. We need to evaluate, how well the methodology and the tool do support the user in the migration project. In addition, we need to track the amount of errors during the migration. Another important aspect for the evaluation is the support of the methodology and the tool for the adaptation of the architecture. This issue especially gains greater significance in the second step of the migration, where a whole component is migrated to the Cloud. Figure 4.1 illustrates the important matter of the second step. The data migration phase of AWS is replaced by the methodology and tool of Bachmann. Since the methodology and tool cover more than the pure data migration, we have to evaluate, which phases of Bachmann and AWS are overlapping. In addition, the same phases can have different characteristics. We have to point out the differences and common features.

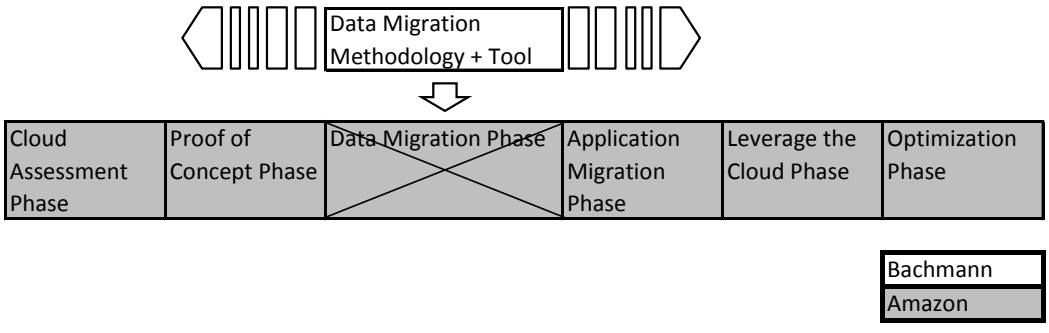

Figure 4.1: Combination of AWS and Bachmann

In Chapter 3 we presented weaknesses of the waterfall model on which the methodology of Bachmann is based on. One point of criticism was the inflexibility to changes of stakeholder requirements. Since there are no stakeholders we can not evaluate this. Another point of criticism was the missing iteration between the phases of the model in practical use. Therefore, we focus on the detection of iterations, when we apply the methodology. Since we use case study as evaluation method, we only utilize the methodology and tool for this purpose. Therefore, the most import software quality attributes for the evaluation of the tool are *understandability*, *operability* and *suitability*. Other aspects are not taken into consideration. More precisely, our evaluation does not cover the whole methodology and the whole tool, e.g. there exist further migration scenarios that are supported.
All challenges and issues during the migration project should lead to continuous improvement and adaptation of the methodology as well as the tool.

4.3 Data Collection Approach

In Section 3.2 we gave an overview of the data classes in evaluation. We focus on *qualitative data*, because qualitative data is useful to get a richer understanding of processes and tools [37]. This is exactly what we aim at. We want to evaluate the methodology and the tool. Therefore, we need to document all shortcomings and problems (we conflate them under error) during the case study in order to improve the methodology and the tool. In the domain of this evaluation an error can have several meanings:

- *traditional error:* e.g. there is an error during the migration of data or the migration tool crashes
- *understanding problem:* e.g. tool and methodology give us a recommendation and we are in doubt what exactly to do
- *time problem:* there is a challenge and we spend a lot of time finding a solution

Our approach for documenting a single error is shown in Table 4.2.

An ID is compound of a letter and an integer. We use a running ID, because this helps us to get a better understanding of the chronology of the errors. If the error occurs in step 1 of an iteration the letter is B (**B**achmann). If the error occurs in step 2 of an iteration the letter is A (**A**mazon Web Services). In addition, each error has a name (e.g. *migration-error*).
The class specifies the belonging of the error. It can take the values *methodology*, *tool* or *others*. If the class is tool, the corresponding software quality attribute has to be added in brackets. Due to our focus (see Section 4.2) we can choose one of the following attributes: *understandability*, *operability* and *suitability*. Since there is no hard limit between the different software quality attributes, it can be hard to categorize the errors. Moreover, a shortcoming can sometimes be assigned to different software quality attributes.
The severity of an error can take one of the following values: *low*, *middle*, *high* or *critical*. Severity can be seen from different perspectives. E.g. severity as the difficulty to fix the error or severity as threat to a successful migration. We score the severity with respect to the impact on the migration result. A wrong database Uniform Resource Locator (URL) is easy to fix,

Property	Value
ID	B17
Name	migration-methodology-unclear
Class	methodology
Severity	high
Description	We did not understand what to do in step *X*.
Error Handling	We were searching in the diploma thesis of Bachmann for more information concerning *X*.
Solution	In *X* it needs to be checked if compound primary keys exist in the database.
Adaptation	More details need to be added to step *X*.

Table 4.2: Error Example

but critical for the migration project. Therefore, we rate an database URL error with severity *high*. This helps us during the discussion in Chapter 6 to prioritize the different possible adaptations.
In the description we write as much information as needed to understand the error. A critical error is a special kind of error. It is an error, that leads to a step back in a prior phase (see Section 4.5).
The error handling describes our approach to find a solution for the problem. If we can solve the problem, we describe the solution. If we can't work something out directly, the solution contains our substitute solution. Sometimes it is not possible to specify an error handling or a solution. E.g. if there is a problem concerning the understandability (usability) of the tool.
Adaptation is an important property. This field contains the required adaptations to troubleshoot the tool or methodology. In Chapter 6 we discuss the required adaptations. If we already have them documented together with the error, we can reconstruct their origin better.

Even if we focus on the *qualitative data*, we also use *quantitative data*: We measure the time that we spent in each phase. This data helps us to determine the most time consuming phases.

4.4 General Structure

In Section 4.1 we gave an overview of our research design. In iteration 1 we use the methodology of Bachmann and in iteration 2 we use a combination of the AWS migration methodology and Bachmann's methodology. For each phase of the methodologies, we document all occurring errors within the error template shown in Table 4.2. This helps us to minimize step four *process the data* of the ITIL improvement process, since we already have data with context and not just raw data. The following two sections describe the relevant phases of the methodologies with respect to the case study.

4.4.1 Iteration 1 - Bachmann

Since we do not have users and a productive system, we do not use the phases *deployment* and *post-production support*. Furthermore, there is no need for the *optimization* phase, because we do not have high workload on the prototype. In order to structure the occurring errors, we collect them for the following phases:

- Assessment
- Analysis and Design
- Migration
- Test

Additionally, we provide a brief conclusion for each phase of the methodology.

4.4.2 Iteration 2 - AWS and Bachmann

We do not apply all phases of the AWS migration methodology to the SimTech prototype. The phase *leveraging the Cloud* focuses on automating elasticity and security hardening [52]. In our case this is less relevant for the case study. In addition, we leave out the *optimization* phase, since we do not have high workload on the prototype. In order to structure the occurring errors, we collect them for the following phases:

- Cloud Assessment
- Proof of Concept
- Data Migration Phase
- Application Migration Phase

Additionally, we provide a brief conclusion for each phase of the methodology.

4.5 Data Processing Approach

In Section 4.2 we outlined, that we also focus on the detection of iterations, when we apply the methodology. Therefore, we visualize the appearance of errors that lead to a return to a prior phase of the methodology. A possible iteration of the different phases during a migration scenario is shown in Figure 4.2. Bachmann defined matching conditions for the phases [15]. If we meet all the requirements of a phase, we can enter it. Following the methodology we should not have to step back in a prior phase. Nevertheless, it is feasible that we have to iterate between the phases. If we collect the errors by sequence only, the evaluation is missing structure. If we collect the errors only for each phase, we are missing the chronology. Therefore, we use the grouping of errors for each phase in combination with the visualization shown in Figure 4.2. A step in time is marked green, when no critical errors occurred. A critical error is an error, that leads to a step back in a prior phase. If such an error occurs, the

