Institut für Architektur von Anwendungssystemen (IAAS)
Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3469

Evaluierung einer Methodik für die Migration der Datenbankschicht in die Cloud basierend auf einer Fallstudie aus der Industrie

Nikolay Nachev

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer aus Industrie:	Dipl. -Ing. Albrecht Stäbler
Betreuer aus Wissenschaft:	Steve Strauch
begonnen am:	Februar 28, 2013
beendet am:	August 30, 2013
CR-Klassifikation:	C.2.4, C.4, D.2.8, D.2.11, H.2.1, H.2.4, H.3.4, H.4.2

Abstract

Im Laufe der letzten Jahre ist die Popularität und die Akzeptanz des Cloud Computing stetig gewachsen. Nicht nur durch die höhere Skalierbarkeit und Verfügbarkeit, sondern auch durch sein erhebliches Potenzial Kosten zu minimieren, ist Cloud Computing bei Unternehmen von großem Interesse.

Diese Diplomarbeit widmet sich der Evaluation einer Methodik und eines Tools für die Migration der Datenbankschicht einer Anwendung in die Cloud. Die Evaluation basiert auf einer Fallstudie aus der Industrie, die von der Firma NovaTec Holding GmbH zur Verfügung gestellt wurde. Um ein besseres Verständnis bezüglich des Evaluationsprozesses zu bekommen, wird ein Evaluationsentwurf erstellt, mit dessen Hilfe Daten gesammelt werden. Alle auftretenden Fehler werden dabei dokumentiert und analysiert. Die gewonnenen Erkenntnisse streben die kontinuierliche Verbesserung der Methodik und des Tools an.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

1. Einführung

Im Laufe der letzten Jahre ist die Popularität und die Akzeptanz des Cloud Computing stetig gewachsen. Nicht nur durch die höhere Skalierbarkeit und Verfügbarkeit aber auch vor allem durch sein erhebliches Kostenersparnis-Potenzial, ist Cloud Computing bei Unternehmen von großem Interesse. Dieses Potenzial ermöglicht die höheren Kapitalkosten (Capex) in regelmäßigen kleineren Betriebskosten (Opex) umzuwandeln und infolgedessen die Investitionsausgaben für Software, Hardware und Netzwerk Equipment zu reduzieren.

Wie alle anderen Migrationsprojekte erfordert die Migration in die Cloud eine sorgfältige Planung und eine ausführliche Methodik, um die erfolgreiche Ausführung zu gewährleisten [40]. Grundsätzlich existieren zwei Wege eine bereits entwickelte Anwendung in die Cloud zu migrieren. Der erste Weg beschreibt die Migration der einzelnen Schichten (siehe Abschnitt 2.1) einer Applikation in die Cloud, wobei die Migration der Datenbankschicht die am häufigsten vorkommende Migrationsart ist [40]. Diese ermöglicht dem Benutzer, falls keinen Inkompatiblitäten auftreten, die restlichen Schichten einer Anwendung, bis auf wenige Anpassungen, unangefasst zu lassen. Im Fall einer veralteten Anwendung, die schwer zu warten oder zu aktualisieren ist, wird diese mit Hilfe aktueller Technologien und Programmiersprachen in der Cloud neu entwickelt, um den aktuellen Business Anforderungen zu genügen. Eine solche Vorgehensweise wird zu einem neuen Entwicklungsprojekt anstatt einer einfacheren Plattformmigration [40].

Im Rahmen seiner Diplomarbeit hat Thomas Bachmann eine Methodik und ein Tool für die Migration der Datenbankschicht entwickelt [39]. Die Methodik deckt ein breites Spektrum von Migrationsszenarien ab und das Tool ermöglicht den richtigen Cloud Anbieter auszuwählen. Deren Evaluation, basierend auf einer Industrie Fallstudie, ist das Hauptziel dieser Arbeit.

1.1. Motivation

In vielen Unternehmen werden die Geschäftsprozesse immer noch in manuellen Schritten ausgeführt und die Datenhaltung und -erfassung erfolgt in Excel Sheets. Dieser Ansatz zur Datenhaltung zeichnet sich durch eine höhere Fehleranfälligkeit aus und ist mit großem Aufwand zu warten. Um diese Hindernisse zu überwinden, wurde in der Firma NovaTec [3] ein Entwicklungsprojekt namens NovaERM [29] gestartet. Das Ziel dieses Projektes ist die Entwicklung einer Anwendung, die eine adäquate IT-Unterstützung anbietet.

NovaERM soll die Geschäftsprozesse automatisieren und mit Hilfe eines Business Process Management Systems (BPMS) ausführen. Da das Tool ständig weiterentwickelt wird, liegt

der Fokus der Funktionalität aktuell auf der Automatisierung des Einstellungsprozesses eines Mitarbeiters und der Erstellung eines Kompensationsplans [29].

Der modulare Aufbau von NovaERM (siehe Abschnitt 2.3) ermöglicht die zentralisierte Auslagerung den NovaERM Datenbanken und den gleichzeitigen Zugriff auf diese durch unterschiedliche Instanzen der Webapplikation.

Im Rahmen dieser Arbeit werden die zwei Datenbanken von NovaERM in die Cloud migriert, indem die von Bachmann beschriebenen Methodik und Tool sowie das Cloud Datenmigrationsszenario *Reines Outsourcing des Datenbank Schicht (DBL)* verwendet werden [39].

1.2. Problemstellung und Zielsetzung

Das Ziel dieser Arbeit ist die Evaluation der Methodik zur Migration der Datenbankschicht in die Cloud, die im Rahmen der Diplomarbeit von Bachmann [39] entwickelt wurde. Die Evaluation basiert auf einer Industrie Fallstudie, die von der Firma NovaTec Holding GmbH[1] zu Verfügung gestellt wird. Der Fokus liegt dabei auf der Migration der Datenbankschicht in die Cloud. In einem zweiten Schritt wird noch die dazugehörige Webapplikation migriert, um die Latenz in der Kommunikation zwischen den einzelnen Komponenten zu reduzieren und die möglichen Vorteile des Cloud Computing auszunutzen. Ein weiterer Aspekt in diesem Schritt ist die Überprüfung, ob die Methodik von Bachmann mit anderen kombinierbar ist, um eine ganzheitliche Methodik für die Migration der gesamten Anwendung zu ermöglichen.

Die Migration Datenbankschicht erfolgt mit Hilfe des Tools von Bachmann. Die Eignung dieses Tools zur Durchführung und Unterstützung der Migration ist Teil unserer Evaluation.