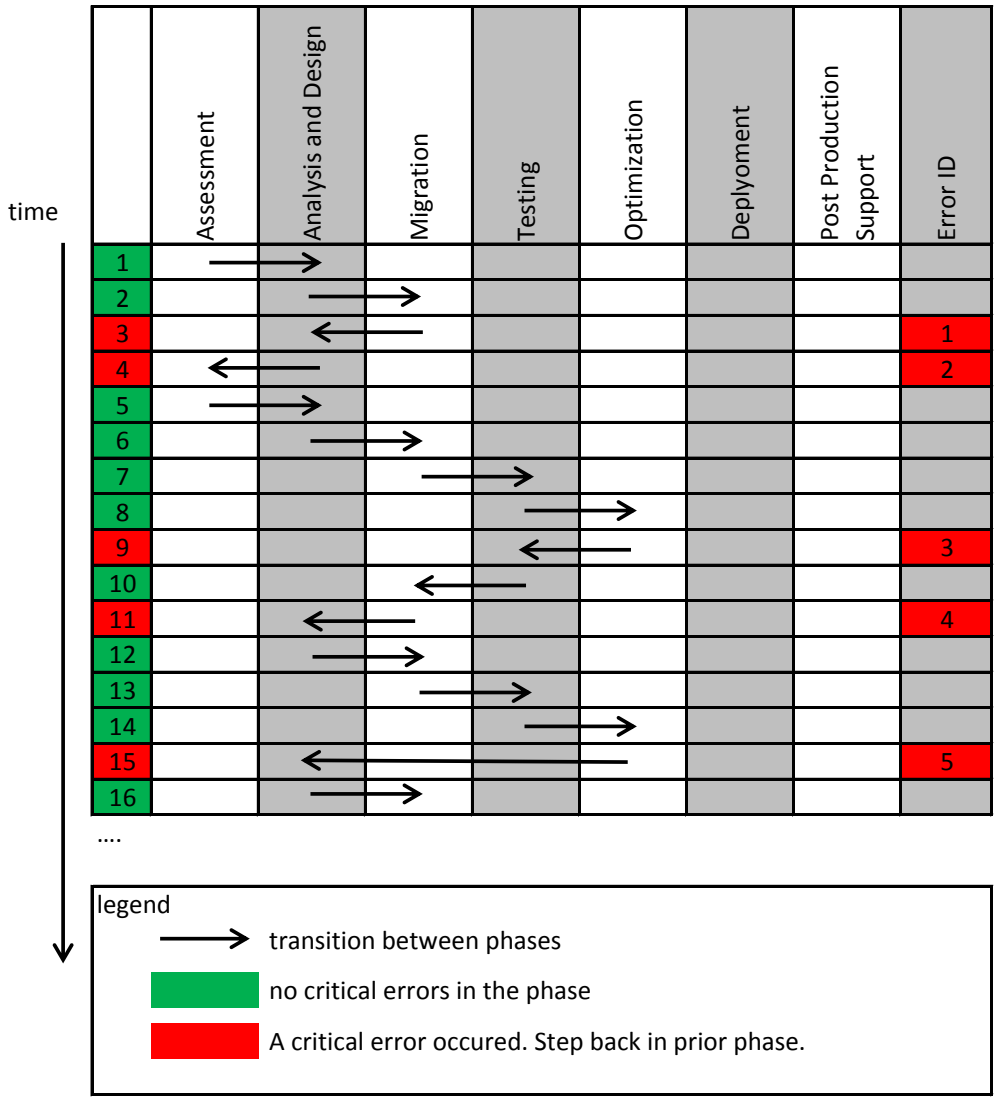

Figure 4.2: Iteration Between Phases of the Methodology During a Migration Project

step in time gets marked red. Dependent on the origin of the error, we step back in a prior phase. The error with ID 5 leads to a step back from the optimization phase into the analysis and design phase (see Figure 4.2). An arrow marks the transitions between phases.

In Section 4.3 we described, that we measure the time spent in the different phases. Due to possible iterations we have to summarize the time for each phase at the end of the migration project. A possible distribution of time is shown in Figure 4.3. This should help us to detect the most time consuming phases. As a consequence, we could support the user with more detailed information for this phase or add additional intermediate steps.

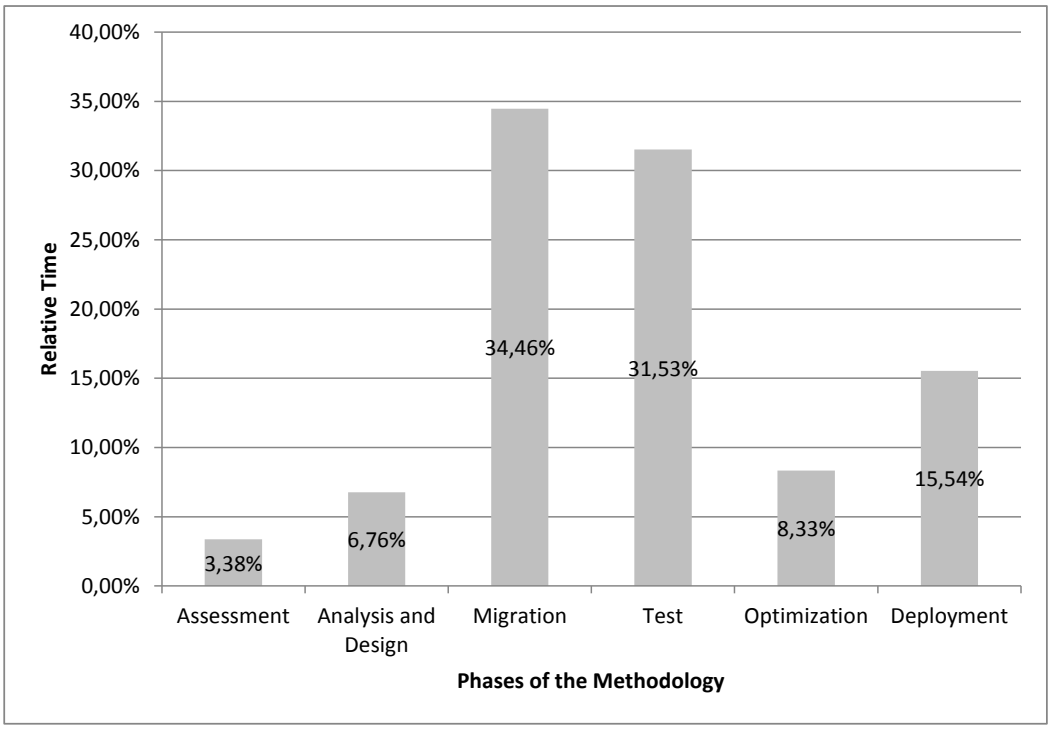

Figure 4.3: Relative Time Spent in Each Phase

5 Evaluation Data Collection and Processing

In Figure 1.1 we presented the overview of our research design. Following this, we apply the migration methodology and the tool of Bachmann to the MySQL database of the Opal services in Section 5.1. We only apply the phases *assessment* to *test* to the SimTech prototype (see Section 4.4.1). The phases *deyployment* and *post-production support* are not part of the case study, because we do not have a productive system. Due to the characteristics of the MySQL database, which is mainly used for storing configuration data, we do not have to apply the phase *optimization*. In Section 5.2 we migrate whole parts of the SimTech prototype to the Cloud. Therefore, we use the migration methodology of AWS in combination with the methodology and the tool of Bachmann. In this section we pay special attention to the ability of the methodology and the tool to support the adaptation of the architecture of the application. In Chapter 4 we presented the evaluation design. We defined, which data we want to collect for the evaluation and how we structure the collection. In addition to that, we describe our actions and the results of the intermediate steps in the different phases to make the whole evaluation more comprehensible.
The evaluation data, that we collect in this chapter, is analysed and discussed in Chapter 6.

After the completion of the first iteration (Opal services), we do not perform the migration of the auditing application. We do not expect to gain new relevant insights concerning the evaluation of the methodology and the tool by migrating the MySQL database of the auditing application. The migration should work equivalent to the Opal services, because we migrate from a local MySQL to a RDS MySQL instance. Most of all we proved in the first iteration that the auditing application works well with an RDS MySQL instance.

5.1 Migration Method Bachmann

In this section we apply the methodology of Bachmann and the tool to the MySQL database of the Opal services. In Section 2.5 we presented a rough overview of the different phases of Bachmann's methodology. Nevertheless, we follow every single step of the methodology. The more detailed methodology can be found in [15]. Without the knowledge of the detailed methodology, it can be hard to understand our actions and decisions that we present in the following.

5.1.1 Assessment

In the following the results of the intermediate steps are presented:

1. The migration is part of the student thesis. The focus was presented in Section 4.2.
2. Only the Opal manager and the resource manager access the MySQL database. We have to change the DAL.
3. Since we migrate a MySQL database, we also use a relational database service in the Cloud. The preferred database is a MySQL database in order to keep the adjustments small. We choose AWS RDS as database service in Section 2.7. The MySQL DB engine version on-premise is 5.1.67. Since the version 5.1.6.7 is not available in RDS, we choose the MySQL DB engine version 5.5.27.
4. We use the Cloud data migration tool of Bachmann. There is no need for additional hardware.
5. We use Cloud burst as migration scenario.
6. We do not use a project plan, because we can't estimate the workload at all. The SimTech prototype is dimensioned for a distribution along different components. We do not expect the appearance of many problems during the migration of the DBL.

Conclusion: We didn't have any problems in this phase.
Time spent: 0,2 days

5.1.2 Analysis and Design

In the following the results of the intermediate steps are presented:

1. We use a MySQL database from Amazon RDS. We choose the micro instance, since we do not expect a high system load.
2. There is no need for additional data cleansing. The data in the local MySQL database is expected to be correct. The data is migrated after the environment has been tested and it was ensured that the environment was running on-premise before the migration.
3. We migrate from a local MySQL database to a MySQL database in the Cloud. We do not carry out modifications on the data.
4. We have to change the database configuration in the configuration files of the Opal manager and the resource manager in order to enable access to the AWS RDS.
5. We do not expect to carry out further adaptations on the system architecture. Therefore, we do not need a plan for additional adaptations.

Instance Size	db.t1.micro
DB Storage	5GiB
DB Engine	mysql
DB Engine Version	5.5.27

Table 5.1: RDS Configuration

Conclusion: We didn't have any problems in this phase.
Time spent: 0,2 days

5.1.3 Migration

In the following the results of the intermediate steps are presented:

1. We only have tables with data and no stored procedures or triggers. There are no composite primary keys and therefore, we do not have to adapt them. We use the Cloud data migration tool of Bachmann in order to migrate the data. No problems occurred.
2. We adapt the configuration files of the Opal manager and the resource manager.
3. No further adaptations on adjacent systems needed.