Als Cloud Datenmigrationsszenario wird das *Reines Outsourcing des Datenbank Schicht* eingesetzt. Bei diesem Szenario bleibt den Typ des Datenspeichers (RDBMS, NoSQL oder BLOB Datenspeicher) unverändert. Die Verwaltung des DBL erfolgt dabei entweder in der Cloud selbst oder durch den Cloud Anbieter. Eine detaillierte Beschreibung dieses Cloud Datenmigrationsszenario ist in [39] zu finden.

Basierend auf den Evaluationsergebnissen wird im Rahmen dieser Arbeit die kontinuierliche Verbesserung der Methodik und des Tools von Bachmann angestrebt.

1.3. Evaluationsplan

Um ein besseres Verständnis bezüglich der Struktur der Evaluation in dieser Arbeit zu erzielen, werden in diesem Abschnitt die einzelnen Evaluationsschritten vorgestellt. Die Durchführung der Evaluation findet in zwei Iterationen statt.

[1] NovaTec: http://www.novatec-gmbh.de

1.3.1. Iteration 1

Wie anhand der Abbildung 1.1 ersichtlich wird, werden während der ersten Iteration die zwei Datenbanken von NovaERM in die NovaTec Cloud (siehe Abschnitt 2.9) migriert. Die Migration erfolgt mit Hilfe der Migrationsmethodik und dem Tool von Bachmann.

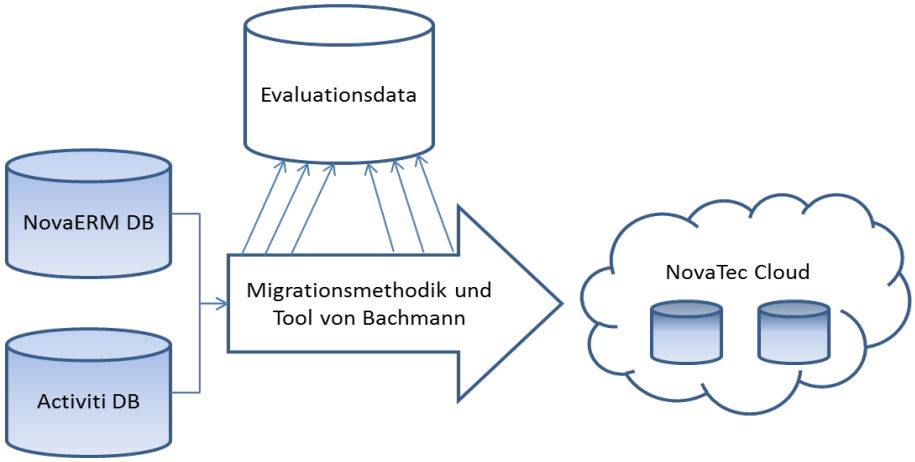

Abbildung 1.1.: Evaluation Iteration 1

1.3.2. Iteration 2

Diese Iteration besteht aus zwei Schritten. Der erste Schritt wiederholt die im vorherigen Abschnitt 1.3.1 beschriebe Migration, aber mit Amazon EC2 als Cloud Anbieter (siehe Abbildung 1.2).

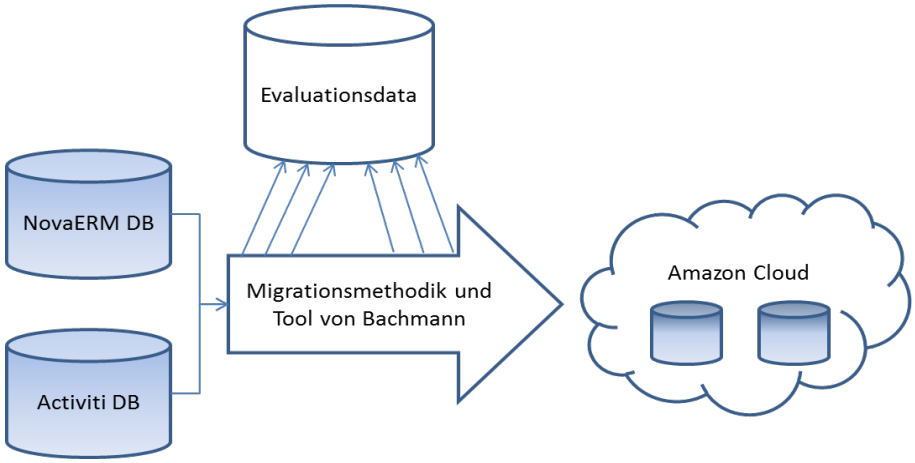

Abbildung 1.2.: Iteration 2: Schritt 1

Im zweiten Schritt wird die phasengetriebene Migrationsmethodik von AWS verwendet (siehe Abschnitt 2.10.1). Diese bietet einen Ansatz, um eine Anwendung in die Cloud zu migrieren. Während diesem Iterationsschritt wird die Datenmigrationsphase der AWS mit der Migrationsmethodik von Bachmann ersetzt, da diese im Laufe dieser Arbeit zu evaluieren ist. Außerdem fokussiert sich die Migrationsmethodik von Bachmann nur auf die Migration der DBL und nicht auf die gesamte Anwendung. Eine detaillierte Beschreibung dieses Szenarios ist im Abschnitt 4.5.3 zu finden.