In this phase the errors with ID B1 (see Table 5.2), B2 (see Table 5.3), B3 (see Table 5.4) and B4 (see Table 5.5) occurred.

ID	B1
Name	Cancel
Class	tool (operability)
Severity	high
Description	Pressing cancel during editing has not the expected effect. Instead of jumping back to the prior page the project page is shown.
Error Handling	Not clicking cancel anymore.
Solution	None
Adaptation	The bug needs to be fixed.

Table 5.2: Error B1

ID	B2
Name	Fill in the form problem
Class	tool (operability)
Severity	high
Description	In Step 2 (*Describe Desired Cloud Data Hosting Solution*) the user is asked about the product version that is used by the Cloud data store in the background. The cursor does not stay in the input field.
Error Handling	The product version can only be typed in, when the left mouse button is pressed down. Otherwise the cursor jumps out again.
Solution	None
Adaptation	This should be corrected so that the cursor stays in the input field.

Table 5.3: Error B2

ID	B3
Name	Status Bar Loading
Class	tool (understandability)
Severity	high
Description	There is a window in which the log for the export is shown and a window in which the log for the import is shown. There was a moment, where we did not know, if the import works automatically or if we had to press the start migration button again.
Error Handling	We were insecure about the status of the export/import and just waited for a few minutes.
Solution	None.
Adaptation	The user should be informed about the current status of the data migration. There should be a status bar that shows the current status e.g. in percentage terms or something like table 5 of 87 is exported. Apart from that it should be pointed out to wait for the completion of the second window.

Table 5.4: Error B3

ID	B4
Name	Logging
Class	tool (understandability)
Severity	high
Description	No timestamps for log messages are displayed.
Error Handling	None
Solution	None
Adaptation	Date and time should be printed for each log message.

Table 5.5: Error B4

Conclusion: In this phase several errors with severity *high* occurred. But we only had minor problems to fix or avoid them.
Time spent: 0,6 days

5.1.4 Test

In the following the results of the intermediate steps are presented:

1. Since the amount of data is small, we can compare all tables and all data sets one by one. We do not find any discrepancy in the migrated data. The migration in the domain of the case study works correctly.
2. Since we do not detect abnormalities in the migrated data, there is no need for adaptations on the system.
3. The simulation was run with the SimTech prototype and the MySQL database hosted on-premise. The same simulation was done with the migrated MySQL database. The SimTech prototype delivered the same results for both simulations. As a consequence, we do not perform further tests in the context of the case study.
4. No need for adaptations on adjacent systems

In this phase the errors with ID B5 (see Table 5.6), B6 (see Table 5.7) and B7 (see Table 5.8) occurred.

ID	B5
Name	Connection failed
Class	tool (operability)
Severity	high
Description	After the successful migration of the data the application was not able to connect to the MySQL database in the Cloud.
Error Handling	We tried to use the root connection. The application could then connect to the MySQL database in the Cloud.
Solution	With the user root and the root-password the application could connect to the MySQL database in the Cloud.
Adaptation	Since this is just a work-around we do not recommend to use the root and the root-password for this purpose.

Table 5.6: Error B5

ID	B6
Name	Connection failed
Class	tool (operability)
Severity	high
Description	After the application could successful connect to the MySQL database in the Cloud with root rights, we wanted the application to use the standard connection information. Therefore, we created the required users and granted the rights on the tables via a script. The application still could not connect to the database.
Error Handling	We were searching for information about settings on the MySQL Workbench for user administration.
Solution	We added a new server instance to the server administration in the MySQL Workbench for our AWS MySQL database. In the security options we assigned the administrative role *DBManager* to the new users.
Adaptation	The user should get a hint to create the needed users and to add the required administrative roles.

Table 5.7: Error B6

ID	B7
Name	Connection failed
Class	tool (operability)
Severity	high
Description	Although correct users with the required administrative roles existed in the MySQL database in the Cloud, the application could not connect to the database.
Error Handling	We were going through all the security (user and privilege) settings in the MySQL Workbench.
Solution	We set *max queries*, *max. updates*, *max connections* on a value greater than zero for each user.
Adaptation	The user should get information about the limitations for the different accounts (users).

Table 5.8: Error B7

Conclusion: In this phase several errors with severity *high* occurred. But we only had minor problems to fix them. We suggest to add information concerning the adjustment of the database security options. Furthermore, we would present information to the user about necessary adaptations in the user administration.
Time spent: 1,0 days

5.1.5 Data Processing

In Section 4.5 we presented a way to visualize iterations in a phase-driven migration project. However, no critical errors occurred during the migration of the MySQL database to the Cloud. This is why we leave out the visualization for this phase.
In Table 6.4 the amount of time we spent in the different phases of the methodology is shown. We did not manage to quantify the time we spent more accurate than in 0,2 day steps. Sometimes we forget to measure the exact time and sometimes we got distracted from our work. In the end we round to 0,2 day steps.

Assessment	0,2 days
Analysis and Design	0,2 days
Migration	0,6 days
Test	1,0 days

Table 5.9: Overview of the Time we Spent in the Phases of Bachmann's Methodology

5.2 Migration Method AWS and Bachmann

In this phase we apply the migration methodology of AWS in combination with the methodology and tool of Bachmann. We introduced the methodology to migrate an application to AWS in Section 2.7.1.

5.2.1 Cloud Assessment

The migration methodology of AWS addresses business users (see Section 2.7). They got additional needs for a migration. For our purpose *financial assessment*, *security assessment* and *compliance assessment* play a minor part. Therefore, we focus on the task *plan creation* in this phase. The tendency of the AWS plan is toward project management and involves the definition of success criteria for the migration project [52]. This includes *CAPEX*, *OPEX*, *time to market* and *availability* [52].
In the domain of the case study those things take a back seat to the technical issues. This is why our plan mostly addresses technical aspects and consists of the following steps:

1. Create an EC2 instance and choose an AMI (We chose the AWS Cloud services in Section 2.7).

2. Verify that the SimTech prototype runs locally on the EC2 instance (all components are compatible).
3. Define the target architecture (see Figure 5.1).
4. Identify software packages that need to be installed on the different EC2 instances.
5. Install the software packages on the EC2 instance and configure the databases.
6. Check if all components work on the different Cloud services.
7. Identify all components that need adjustments. The architecture of the SimTech prototype can be seen in Figure 5.1. All components that interact with the migrated component need to be adapted.
8. Test, if the connection to the different components work.
9. Validate that the SimTech prototype delivers correct results with the migrated components.

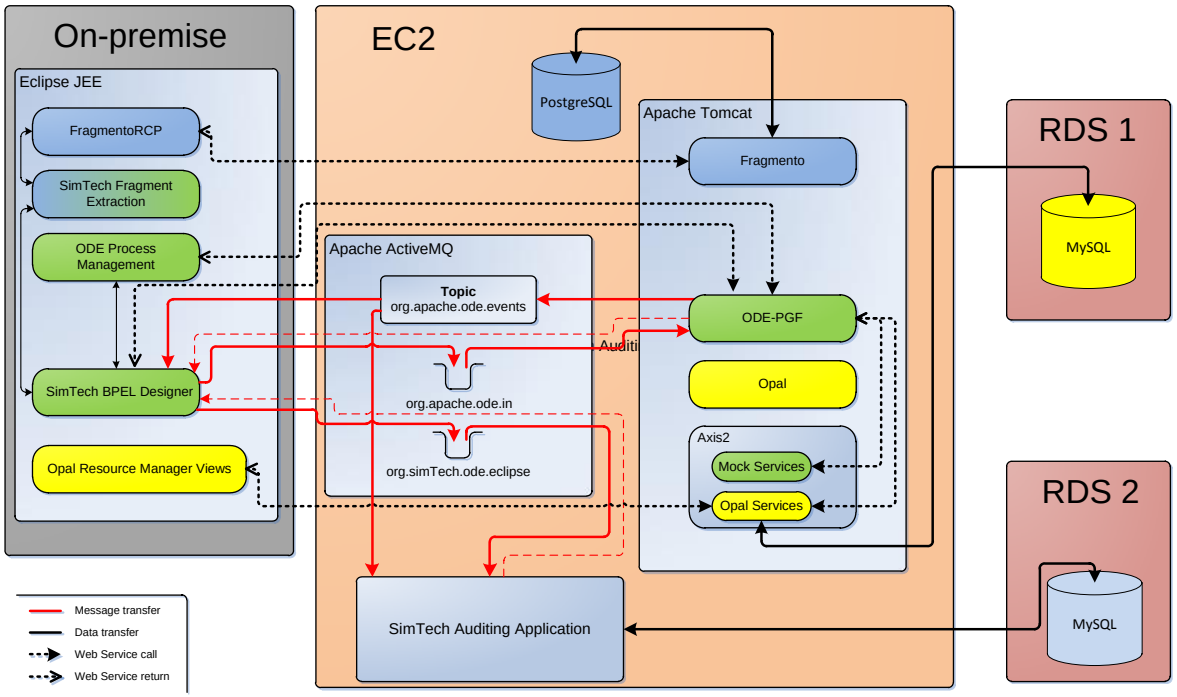

Figure 5.1: Distribution of the SimTech Prototype Components [25]

Conclusion: We didn't have any problems in this phase.
Time spent: 0,6 days

5.2.2 Proof of Concept

In the following the results of this phase are presented:

1. We create an EC2 instance and choose an AMI (see Table 5.10).
2. We install the whole prototype on the EC2 instance to make sure that the software components are compatible with the selected AMI.
3. We migrate the MySQL database of the Opal services from EC2 to AWS RDS. The MySQL DB engine version on-premise is 5.1.67. Since the version 5.1.6.7 is not available in RDS, we chose MySQL DB engine version 5.5.27. The interaction between EC2 and the migrated database works. The SimTech prototype produces the same results like the local installation. The specification of the RDS instance can be seen in Table 5.1.

Instance Size	m1.medium
Architecture	x86
Total Memory	3.75 GB
Processing Power	2 ECU
AMI-ID	ami-f1fa6ec1

Table 5.10: EC2 Configuration

In this phase the errors with ID A1 (see Table 5.11), A2 (see Table 5.12) and A3 (see Table 5.13) occurred.

ID	A1
Name	Connection to MySQL database failed
Class	tool (operability)
Severity	high
Description	After the successful migration of the data, the application was not able to connect to the MySQL database in the Cloud.
Error Handling	Configuring the AWS database security groups.
Solution	Granting the EC2 instance access to the RDS MySQL database. Each EC2 belongs to one EC2 security group. Each RDS instance belongs to one RDS security group. For each security group there can be applied several rules. With one rule we granted the EC2 security group access to the RDS security group (see Figure 5.2). As a consequence the EC2 instance was able to connect to the RDS instance.
Adaptation	Provide general information about security configuration of the Cloud services to the user.

Table 5.11: Error A1

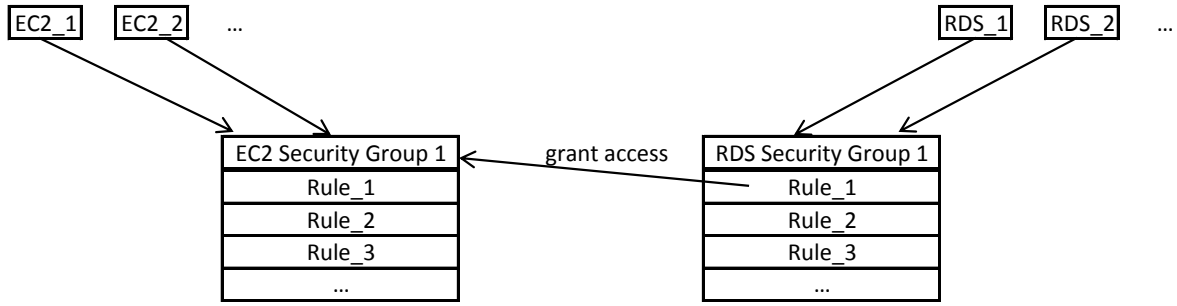

Figure 5.2: Amazon Web Services Security Groups

ID	A2
Name	Connection EC2 failed
Class	tool (Operability)
Severity	high
Description	After configuring the AWS security groups the EC2 instance could not be accessed from our local machine.
Error Handling	Configuring the windows firewall of the EC2 instance.
Solution	We disabled the windows firewall of the EC2 instance. Then we could access the EC2 instance.
Adaptation	No adaptations needed. This information was provided by the Cloud data migration tool, but we missed it.

Table 5.12: Error A2

ID	A3
Name	SimTech related Challenges
Class	others
Severity	high
Description	After installing the SimTech prototype on a EC2 instance with a 64-bit Windows Server 2008 R2 AMI, the Opal services did not run correctly. Tomcat always crashed with several Java and BPEL faults.
Error Handling	We tried to solve the problem by adding additional parameters and configuring parameters etc. We reinstalled all software components.
Solution	We installed the whole SimTech prototype on a EC2 instance with a 32-bit Windows Server 2008 R2 AMI. We are still not sure, whether we might got the installation wrong on the 64-bit instance or we configured something wrong. The SimTech prototype is running on the 32-bit version and therefore, we recommend to install it on this version.
Adaptation	We provide all lessons learned concerning the migration of the SimTech prototype to the Cloud in a step-by-step guide [34]. This guide can be found on the CD belonging to this student thesis.

Table 5.13: Error A3

Conclusion: There where several challenges in this phase. The major challenge of this phase was to get the Opal services to work on-premise. If we had more experience with the technologies related to the SimTech prototype and the SimTech prototype itself, we would not have spent so much time in this phase.
Time spent: 10,2 days

5.2.3 Data Migration Phase

We determined a MySQL database of AWS RDS as target database for the migration in Section 2.7. The concrete version of the MySQL DB engine was determined in Section 5.2.2. Therefore, we only apply the phases *migration* and *test* of Bachmann's methodology. We migrate the data of the MySQL database with the Cloud data migration tool of Bachmann. This time no errors occur, because we know how to set up the RDS instance correctly due to our experiences gained in Section 5.1 and Section 5.2.2.

Conclusion: We didn't have any problems in this phase. The Cloud data migration tool provides propositions for the adaptation of the DAL and the adaptation of the application logic layer. In combination with our experiences of the *proof of concept* phase this helps us to accomplish the *application migration* in the next phase.
Time spent: 0,6 days

5.2.4 Application Migration Phase

In the following the results of this phase are presented:

1. We identify all components that need to be changed. The target architecture of the distributed SimTech prototype can be seen in Figure 5.1. All components that interact with the migrated components need to be adapted. Amazon distinguishes two migration strategies in this phase: The *Forklift Migration Strategy* picks a whole application and moves it to the Cloud, whereas the *Hybrid Migration Strategy* uses an iterative approach, where parts of the system are moved to the Cloud [52]. We use the *Hybrid Migration Strategy*, since only parts of the SimTech prototype are moved to the Cloud (see Figure 5.1).

2. The connection to the different components works.

3. The prototype delivered correct results.

Conclusion: We didn't have any problems in this phase. Most of the propositions of the Cloud data migration tool could not be applied to the SimTech prototype, because we also used a MySQL database in the Cloud and did not change the database schema. The tool supports the user with hints concerning changes of database type and hints concerning confidentiality and database product change.
Time spent: 1,0

5.2.5 Data Processing

In Section 4.5 we presented a way to visualize iterations in a phase-driven migration project. However, no critical errors occurred during the migration of the Opal services to the Cloud. This is why we leave out the visualization for this phase.
In Table 5.14 the time we spent in the different phases of the methodology is shown. We did not manage to quantify the time we spent more accurate than in 0,2 day steps. Sometimes we forget to measure the exact time and sometimes we got distracted from our work. In the end we round to 0,2 day steps.

Cloud Assessment	0,6 days
Proof of Concept	10,2 days
Data Migration Phase	0,6 days
Application Migration Phase	1,0 days

Table 5.14: Overview of the Time we Spent in the Different Phases of AWS Methodology in Combination With Bachmann's Methodology

6 Discussion and Lessons Learned

In this chapter we discuss aspects concerning the migration of the SimTech prototype. According to our evaluation design presented in Chapter 4, we analyse the evaluation data with the focus presented in Section 4.2. First of all we describe the challenges with the SimTech prototype and their effects on the evaluation (Section 6.1). In Section 6.2 we analyse all data that leads to improvement suggestions for the Cloud data migration tool. In addition different implementations for the improvement suggestions are discussed. Section 6.3 comprises all lessons learned regarding the migration methodology of Bachmann. Another important aspect is the overall quality of the migration. The quality of the migration and the final assessment of the migration is comprised by Section 6.4. In order to make the SimTech prototype run in the AWS Cloud several adaptations need to be performed. An overview of the most important adaptations is presented in Section 6.5. A step-by-step migration manual can be found in [34].

6.1 SimTech Prototype Challenges

We had to face several challenges with the SimTech prototype. We describe them in the following, because they influence the results of the evaluation.
The SimTech prototype consists of many different software packages. Although there was an installation guide provided, we were not able to get the SimTech prototype to work on our local machine at the beginning. Not all current versions of the software and the SimTech components are compatible. After we got new informations regarding the installation (see [53]), we installed all correct components and nearly got the prototype to work. The Opal services produced intermediate results until a certain workflow activity. As soon as the *createOpalMedia* sequence was entered, the simulation stopped with an fault.[1] Then we started to evaluate the methodology and the tool with the AWS migration methodology. Therefore, we had to migrate the SimTech prototype to the Cloud. In the AWS proof of concept phase we selected an AMI with an operating system. We used a 64-bit system, because the Opal services were running to a certain point on the 64-bit local machine. Indeed, several java and BPEL engine faults occurred, when we started the Opal services on the EC2 instance (see Table 5.13). Tomcat crashed before the *createOpalMedia* sequence was reached. Since we were unsure about the origin of the errors, we tried to solve them by searching on google for the error messages. Meanwhile we tried to get the Opal services to work fully on the local machine. Those two processes are illustrated in Figure 6.1.