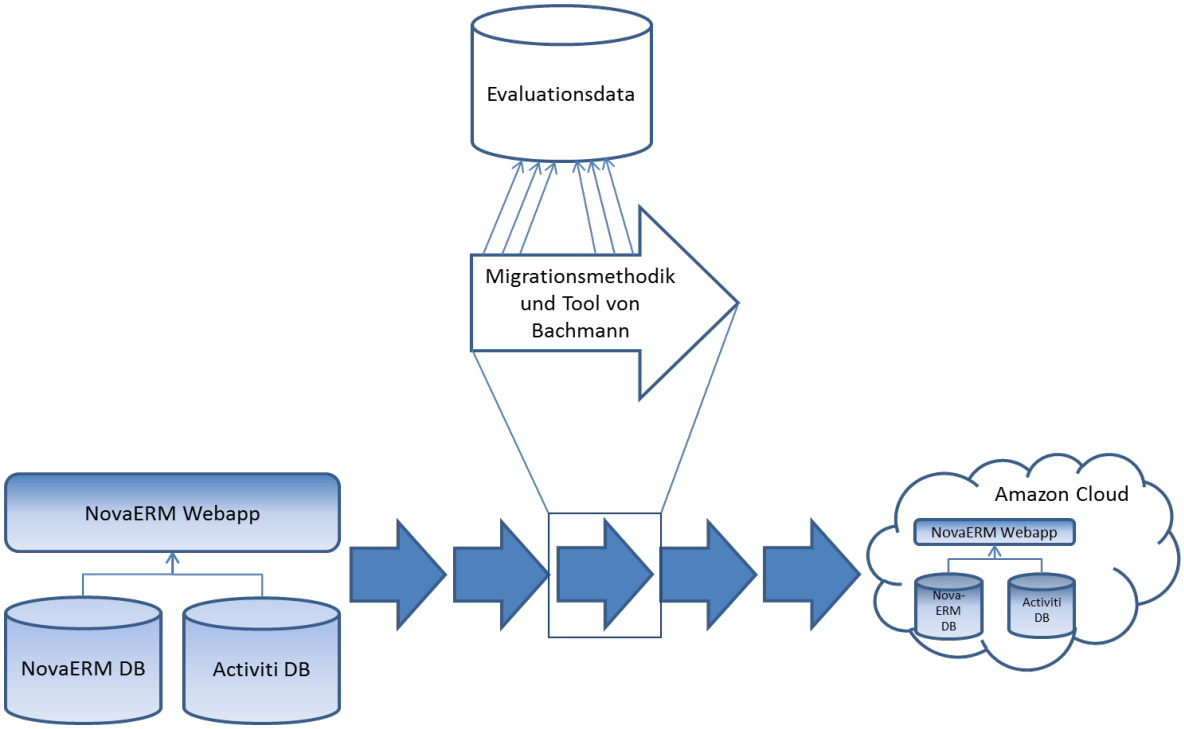

Abbildung 1.3.: Iteration 2: Schritt 2

1.4. Gliederung

In diesem Abschnitt wird die Gliederung dieser Arbeit beschrieben und eine Zusammenfassung der einzelnen Kapitel gegeben.

Das erste Kapitel dieser Arbeit beinhaltet die Motivation und die Beschreibung der Problemstellung und Zielsetzung sowie die Repräsentation des Evaluationsplanes. Das zweite Kapitel stellt eine Beschreibung der verwandten Themengebiete sowie der Tools und Methodiken dar, die während unserer Arbeit zum Einsatz kamen. In Kapitel 3 wird ein Überblick über bereits existierende Ansätze für die Durchführung von Evaluationen von Prozessen und Tools sowie die dafür benötigten Metriken gegeben. Des Weiteren wird in Kapitel 3 unser Beitrag im Rahmen dieser Arbeit in Bezug auf die existierenden Arbeiten positioniert.

Nachdem die Evaluationsmethoden und Metriken in Kapitel 3 beschrieben wurden, werden in Kapitel 4 das Ziel unserer Evaluation und eine umfassende Beschreibung ihrer einzelnen Schritte repräsentiert. Anschließend werden die Szenarien für die Durchführung der Migration vorgestellt.

In Kapitel 5 sind die Implementierungsaspekte dieser Arbeit näher beschrieben und erläutert. Die Durchführung der Migration und die dabei erhobenen Daten sind in Kapitel 6 beschrieben. In Kapitel 7 werden diese Daten analysiert. Weiterhin beinhaltet Kapitel 7 Verbesserungsvorschläge und eine Beschreibung der aufgetretenen Herausforderungen und wie diese gemeistert werden können. Abschließend werden die Ergebnisse dieser Arbeit in Kapitel 8 zusammengefasst und ein Ausblick gegeben.

1.5. Abkürzungsverzeichnis

Dieser Abschnitt enthält eine Liste der Abkürzungen, die in dieser Arbeit verwendet werden.

Amazon EC2 Amazon Elastic Compute Cloud
AMI Amazon Machine Images
APM Application Performance Management
AWS Amazon Web Services
BPM Business Process Management
BPMS Business Process Management Systems
CDDL Common development and distribution license
CLI Command Line Interface
CMMI Capability Maturity Model Integration
CMR Central Measurement Repository
CSI Continual Service Improvement
DAL Datenbank Zugriff Schicht
DB Datenbank
DBL Datenbank Schicht
DeGEval Gesellschaft für Evaluation
EJB Enterprise JavaBeans
GUI Graphical User Interface
HTML Hypertext Markup Language
HTTP HyperText Transfer Protocol
HTTPS HyperText Transfer Protocol Secure
IaaS Infrastructure as a Service
IEC Internationale elektrotechnische Kommission
ITIL Information Technology Infrastructure Library
ISO Internationale Organisation für Normung
ITSM IT Service Management
JDBC Java Database Connectivity
JDK Java Development Kit

JVM Java Virtual Machine
LAN Local Area Network
LOC Lines of Code
KLOC 1000 Lines of Code
KPI Key Performance Indicator
MTTF Mean time to failure
NIST National Institute of Standards and Technology
NTRP NovaTec Referenzplattform
OSGi Open Services Gateway initiative
PaaS Platform as a Service
RAM Random-Access-Memory
RDS Relational Database Service
RDBMS Relational Database Management System
RHEL Red Hat Enterprise Linux
SaaS Software as a Service
SDLC Systems Development Life Cycle
SEI Software Engineering Institute
SQL Structured Query Language
SQuaRE Software product Quality Requirements and Evaluation
SSH Secure Shell
UI User Interface
VM Virtuelle Maschine

2. Grundlagen

Im folgenden Kapitel werden die Grundlagen unserer Arbeit vorgestellt auf die später aufgebaut wird.

2.1. Applikationsarchitektur im 3-Schichten Modell

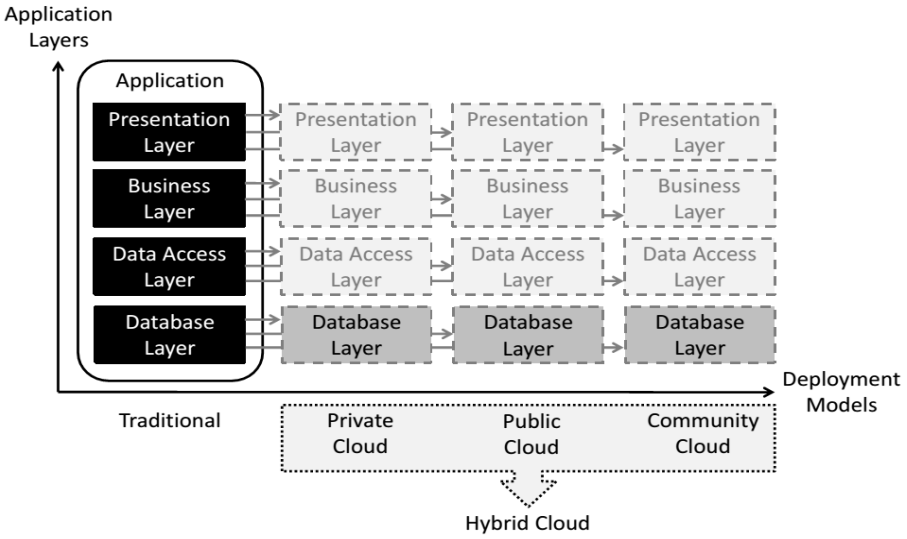

Abbildung 2.1.: Überblick über den Zusammenhang von Cloud Deployment Modellen und Schichten der Anwendungsarchitektur [38]

Im Softwareengineering wird der Aufbau einer Anwendung in mehrere Schichten aufgeteilt. Somit werden sowohl die Komplexität als auch die Abhängigkeiten zwischen die einzelnen Komponenten stark reduziert. Eine typische Architektur einer Anwendung besteht aus drei Schichten - Präsentationsschicht, Anwendungsschicht und Datenschicht, wobei die Datenschicht weiterhin in mehrere Schichten aufgeteilt werden kann (siehe Abbildung 2.1).

- Die *Präsentationsschicht* ist für die Darstellung verantwortlich und befasst sich mit der Interaktion mit den Benutzern.
- Die *Anwendungsschicht* realisiert die Anwendungslogik.
- Die *Datenschicht* kapselt den Zugriff auf die Daten sowie die dazugehörigen Technologien. Des Weiteren wird oft die Datenbankschicht in Datenbank Zugriff Schicht (DAL) und DBL aufgeteilt. Die DAL beinhaltet die Funktionalität, die für den Zugriff auf

die Datenbank verantwortlich ist, während die DBL die Funktionalität bezüglich der Persistenz und der Datenmanipulation beinhaltet.

Weiterhin ermöglicht die Aufteilung einer Anwendung in Schichten, das Einführen neuer Schichten zwischen den alten. Somit kann sowohl die Funktionalität einer Anwendung erweitert als auch eine Skalierung ermöglicht werden. Als Nachteil können die Latenzen angesehen werden, die solche eine Aufteilung mit sich bringt, vorallem dann, wenn die Kommunikation zwischen die einzelnen Schichten über das Netzwerk erfolgt [39].

2.2. Cloud Computing

Der National Institute of Standards and Technology (NIST) [2] definiert den Begriff des Cloud Computing als:

> *"... is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [31]."*

Weiterhin zeichnet sich Cloud Computing durch Charakteristiken wie On-demand self-service, Broad network access, Resource pooling, Rapid elasticity und Measured Service aus [31].

Eine weitere große Rolle spielen die so genannte Service Modelle. Bei Cloud Computing wird zwischen den folgenden drei unterschieden [31]:

- *Infrastructure as a Service (IaaS)* - Dieses Modell stellt grundlegenden Ressourcen wie Rechenleistung, Storage oder Netzwerk zur Verfügung, wobei der Anwender in der Lage ist Betriebssysteme bzw. Anwendungen zu installieren und zu verwalten. Einen Zugriff auf die grundlegende Infrastruktur ist dem Benutzer nicht gestattet.
- *Platform as a Service (PaaS)* - Dieses Modell ermöglicht dem Anwender eigene Anwendungen auf der bereitgestellten Cloud Infrastruktur hochzuladen oder zu erstellen und dann schlussendlich auszuführen. Die Anwendungen müssen den vom Cloud Anbieter zur Verfügung gestellten Ressourcen wie Programmiersprachen, Bibliotheken, Services und Tools unterstützt werden. Einen Zugriff auf die grundlegende Infrastruktur, Betriebssystem oder Software ist dem Benutzer nicht gestattet.
- *Software as a Service (SaaS)* - Bei diesem Service Modell werden den Benutzern Anwendungen vom Cloud Anbieter bereitgestellt. Diese sind entweder über eine eigene Weboberfläche oder über ein Programminterface zu erreichen. Den Benutzer ist nicht gestattet die grundlegende Infrastruktur einschließlich des Netzwerks, der Server, des Betriebssystems oder des Speichers zu verwalten.

Die bisher beschriebenen Service Modelle können in Form von unterschiedlichen Deployment-Modellen genutzt werden [31]:

- *Private Cloud* - Bei diesem Deployment Modell ist die Cloud Infrastruktur für die ausschließliche Verwendung von einer einzigen Organisation bereitgestellt. Die Infrastruktur kann sowohl on-premise als auch off-premise betrieben werden.
- *Community Cloud* - Die Cloud Infrastruktur bei dem *Community Cloud* Modell wird für die Verwendung von einer bestimmten Gemeinschaft einer Organisation, die gemeinsame Anliegen haben, bereitgestellt. Die Infrastruktur kann sowohl on-premise als auch off-premise betrieben werden.
- *Public Cloud* - Die Infrastruktur bei dem *Public Cloud* Modell ist für die offene Verwendung durch die Allgemeinheit bereitgestellt. Diese kann nur on-premise existieren.
- *Hybrid Cloud* - Die Cloud Infrastruktur ist eine Zusammensetzung aus zwei oder mehr verschiedenen Cloud Infrastrukturen (Private, Community oder Public) und wird mit Hilfe von verschiedenen Technologien zusammengesetzt.

2.3. NovaERM

Wie durch das Thema unserer Arbeit definiert, basiert unsere Evaluation auf einer Fallstudie aus der Industrie. NovaERM stellt die Applikation dar, die in die Cloud zu migrieren ist. Deswegen werden wir auf sie in diesem Abschnitt näher eingehen und ihre einzelnen Komponenten detailliert beschreiben.

Bei NovaERM handelt es sich um ein Tool, mit dessen Hilfe die Geschäftsprozesse der Firma NovaTec [3] automatisiert und unterstützt werden sollen. Da das Tool ständig weiterentwickelt wird, liegt der Fokus der Funktionalität aktuell auf der Automatisierung des Einstellungsprozesses eines Mitarbeiters und der Erstellung eines Kompensationsplans [29].

2.3.1. NovaERM Architektur

Da während unserer Evaluation basierend auf dem Migrationsszenario sowohl einzelne Komponenten als auch die gesamte Anwendung in die Cloud zu migrieren sind, werden wir in diesem Abschnitt die Systemarchitektur des Tools detailliert erläutern. Eine Darstellung der Systemarchitektur ist in Abbildung 2.2 zu sehen.

Wie anhand der Abbildung 2.2 ersichtlich wird, setzt sich NovaERM aus mehreren Komponenten zusammen. Im Allgemeinen können diese in drei Teile zusammengefasst werden - *NovaERM Webapplikation*, *Activiti Datenbank* und *Novaerm Datenbank*. Diese drei Komponenten sind lose gekoppelt und können deshalb einzeln migriert werden. Im Gegensatz zu den zwei Datenbanken, die direkt in der Systemarchitektur zu erkennen sind, setzt sich die *NovaERM Webapplikation* aus mehreren Komponenten zusammen. Diese kommunizieren miteinander mit Hilfe der Services wie z.B. dem Bewerbung Task Service (siehe Abbildung 2.2).