[1] There is no section in this student thesis, that describes the Opal services in detail. However, the services can be found on the CD belonging to this student thesis.

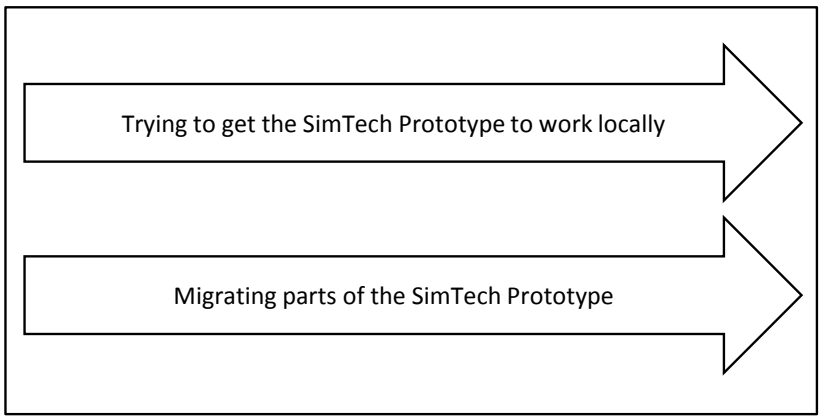

Figure 6.1: Two Parallel Processes in the AWS Proof of Concept Phase

Due to lack of time we had to work in parallel. This is why we spent additional time in the AWS proof of concept phase, although several testing and configuration actions were not migration related. We were not able to distinguish our actions and only assign the time of our migration related actions to the proof of concept phase. The ideal chronology of actions for our migration project is shown in Figure 6.2. We should have gotten the SimTech prototype to run on our local machine first. With the running SimTech prototype as pre-condition for the migration-project, we should have started with the migration in order to evaluate the methodology and the tool of Bachmann.

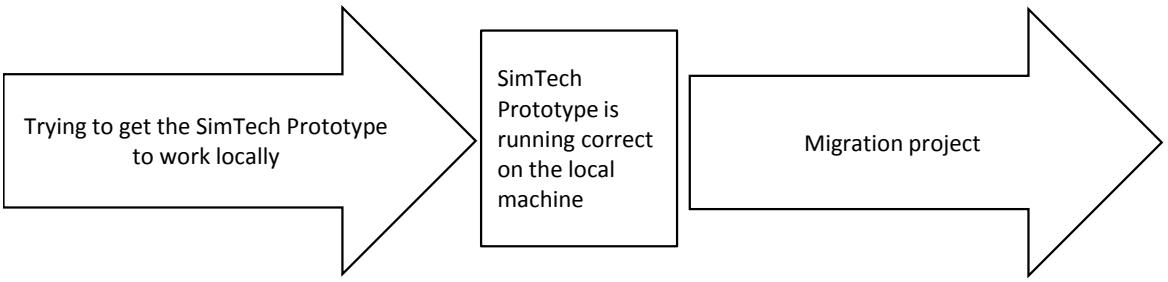

Figure 6.2: Ideal Sequence of Actions for the Case Study

Once we got the Opal services to work on the local machine, we created a new EC2 instance with a 32-bit AMI and installed all components. This time we chose a 32-bit version, because in the latest installation guide also a 32-bit machine was used [53]. After the testing in the weeks before we knew exactly how to install and set up everything correctly. Then the Opal services were also running in the Cloud without any problems. A detailed description of all required software packages can be found in [34].

6.2 Lessons Learned Tool

In this section we analyse the data that should lead to improvement suggestions on the Cloud data migration tool of Bachmann. All errors collected in Chapter 5 related to the Cloud data migration tool are analysed.
Firstly we present improvement suggestions with high priority. Secondly we focus on the improvement proposals with middle priority. There are no proposals with low priority.

6.2.1 High Priority

Related to the error with ID B1 (see Table 5.2):
This improvement suggestion is based on a bug. Pressing cancel during editing has not the expected effect. Instead of jumping back to the prior page the project page is shown. This needs to be fixed, because such a bug subtracts the acceptance of the end user.

Related to the error with ID B3 (see Table 5.4):
With focus on usability, a status bar that shows the current export or import process is really helpful to the user. He knows that the export or import has not finished yet and therefore, he should wait for the progress to finish and not execute further actions. During the evaluation there was a moment, where we did not know the current status of the application. There is a log window for the export and a log window for the import of data. The export seemed to be finished and we were not sure, whether we had to execute further actions (e.g. click the migrate button again) or the import works automatically. A status bar can be implemented in different ways. A neatly solution would base the progress on the amount of data that has already been exported or imported. An easier solution would base the progress on the number of tables that has already been exported or imported. If there is no time for implementing a status bar, a simple hint to wait for the completion of export and import would be helpful, too.

The errors with the ID B5 (see Table 5.6), B6 (see Table 5.7) and B7 (see Table 5.8) are strongly related to database administration and database security. Therefore, we propose to add the information presented in Table 6.1 to step 5 of the Cloud data migration tool. To be more exact, we would add this information to the intermediate step *Data Access Layer*.

USERS	All necessary users need to be added to the new database in the Cloud. All those users need to be dedicated all the necessary administrative roles. In addition their access rights (e.g. max queries, max connections) need to be configured properly.

Table 6.1: Improvement Suggestion for the Cloud Data Migration Tool

The errors with the ID A1 (see Table 5.11) and A2 (see Table 5.12) are strongly related to Cloud provider specific security. We propose to add the information presented in Table 6.2 to step 5 of the Cloud data migration tool. To be more exact, we would add this information to the intermediate step *Network Layer*.

CLOUD PROVIDER SECURITY	It needs to be checked if the Cloud services require additional security configuration. (E.g. the Cloud services RDS of AWS require additional configuration in security group settings. Otherwise the RDS instances can't be accessed.)

Table 6.2: Improvement Suggestion for the Cloud Data Migration Tool

6.2.2 Middle Priority

Related to the error with ID B2 (see Table 5.3):
In Step 2 (*Describe Desired Cloud Data Hosting Solution*) of the Cloud data migration tool, the user is asked for the product version that is used by the Cloud data store in the background. The product version can only be entered, when the left mouse button is pressed down. Otherwise the cursor jumps out again. This should be corrected so that the cursor stays in the input field.

Related to the error with ID B4 (see Table 5.5):
Right now the logging information is presented without a timestamp. We consider timestamps to be important for debugging. The sequence of actions of the application can be tracked better. While new timestamps are being produced, the application is still running and has not crashed. So timestamps give also feedback about the current status of the application.

6.2.3 Support Adaptation of the Architecture

Information about the adaptation of the architecture was provided by the tool. However, we missed useful information like the need for configuring the firewall (see Table 5.12). The tool delivered suggestions, that were in most cases not relevant due to the focus of the evaluation and our migration scenario. E.g. the tool suggested to take care of different semantics of NULL. Since we migrated from MySQL to MySQL, we did not have to consider this aspect. In a more complex scenario, where different kinds of databases are involved, the provided hints would be useful.

6.3 Lessons Learned Methodology

This section focuses on the important aspects of the evaluation for the methodology (see Section 4.2).

6.3.1 Methodology fit in

One important question is how well the migration methodology of Bachmann fits into a holistic migration scenario. Therefore, we have to analyse which phases of Bachmann's methodology and the AWS methodology are overlapping. Figure 6.3 shows all phases pertinent to the case study. During our evaluation with the SimTech prototype in Chapter 5,

we realized that the phases *assessment* and *analysis and design* by Bachmann were covered by the *Cloud assessment phase* and *proof of concept phase* of the AWS approach. The cooperation of the migration methodology and the Cloud data migration tool of Bachmann and the AWS migration approach is shown in Figure 6.3. The tool also provided us many proposals concerning the adaptation of the DAL. However, due to the migration of a MySQL database to a RDS MySQL instance, we did not have compatibility problems. Therefore, those suggestions were not helpful in our case. We propose to hide all unnecessary hints.

In general Bachmann's methodology and tool fit in well into the AWS approach in the case study .

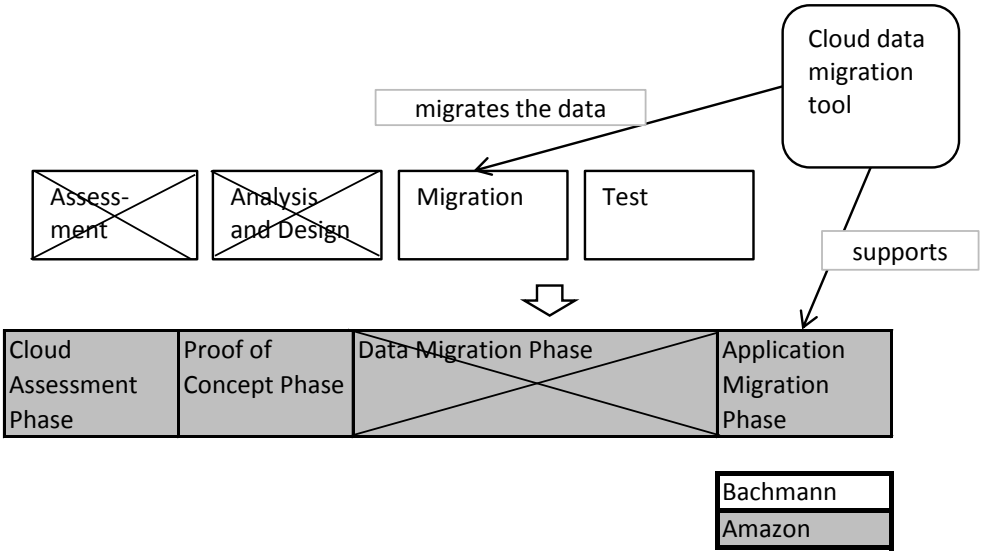

Figure 6.3: Result of Fitting in Bachmann's Methodology in the Methodology of AWS

6.3.2 Possible Iterations

The evaluation should also focus on the detection of iterations between the different phases of Bachmann's methodology (see Chapter 4). Since no *critical errors* (see Section 4.5) occurred, there was no need for a step back in a prior phase. In conclusion there were no iterations between the phases in the case study.