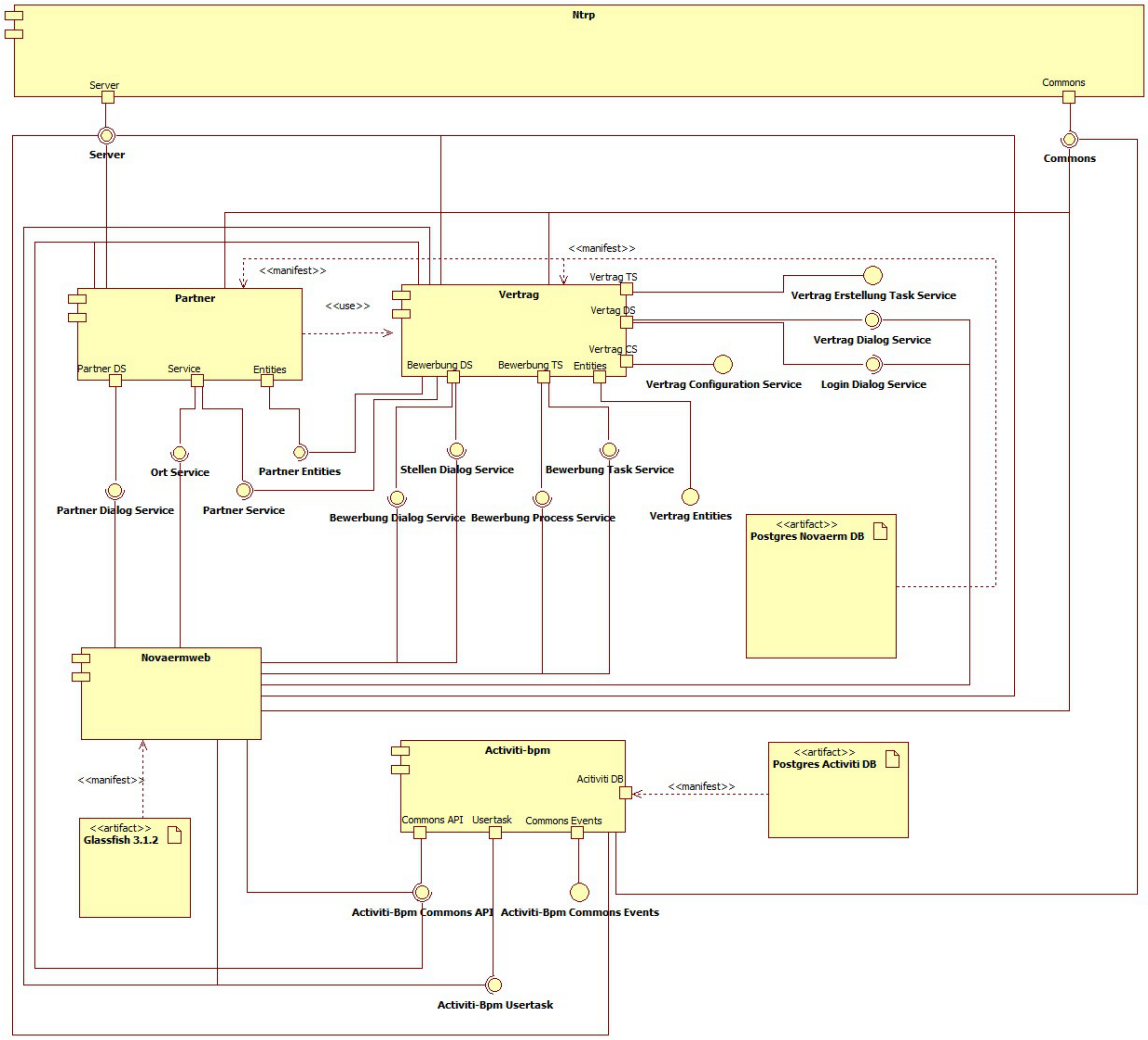

Abbildung 2.2.: NovaERM Systemarchitektur

Die *NovaTec Referenzplattform (NTRP)* Komponente stellt die Grundlagen der NovaERM Applikation dar und enthält Komponenten, die nicht anwendungsspezifisch sind. Einen weiteren wichtigen Teil der Webanwendung stellen die Komponenten *Partner* und *Vertrag* dar. Durch die *Partner* Komponente wird entweder ein Bewerber oder ein Mitarbeiter repräsentiert. Des Weiteren wird durch die *Vertrag* Komponente, der allgemeinen Bewerbungsprozess sowie das Abschließen eines Vertrags abgebildet. Der gesamte Bewerbungsprozess vom Eintritt der Bewerbung bis hin zum Abschließen eines Vertrags, wird von einer Business Process Management (BPM) Plattform namens Activiti [1] durchgeführt. Diese wird durch die Komponente *Activiti-bpm* dargestellt. Das letzte Element der NovaERM Webapplikation wird durch die *Novaermweb* Komponente abgebildet. Novaermweb kümmert sich um die Interaktion mit den Benutzern und implementiert somit die grafische Oberfläche der NovaERM Applikation. Alle bis hier beschriebenen Komponenten sind Teil der NovaERM Webapplikation.

[1] Activiti BPM Platform: http://www.activiti.org/

Die anderen zwei Komponenten der Applikation sind durch die zwei Datenbanken - *Novaerm* und *Activiti* dargestellt. In der Novaerm Datenbank werden alle spezifischen Daten gespeichert, die durch die Komponenten *Partner* und *Vertrag* generiert werden. Im Gegensatz dazu werden in der *Activiti* Datenbank nur wenig Daten gespeichert, die von der Activiti BPM Plattform generiert werden.

Alle bisher beschriebenen Teile von NovaERM sowie die dazugehörigen Software Komponenten, der Applikationserver *Glassfish* und die Datenbank *PostgreSQL*, stellen das Grundsystem dar, das zu migrieren ist, wobei der Fokus auf der Migration der Datenbank liegt. Wie die einzelnen Komponenten von NovaERM migriert werden und welche Migrationsszenarien durchgeführt werden, ist im Abschnitt 4.5 detailliert beschrieben.