6.3.3 Analysis of the Time Spent

Another important aspect was the discovery of the most time consuming phases (see Chapter 4). We assume that the most time consuming phases are more difficult and therefore, we might provide more helpful information to the user in those phases (see Section 4.5). We split the analysis of the time spent in two parts. The first part focuses on the time spent in Bachmann's methodology, whereas the second part analyses the time spent in the holistic migration scenario with AWS.

Bachmann

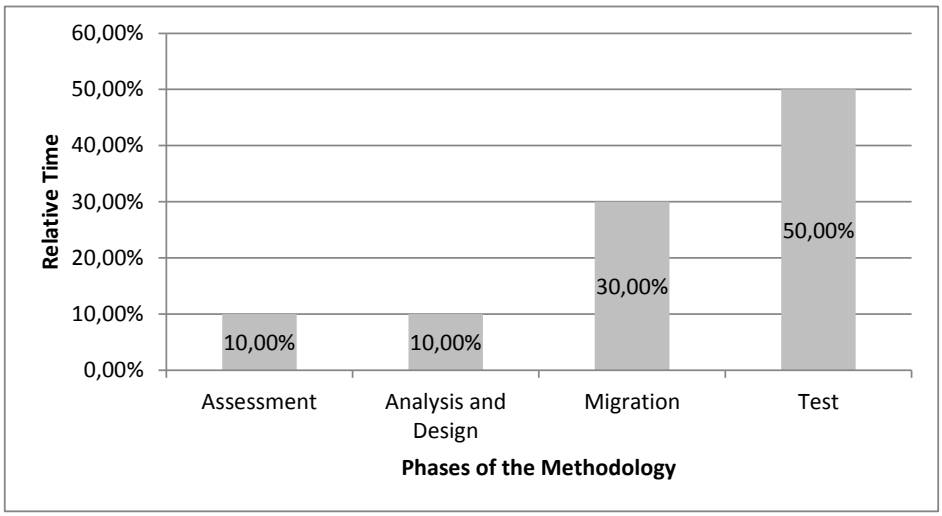

Figure 6.4: Relative Time Spent in the Different Phases of Bachmann's Methodology

In Figure 6.4 the relative time we spent in the first step of the iteration is shown (see Section 4.1).

The phases *assessment* and *analysis and design* were applied straightforward. No errors occurred and this is why we propose to not change those phases.

During the *migration* phase we used the Cloud data migration tool. In this phase we detected some shortcomings concerning the tool. Those improvement suggestions were discussed in Section 6.2.

The most time consuming phase was the *testing* phase. Three errors occurred. All of them are related to the Cloud data migration tool. They have been discussed in Section 6.2.

AWS

In Figure 6.5 the relative time we spent in the second step of the iteration is shown. The phase *Cloud assessment* was applied straightforward. No errors occurred and we do not recommend to change anything in this phase.

The phase *proof of concept* has a considerable proportion of the time spent in the second step of the iteration. It is also noticeable, that only errors in this phase of the methodology occurred. On the one hand this can be explained due to the challenges with the SimTech prototype (see Section 6.1). On the other hand this phase is predetermined for first tests and technology validating [52]. This phase helped us to get to know the AWS EC2, because we had not worked with AWS before. In our case, we spent a long time there, because we could not get the Opal services to work *locally*. *Locally* in this case means locally on the EC2 instance without distributing components of the SimTech prototype using further services from AWS.

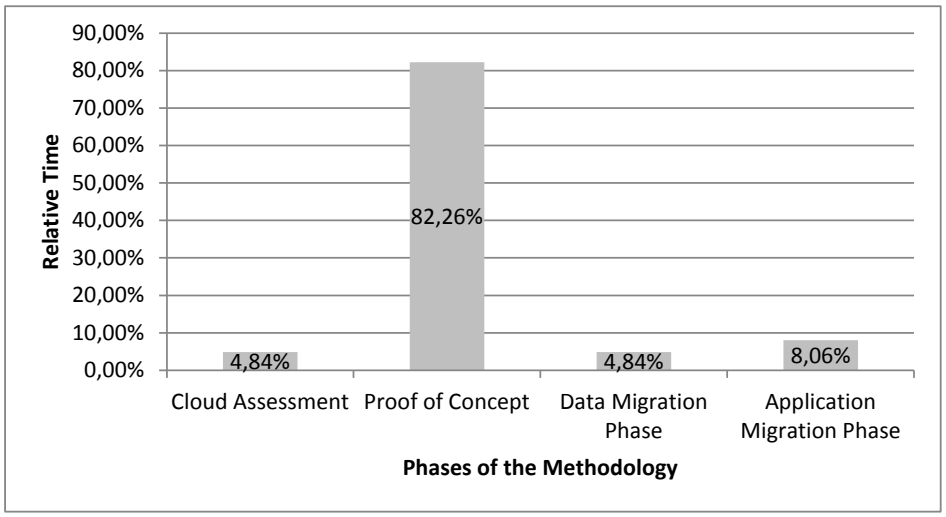

Figure 6.5: Relative Time Spent in the Different Phases of the AWS Methodology in Combination with Bachmann's Methodology

The important question is now, whether to add such a *proof of concept phase* to the methodology of Bachmann. This question is not easy to answer, because from our point of view there is no universal answer to this.
In the first iteration of our migration project (see Section 4.1) we would not extract advantages from a *proof of concept phase*. To be more exact, the errors would occur in the *proof of concept* phase and the succeeding *migration phase* would be faultless. So just the name of the phase, in which the errors occur, would change. Therefore, the adding of a *proof of concept phase* to Bachmann's methodology would not yield an advantage.
Considering the background of a very complex system, a benefit of from a *proof of concept phase* can be derived. A pilot that encloses the most important and critical parts of the system can be tested. Furthermore, a smaller system has less potential sources of trouble due to the smaller complexity. Especially, since people have not many years experience with the different Cloud solutions, a *proof of concept phase* helps to validate the technology. This leads to a mitigation of risk. In such a case we propose to add a *proof of concept phase* to Bachmann's methodology. However, the *proof of concept phase* needs to be clearly separated from the *testing phase*, where e.g. the different modules are integrated.
A point of criticism is the missing iteration of the phases in practical use [39]. In a project, for which many iterations between the phases of the methodology are presumed, a *proof of concept phase* can minimize those iterations by detecting errors in an early stage of the migration project. In this case a *proof of concept phase* is helpful, too.
Since there exist convincing arguments in favour of both sides, we propose to add a optional *proof of concept phase* to the migration methodology of Bachmann. In complex situations the *proof of concept phase* can be applied.

In Figure 6.6 the absolute time we spent in all phases of both approaches is shown. It took us 0,6 days to finish the *data migration phase*. The *data migration phase* in our evaluation consists of the phases *migration* and *test* of Bachmann (see Figure 6.3). In the first step of the iteration it took us 1,6 days to finish both phases. This means, that we spent 1 day less for executing

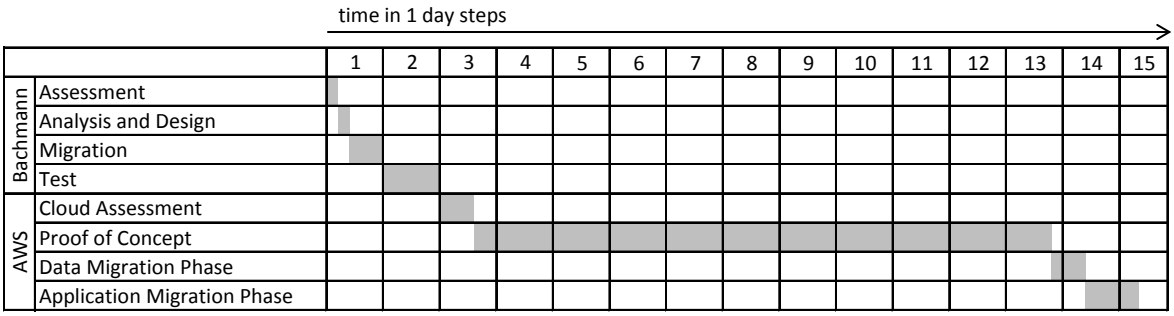

Figure 6.6: Migration Projects Time Line

the same phases during the second step of the iteration. In addition, no errors occurred in the second step compared to four errors in the first step. We trace this back to the fact, that we profited from *learning effects* of the first iteration.