2.4. automaIT

Bei automaIT [26] handelt es sich um ein Java-basiertes Provisionierungswerkzeug, das von der Firma NovaTec [3] entwickelt wurde. Mittels automaIT lassen sich manuelle Abläufe automatisieren und aufwändige Aufgaben des Life-Cycle-Managements steuern. Das Kernprinzip von automaIT verfolgt einen modellbasierten Ansatz. Weiterhin wird zwischen einem Modell, in dem der Zielzustand beschrieben wird und der realen Welt, die durch Provisionierung des Modells entsteht, unterschieden [30]. Die definierten Modelle, verfolgen einen objektorientierten Ansatz. Dies ermöglicht, dass die Vorteile der Vererbung, der Polymorphie und der Datenkapselung zur besseren Strukturierung der Modelle genutzt werden können [30]. Des Weiteren wird dadurch einer bessere Wartbarkeit und Wiederverwendbarkeit erreicht [30]. Um sicherzustellen, dass die notwendige Voraussetzungen zur Provisionierung eines IT-Services geschaffen sind, ermöglichen die Modellen Abhängigkeiten zu definieren. Dies führt zur Verhinderung von Inkonsistenzen bei der Installation und De-Installation eines Services.

Wie anhand der Abbildung 2.3 zu erkennen ist, verfügt automaIT sowohl über ein Web Frontend zur einfachen Bedienung als auch über ein Command Line Interface (CLI). Des Weiteren ist es ersichtlich, dass mittels automaIT gleichzeitig mehreren Agents verwaltet werden können. Ein Agent ist eine serverseitige Applikation, die die Kommunikation zwischen dem automaIT Server und dem Zielsystem ermöglicht. Außerdem dient der Agent zum Ausführen von Befehlen, die vom Server vorgegeben werden. Die Kommunikation zwischen allen Komponenten erfolgt grundsätzlich verschlüsselt über HTTPS/SSH.

Im Rahmen dieser Arbeit wird automaIT im Rahmen der Migration von NovaERM in die Cloud, basierend auf verschiedenen Migrationsszenarien, zur Provisionierung verwendet. Die einzelnen Migrationsszenarien sind im Abschnitt 4.5 beschrieben.

Außerdem wird automaIT bei der Vorbereitung jedes Migrationsszenarios (siehe Abschnitt 4.5), zum Aufsetzen sowohl der lokal gehosteten Komponenten als auch zum Bestimmen der Baseline, verwendet.

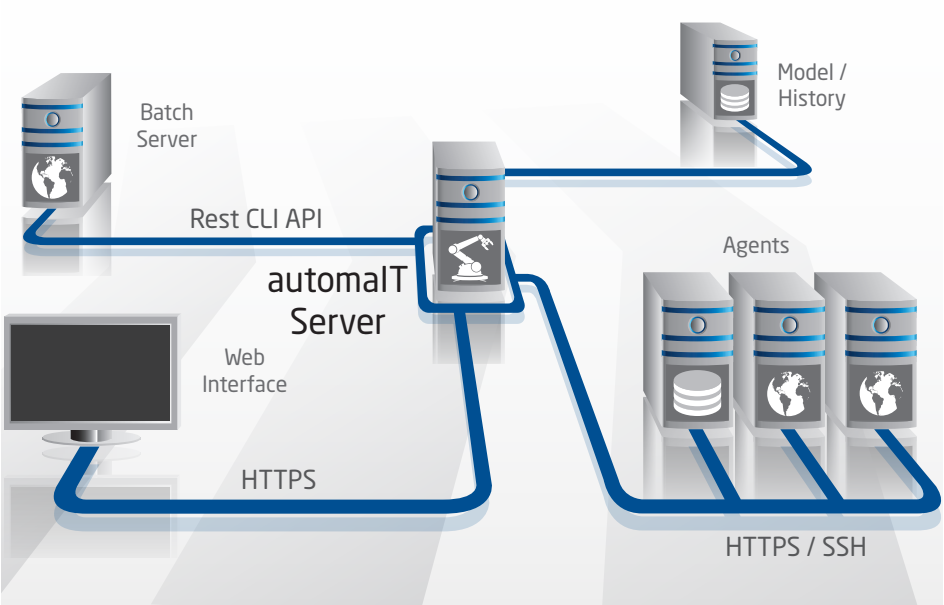

Abbildung 2.3.: automaIT Systemarchitektur [30]

2.5. inspectIT

inspectIT ist eine Client-Server Anwendung, die von der Firma NovaTec [3] entwickelt wurde. Diese eignet sich sowohl zur Performance Analyse als auch für Überwachung von Applikationen während der Ausführung [28]. inspectIT integriert sich nahtlos und transparent mit Hilfe von zwei Startup Parameter in der zu untersuchenden Anwendung. Ein Beispiel Aufbau von inspectIT ist in der Abbildung 2.4 zu sehen.

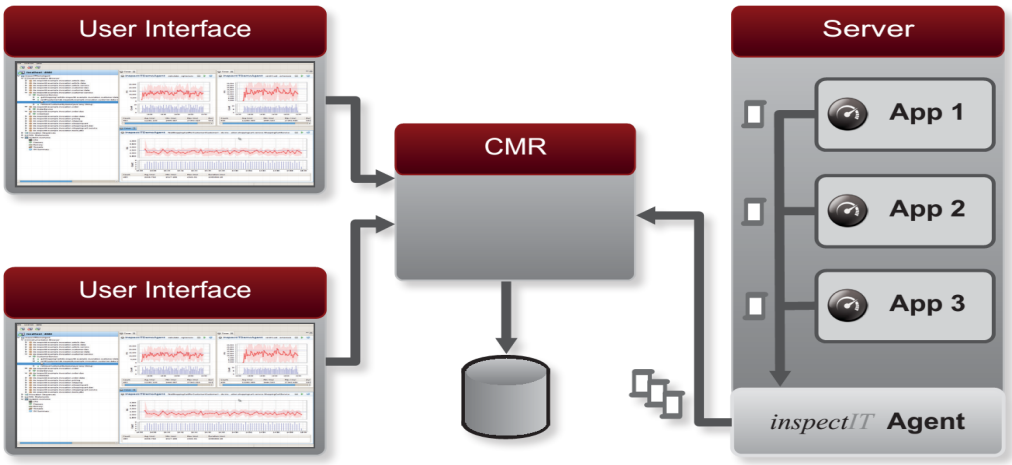

Abbildung 2.4.: inspectIT Systemarchitektur [28]

Wie anhand der Abbildung 2.4 zu erkennen ist, besteht inspectIT aus insgesamt drei Komponenten [27]:

1. *inspectIT Agent*: Der Agent stellt eine serverseitige Anwendung dar, die sich transparent in die JVM/Anwendung integriert und trägt zu deren Überwachung oder Analyse bei. Jeder Agent besitzt eine Konfigurationsdatei. In dieser werden Einstellungen wie z.B. der Name des Agents und die Lokation des Central Measurement Repository (CMR) gespeichert. Des Weiteren werden in der Konfigurationsdatei alle benötigten Sensoren definiert. Außerdem werden alle von dem Agent gesammelten Daten an den CMR Server zur Analyse gesendet.
2. *Central Measurement Repository (CMR)*: Das CMR agiert als Serverkomponente in der inspecIT Architektur. Das CMR nimmt alle Messdaten, die vom Agent zur Verfügung gestellt werden, entgegen und stellt diese wiederum dem Client zur Verfügung. Des Weiteren ist es möglich, dass ein CMR gleichzeitig die Daten von mehreren Clients entgegennehmen kann.
3. *Client*: Der Client stellt dem Benutzer eine graphische Benutzeroberfläche zur Verfügung und wird zur Analyse und zum Monitoring der Anwendung verwendet. Jeder Client kann die Daten von mehreren CMRs gleichzeitig erfassen.