The *application migration phase* was applied straightforward. No errors occurred. We had spent a lot of time in the *proof of concept phase* and therefore, we gained enough experience to migrate the application without a challenge.

6.4 Migration Assessment

With respect to the focus presented in Section 4.2 one key aspect for the evaluation is the quality of the technical migration of the DBL. In the domain of the case study the migration quality is perfect. All data was successfully migrated and we did not recognize abnormalities in the migrated data. After all the SimTech prototype is running with services from AWS as shown in Figure 5.1.

Another important lesson learned is, that insides and previous knowledge of an application and components are important factors for a migration project. In our case the application was coupled loosely and therefore, we had good preconditions for a distribution along services from AWS. In addition the architecture of the SimTech prototype was relative easy. Nevertheless, we had to face challenges while migrating the SimTech prototype. The pure data migration in the case study was the easier task. The adaptation of the application to make it run in the Cloud in a provider specific environment was more tricky. Against the backdrop of a more complex system, there might be different specialists for different layers or components of the application. In such case the migration can become a much more complex task. Depending on the changes required and the challenges coming up, the different specialists have to work in a team to accomplish the migration project. Furthermore, they also have to take account of financial, security and compliance aspects. In our case those additional requirements played a minor part. Depending on the migration scenario, the Cloud data migration tool also proposes patterns concerning those aspects. E.g. Cloud data patterns for ensuring security and confidentiality.

6.5 Adaptations SimTech Prototype

In order to migrate the SimTech prototype to the Cloud several adaptations have to be realized. In the following we give an overview of the most important adjustments. Another result of this student thesis is a step-by-step guide, that explains all adaptations on the SimTech prototype in detail [34]. This step-by-step guide can be found on the CD belonging to this student thesis.

In order to realize the architecture presented in Figure 5.1 those modifications have to be carried out:

1. The configuration files of the resource manager and the opal manager have to be changed:
   - database URL
   - database name
   - user and password
2. All service endpoints of the Opal services have to be changed to the new endpoint of the ODE-PGF on the EC2 instance
3. The SimTech properties in SimTech BPEL designer have to be changed:
   - endpoint of the ODE-PGF
   - endpoint of the ActiveMQ
   - Fragmento service Uniform Resource Identifier (URI)
4. The preferences in the SimTech auditing application need to be changed:
   - database URL
   - database name
   - user and password

7 Outcome and Future Work

This student thesis has originated the evaluation of Bachmann's methodology for migration of the DBL to the Cloud. In addition, the Cloud data migration tool of Bachmann was evaluated. Improvement suggestions for both the tool and the methodology resting on the evaluation results were presented.

In Chapter 2 we gave an overview of Cloud computing fundamentals and described the SimTech prototype. Furthermore, Bachmann's methodology, the Cloud data migration tool and AWS were introduced. The main topic of this student thesis is the evaluation of the methodology and the tool. The Chapter 3 provided an overview of existing approaches and state of the art in the domain of the thesis. Our research was positioned towards the work of other authors in several fields of application. It laid the groundwork for the evaluation design presented in Chapter 4. One core contribution of Chapter 3 was the *ITIL CSI seven-step improvement process*, that we used as framework for our evaluation. Additionally, aspects concerning software quality were discussed. On the basis of this understandings, we could focus on the important software quality aspects for the Cloud data migration tool. Furthermore, different kinds of data in evaluation were presented. Based on this results, we could select the convenient data collecting approach in Chapter 4. In addition, Chapter 4 consolidated the findings of the previous Chapter 3. As a result, a structured overview of our research design was given. Most important, the focus of our evaluation was marked out. Finally, in Chapter 5 we migrated the SimTech prototype with the help of Bachmann's methodology, the Cloud data migration tool of Bachmann and the AWS approach to the Cloud. We collected the data for our evaluation based on the evaluation design presented in Chapter 4. All shortcomings concerning the methodology and the tool of Bachmann were documented. In the attaching Chapter 6 we illustrated challenges, that occurred during the evaluation and that influenced the results. We also analysed the evaluation data. Based on the results of the analysis, the lessons learned for the methodology and the Cloud data migration tool were worked out. These lead to precise improvement suggestions. Our results and the special circumstances of our case study were compared to a more complex migration project. At the end, an overview of the required adaptations for migrating the SimTech prototype to AWS was presented. This student thesis contributes two more artefacts regarding the SimTech prototype. Firstly, a step-by-step guide, that explains all necessary details for migrating the SimTech prototype as shown in Figure 5.1. Secondly, an AMI that contains the complete installation of all SimTech components. With those two artefacts the migration of the SimTech prototype to AWS is accelerated.

Currently, the improvement suggestions evolving from this student thesis have not been implemented. In addition, Bachmann also described improvement suggestions, that have not

been realized yet [15]. Future work should at least focus on those suggestions, that are classified with a high priority. After the implementation of the improvements, the methodology and the tool should be further evaluated. The results evolving should be compared to our results in order to measure the improvement success.
The methodology and the Cloud data migration tool were used in a relatively simple migration scenario. It has been shown, that they support the user successfully in a migration project in the context of a student thesis. Nevertheless, only further evaluation in a more complex environment can ensure, that the methodology and tool support effectively in bigger migration projects.
With respect to the SimTech prototype, the migration to AWS as shown in Figure 5.1 is a first achievement. We assumed, that the migration of the MySQL database of the auditing application is equivalent to the migration of the MySQL database of the Opal services. This has to be proofed by migration.
We migrated a prototype to the Cloud. The migration of a productive system differs. E.g. the performance needs to be evaluated. Massive parallel hits of users and the influence of network errors require further examination. It has also to be evaluated, whether the presented solution is also applicable to the productive system. For the backdrop of many parallel simulations,

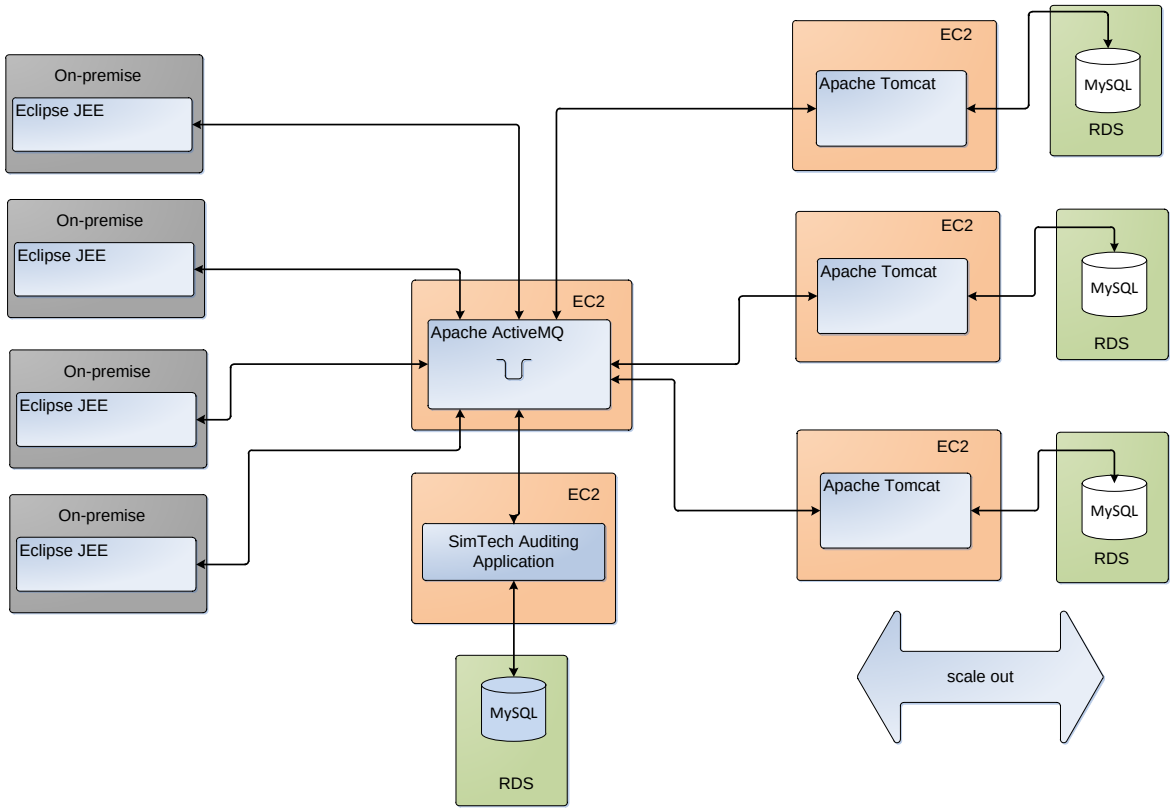

Figure 7.1: Simplified Version of the SimTech Prototype With Clustered Tomcat Servers

a clustering of components of the SimTech prototype as shown in Figure 7.1 is conceivable. Therefore, we propose to let the ActiveMQ run on a separate EC2 instance. All simulations should use the same auditing application, which also runs on a own EC2 instance. If the

load increases, automatic *scaling out* of EC2 instances containing Tomcat can be automated. If the load decreases, different simulations can share an EC2 instance. In general the SimTech prototype is well prepared for a distribution along different services. However, the clustering will bring new challenges. Especially with respect to the migration of running simulations from one EC2 to another EC2. Further research in this domain has to be done.

Bibliography

[1] The ISO 9126 Quality Model Attribute Tree. http://www.springerimages.com/Images/RSS/1-10.1007_978-94-007-2792-2_54-0 (2011)

[2] Al-Qutaish, R.E.: Quality Models in Software Engineering Literature: An Analytical and Comparative Study (2010)

[3] Amazon Web Services, Inc.: Amazon DynamoDB (Beta). https://aws.amazon.com/dynamodb/

[4] Amazon Web Services, Inc.: Amazon Elastic Compute Cloud (Amazon EC2). https://aws.amazon.com/ec2/

[5] Amazon Web Services, Inc.: Amazon Elastic MapReduce (Amazon EMR). https://aws.amazon.com/elasticmapreduce/

[6] Amazon Web Services, Inc.: Amazon ElastiCache (Beta). https://aws.amazon.com/elasticache/

[7] Amazon Web Services, Inc.: Amazon Machine Images (AMIs). https://aws.amazon.com/amis

[8] Amazon Web Services, Inc.: Amazon Redshift (Beta). https://aws.amazon.com/redshift/

[9] Amazon Web Services, Inc.: Amazon Relational Database Service (Amazon RDS). https://aws.amazon.com/rds/

[10] Amazon Web Services, Inc.: Amazon Web Services in Education. http://aws.amazon.com/de/grants/

[11] Amazon Web Services, Inc.: Auto Scaling). https://aws.amazon.com/de/autoscaling/

[12] Amazon Web Services, Inc.: Elastic Load Balancing. https://aws.amazon.com/elasticloadbalancing/

[13] APM Group: Welcome to the Official ITIL® Website. http://www.itil-officialsite.com/

[14] Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (2009)

[15] Bachmann, T.: Entwicklung einer Methodik für die Migration der Datenbankschicht in die Cloud. Diploma Thesis No. 3360, University of Stuttgart, Germany (2012)

[16] Berander, P., Damm, L.O., Eriksson, J., Gorschek, T., Henningsson, K., Jönsson, P., Kaagström, S., Milicic, D., Maartensson, F., Rönkkö, K., Tomaszewski, P.: Software Quality Attributes and Trade-Offs. Blekinge Institute of Technology (2006)

[17] Beywl, W.: Standards für Evaluation / Deutsche Gesellschaft für Evaluation. Geschäftsstelle DeGEval, Köln, 4th edn. (2008)

[18] Botella, P., Burgués, X., Carvallo, J.P., Franch, X., Grau, G., Marco, J., Quer, C.: ISO/IEC 9126 in Practice: What do we need to know? In: Software Measurement European Forum (SMEF 2004)

[19] Carnegie Mellon University. Software Engineering Institute (SEI): `http://www.sei.cmu.edu/`

[20] Daniel, G.: Software Quality Assurance: From Theory To Implementation. Pearson Education (2004)

[21] E. Taylor-Powell, S. Steele: Collecting Evaluation Data: An Overview of Sources and Methods (1996)

[22] Geck, B.: Übersicht der Prozessmodelle CMMI, SPICE, ITIL. `http://www.gebecom.de/Prozess.pdf` (2006)

[23] Gesellschaft für Evaluation e.V.: `http://www.degeval.de/`

[24] Google, Inc.: Uploading and Downloading Data - Google App Engine - Google Code. `https://developers.google.com/appengine/docs/python/tools/uploadingdata` (2012)

[25] Hahn, M.: Entwicklerhandbuch SimTech-Protoyp. `http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/Entwicklerhandbuch.pdf` (2011)

[26] Hahn, M., Vukojevic, K.: Installationsanleitung SimTech-Protoyp. `http://kaaroda.informatik.uni-stuttgart.de:8080/userContent/InstallationsanleitungSimTechPrototyp.docx` (2012)

[27] Hotta, S.: Ausführung von Festkörpersimulationen auf Basis der Workflow Technologie. Diploma Thesis No. 3029, University of Stuttgart, Germany (2010)

[28] ISO/IEC: ISO Standard 9126: Software Engineering – Product Quality. ISO/IEC (2001)

[29] ISO/IEC: ISO/IEC 25010 - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models. Tech. rep. (2010)

[30] Kitchenham, B., Pickard, L., Pfleeger, S.L.: Case Studies for Method and Tool Evaluation. IEEE Softw. 12(4), 52–62 (1995)

[31] Laszewski, T., Nauduri, P.: Migrating to the Cloud: Oracle Client/Server Modernization. Syngress Publishing, 1st edn. (2011)

[32] Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Tech. rep. (2009)

[33] Myers, M.D.: Qualitative Research in Information Systems. MIS Quarterly 21(2) (1997)

[34] Passow, S.: Migrating the SimTech Prototype to Amazon Web Services (2013)

[35] Redkyn, O.: Analyis of the Waterfall Model for the Software Development Process and Possibilities for its Improvement

[36] Schumm, D., Karastoyanova, D., Leymann, F., Strauch, S.: Fragmento: Advanced Process Fragment Library. In: Proceedings of the 19th International Conference on Information Systems Development (ISD'10), Prague, Czech Republic, August 25 - 27, 2010. pp. 659–670. Springer (2010)

[37] Shull, F., Carver, J., Travassos, G.H.: An Empirical Methodology for Introducing Software Processes. In: Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering. pp. 288–296. ESEC/FSE-9, ACM, New York, NY, USA (2001)

[38] Silverman, D.: Interpreting Qualitative Data. SAGE Publications Ltd, 4th edn. (2011)

[39] Sommerville, I.: Software Process Models. ACM Comput. Surv. 28(1), 269–271 (1996)

[40] Sonntag, M., Görlach, K., Karastoyanova, D., Leymann, F., Malets, P., Schumm, D.: Views on Scientific Workflows. In: Proceedings of the 10th International Conference on Perspectives in Business Informatics Research (BIR 2011), 2011. Lecture Notes in Business Information Processing, vol. 90, pp. 321–335. Springer-Verlag (2011)

[41] Sonntag, M., Görlach, K., Karastoyanova, D., Leymann, F., Reiter, M.: Process Space-based Scientific Workflow Enactment. International Journal of Business Process Integration and Management (IJBPIM) Special Issue on Scientific Workflows, Vol 5, No. 1, pp. 32-44 pp. 32–44 (2010)

[42] Sonntag, M., Hahn, M., Karastoyanova, D.: Mayflower - Explorative Modeling of Scientific Workflows with BPEL. In: Proceedings of the Demo Track of the 10th International Conference on Business Process Management (BPM 2012), CEUR Workshop Proceedings, 2012. pp. 1–5. CEUR Workshop Proceedings (2012)

[43] Sonntag, M., Karastoyanova, D.: Next Generation Interactive Scientific Experimenting Based On The Workflow Technology. In: Alhajj, R., Leung, V., Saif, M., Thring, R. (eds.) Proceedings of the 21st IASTED International Conference on Modelling and Simulation (MS 2010), 2010. ACTA Press (2010)

[44] Stationery Office: ITIL Continual Service Improvement 2011. The Stationery Offic Series, Stationery Office (2011)

[45] Strauch, S., Andrikopoulos, V., Bachmann, T., Leymann, F.: Migrating Application Data to the Cloud Using Cloud Data Patterns. In: Proceedings of the 3rd International Conference on Cloud Computing and Service Science, CLOSER 2013, 8-10 May 2013, Aachen, Germany. pp. 0–11. SciTePress (2013)

[46] Strauch, S., Kopp, O., Leymann, F., Unger, T.: A Taxonomy for Cloud Data Hosting Solutions. In: Proceedings of the International Conference on Cloud and Green Computing (CGC '11). pp. 577–584. IEEE Computer Society (2011)

[47] Team, C.P.: CMMI® for Development, Version 1.3. Tech. rep., Carnegie Mellon University (2010)

[48] Tergan, S.O.: Grundlagen der Evaluation: Ein Überblick. Qualitätsbeurteilung multimedialer Lern- und Informationssysteme. Evaluationsmethoden auf dem Prüfstand, BW Bildung und Wissen., Nürnberg (2000)

[49] The Apache Software Foundation: Apache ODE (Orchestration Director Engine). `http://ode.apache.org`

[50] The Apache Software Foundation: The Apache Software Foundation. Apache Axis2/Java. `http://axis.apache.org/axis2/java/core/`

[51] The Eclipse Foundation: BPEL Designer Project. `http://eclipse.org/bpel`

[52] Varia, J.: Migrating Your Existing Applications to the AWS Cloud. A Phase-Driven Approach to Cloud-Migration (2010)

[53] Vukojevic, K.: Installation der SimTech Umgebung für Entwickler - Hinweise, Ergänzungen, Fehlerbehandlung (2012)

[54] Wottawa, H.: Evaluation. Pädagogische Psychologie. Ein Lehrbuch., Beltz PVU., Weinheim (2001)

All links were last followed on April 22, 2013.

Declaration

All the work contained within this thesis, except where otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

Stuttgart, April 24, 2013 _____

(Stephan Passow)