Aus Performance Gründen und um die Messergebnisse nicht zu beeinflussen, ist es zu empfehlen die einzelnen Komponenten von inspectIT auf unterschiedlichen Maschinen aufzusetzen. Vor allem ist die Hardwaretrennung des *inspectIT Agents* und der anderen zwei *inspectIT* Komponenten sehr wichtig.

Nachdem wir die einzelnen Komponenten von inspectIT erläutert haben, werden wir im folgenden die einzelnen Ansätzen zur Sammlung von Anwendungsdaten näher erläutern. Wie bereits beschrieben, liefert der inspectIT Agent eine Konfigurationsdatei, in der Seonsoren definiert werden. Die Sensoren sammeln Informationen aus der jeweiligen Anwendung. inspetIT definiert sechs Sensoren, die im Folgenden detailliert beschrieben sind.

2.5.1. Timer Sensor

Mittels des *Timer Sensors* werden die Ausführungszeiten der Methoden einer Anwendung überwacht. Der Sensor liefert aggregierte Durchschnittswerte, die eine Methode beansprucht hat. Somit können sowohl die reine Methoden-, als auch die CPU-Zeit erfasst werden. Die Informationen, geben einen Aufschluss darüber, ob der Grund für die lange Ausführungsdauer einer Methode die eigene komplexe Logik war oder ob eine Methode z.B. auf eine parallele Verarbeitung warten musste [28].

2.5.2. Datenbank Sensor

Die Ausführung von Datenbankabfragen stellt oft den Grund für Performance Probleme in einer Anwendung dar. Der Datenbank Sensor hilft bei der Analyse aller durchgeführten Datenbankabfragen und liefert Informationen über deren Struktur und Dauer. Dadurch können Structured Query Language (SQL)-Probleme identifiziert, analysiert und behoben werden [28].

2.5.3. Exception Sensor

Wie der Name des Sensors nahelegt, handelt es sich um einen Sensor, der Informationen darüber liefert, ob und an welcher Stelle Fehler aufgetreten ist. Somit kann die Stabilität einer Anwendung verbessert werden [28].

2.5.4. Plattform Sensor

Der Plattform Sensor liefert detaillierte Auskunft über den Zustand des Systems zur Zeit der Ausführung einer Anwendung. Mit Hilfe dieses Sensors können Systemressourcen wie CPU-Auslastung, Speicherauslastung, Threads, Klassen und JVM Parameter überwacht werden. Somit können Performanceprobleme erkannt werden, die z.B. auf eine Systemressource ausgelöst werden [28].

2.5.5. HTTP Sensor

Mit Hilfe des *HTTP Sensors* werden aggregierte Durchschnittswerte für die Aufrufe von Webseiten geliefert. Die Aggregierung erfolgt anhand der aufgerufenen Adresse. Dieser Sensor ist vor allem für die Überwachung von Webanwendungen interessant [28].

2.5.6. Invocation Sequence Sensor

Alle bisher beschriebenen Sensoren bieten eine eigenständige Ausführung. Der *Invocation Sequence Sensor* ermöglicht es, dass die Messpunkte alle anderen Sensoren komplett transparent in einen konkreten Aufrufspfad zu verbinden. Somit wird eine Voraussetzung geschaffen einzelnen Aufrufsequenzen zu analysieren [28].

Im Rahmen unserer Arbeit wird inspectIT zur Performance Messung von NovaERM verwenden. Um die Wiederholbarkeit und Konsistenz der Tests sicherzustellen, werden sie mit dem Softwaretest Tool seleniumHQ durchgeführt (siehe Abschnitt 2.6). Anhand der größten Anzahl von Sensoren und den begrenzten Zeitraum, der uns während unserer Arbeit zur Verfügung steht, werden wir uns auf den *Timer Sensor*, *Datenbank Sensor* und *HTTP Sensor* für die Performance konzentrieren.

2.6. seleniumHQ

seleniumHQ [4] ist ein Softwaretest Tool, das auf reines Hypertext Markup Language (HTML) und JavaScript basiert. Es ermöglicht die automatisierte Durchführung von User Interface (UI) Tests von Webanwendungen in jeder Phase des Software Lebenszyklus. Das Tool interagiert mit dem Browser, um Benutzerinteraktionen nachzumachen.

Im Rahmen unserer Arbeit wird das Tool als Testwerkzeug verwenden, um Last während der Performance Messung zu erzeugen. seleniumHQ ermöglicht uns die Interaktion mit NovaERM, basierend auf den vom NovaERM Entwicklungsteam zur Verfügung gestellten Testcases, beliebig oft zu wiederholen. Dies verhindert die Verfälschung der Tests, da sie immer in derselben Reihenfolge ausgeführt werden.

2.7. Migrationsmethodik

Basierend auf dem Thema unserer Arbeit, stehen die Migrationsmethodik und das Tool von Bachmann [39] als Grundlage für unsere Evaluation. Im Laufe dieses Abschnitts werden wir die Migrationsmethodik von Bachmann vorstellen und die einzelnen Schritte erläutern.

Bachmann entwickelte seine Methodik, indem er die Methodik für Migrationsprojekte von Laszewski [40] aufgriff und weiter entwickelte. Dadurch entstand eine Anbieter- und Datenbanktechnologie unabhängigen sukzessive Migrationsmethodik. Weiterhin schließt seine Methodik neben dem teilweise auch von Laszewski vorgestellten reinen Outsourcing des DBL, neun zusätzliche Migrationsszenarien und zwei weitere Quell- und Zieldatenspeichertypen ein [39]. Die Methodik beinhaltet insgesamt sieben Phasen, die in den folgenden Abschnitten kurz erläutert werden.

2.7.1. Auswahl des Migrationsszenarios

Die erste Phase der Migrationsmethodik von Bachmann ist in zwei Schritte unterteilt. Im ersten Schritt dieser Phase muss ein Migrationsszenario ausgewählt werden. Bachmann beschreibt insgesamt zehn Szenarien in seiner Methodik wie z.B. *Plain Outsourcing*, *Geograhic Replication*, *Sharding*, *Off-Loading of Peak Loads (Cloud Burst)*, *Backup*, *Archive* [39]. Nachdem ein passendes Szenario ausgewählt wurde, kann im zweiten Schritt die Migrationsstrategie definiert werden. Diese wird durch Eigenschaften wie z.B. *Source/Target Data Store Type*, *Live/Non-Live Migration*, *Migration Durability*, *Migration Degree* spezifiziert. Diese Eigenschaften helfen dem Benutzer mögliche Konflikte des ausgewählten Szenario zu identifizieren und gegebenenfalls ein anderes auszuwählen.

2.7.2. Beschreibung der gewünschten Cloud Data Hosting Solution

In der zweite Phase der Migrationsmethodik können alle funktionalen und nicht-funktionalen Anforderungen im Bezug auf den Zieldatenspeicher spezifiziert werden. Diese sind in verschiedene Kategorien aufgeteilt, wobei jede Kategorie mehrere Eigenschaften enthalten kann. Eine Übersicht über die wichtigsten Kategorien und deren Eigenschaften ist in der Tabelle 2.1[2] zu sehen.

[2]Die Bezeichnungen aller Kategorien und Eigenschaften sind von der Methodik übernommen.

Kategorien	Eigenschaft	Werte
Scalability	Degree of Automation	Manual, Automated
	Type	Horizontal, Vertical
	Degree	Virtually Unlimited, Limited
	Time to Launch new Instance	None, Duration in Minutes
Availability	Replication	Yes, No
	Replication Type	Master-Slave, Master-Master
	Replication Method	Synchronous, Asynchronous
	Replication Location (Data Center)	Same , Different (Same Region)
	Automatic Failover	Yes, No
	Degree	99.9%, 99.999%
Security	Storage Encryption	Yes, No
	Transfer Encryption	Yes, No
	Firewall	Yes, No
	Authentication	Yes, No
	Confidentiality	Yes, No
	Integrity	Yes, No
	Authorization	Yes, No
Interoperability	Data Portability	None, Import, Export, One-Way-Synchronization
	Data Exchange Format	XML, JSON, Proprietary
	Storage Access	SOA, REST-API, SQL, Proprietary
	Object Relational Mappers	JPA, JDO, LINQ
	Migration & Deployment Support	Yes, No
	Supported IDE	Eclipse, NetBeans, IntelliJ IDEA
	Developer SDKs	Java, .Net, PHP, Ruby
Storage	Storage Type	RDBMS, NoSQL
CAP	Consistency Model	Strong, Weak, Eventual
	Availability in Case of Partitioning	Available, Not Available
Cloud Computing	Service Model	IaaS, PaaS, SaaS
	Deployment Model	Private-, Public-, Community-, Hybrid Cloud
Multi-tenancy	Multi-tenancy Capability	Yes, No
	Multi-tenancy Type	Multiple Instances, Native Multi-Tenancy

Tabelle 2.1.: Cloud Data Hosting Solution Eigenschaften [39]

2.7.3. Auswahl des Cloud Data Stores

Anhand der bereits ausgewählten Kriterien im zweiten Schritt der Migrationsmethodik (siehe Abschnitt 2.7.2), werden in diesem Schritt unterschiedliche Cloud Data Stores empfohlen. Die Empfehlung erfolgt durch die Abbildung der bereits ausgewählten Kriterien auf die Kriterien, die vom Cloud Data Stores abgedeckt werden. Dieser Schritt setzt voraus, dass Cloud Data Stores zuvor definiert wurden.

2.7.4. Beschreibung des Quelldatenspeichers

Um möglichen Konflikten zu dem Zieldatenspeicher zu erkennen, werden die funktionalen und nicht-funktionalen Anforderungen des Quelldatenspeichers während dieses Migrationsschritts spezifiziert wie z.B. des Typs des Speichers (RDBMS, NoSQL, Blob Store) oder seine Lage (On-premise, Off-premise).

2.7.5. Identifizierung von Pattern, um mögliche Migrationskonflikte zu erkennen

In dieser Phase der Migrationsmethodik werden mögliche Konflikte zwischen dem Zieldatenspeicher und dem Quelldatenspeicher identifiziert und eine mögliche Lösung empfohlen. Die Erkennung basiert auf den ausgewählten Eigenschaften der Ziel- und Quelldatenspeicher in den Schritten drei und vier der Methodik. Des Weiteren werden Pattern vorgeschlagen, als Lösung wiederkehrender Probleme oder um fehlende Funktionalität bei dem Zieldatenspeicher nachzubilden. Eine Liste von möglichen Cloud Data Pattern ist in [37] zu finden.

2.7.6. Refactoring der Applikation Architektur

Unabhängig von der Auswahl des Migrationsszenarios hat jede Migration der DBL Auswirkungen auf die anderen Anwendungsschichten. Deswegen beinhaltet dieser Phase zusätzlich Vorschläge zur Anpassung höherer Anwendungsschichten, die zu einem notwendigen Refactoring der Anwendung führen können. Der Fokus dabei liegt auf der Anpassung der Netzwerkschicht und der DAL. Weiterhin beinhaltet dieser Phase Vorschläge für die Adaption der Anwendungslogik. Vor allem sind diese von großer Bedeutung, wenn es sich bei der Migration um einen Produktwechsel handelt (von RDBMS zu NoSQL oder z.B. von MySQL zu PostgreSQL) oder wenn die Daten in der Datenbank sowie die Kommunikation zwischen den DBL und die Applikation anonymisiert werden müssen.

2.7.7. Datenmigration

Die Datenmigrationsphase ist die letzte Phase der Methodik. Sie beinhaltet die Konfiguration der Verbindungsdaten für den Quell- und Zieldatenspeicher wie z.B. Benutzername, Password, Datenbankname, Datenbanklokation usw.. Außerdem werden während dieser Phase möglichen Inkonsistenzen zwischen dem Quell- und Zieldatenspeicher mit Hilfe von Adaptern behoben. Diese Anpassungen der Exportdaten ermöglichen z.B. eine Migration zwischen einer MySQL Datenbank als Quelldatenspeicher und einem AWS Relational Database Service (RDS) mit einem Oracle Engine als Zieldatenspeicher.

2.7.8. Weitere Migrationsschritte

Wie bereits am Anfang dieses Abschnitts beschrieben, basiert die Migrationsmethodik von Bachmann auf die Methodik von Laszewski [40]. Eine Abbildung der Migrationsmethodik von Bachmann auf die von Laszewski ist in der Abbildung 2.5 zu sehen.

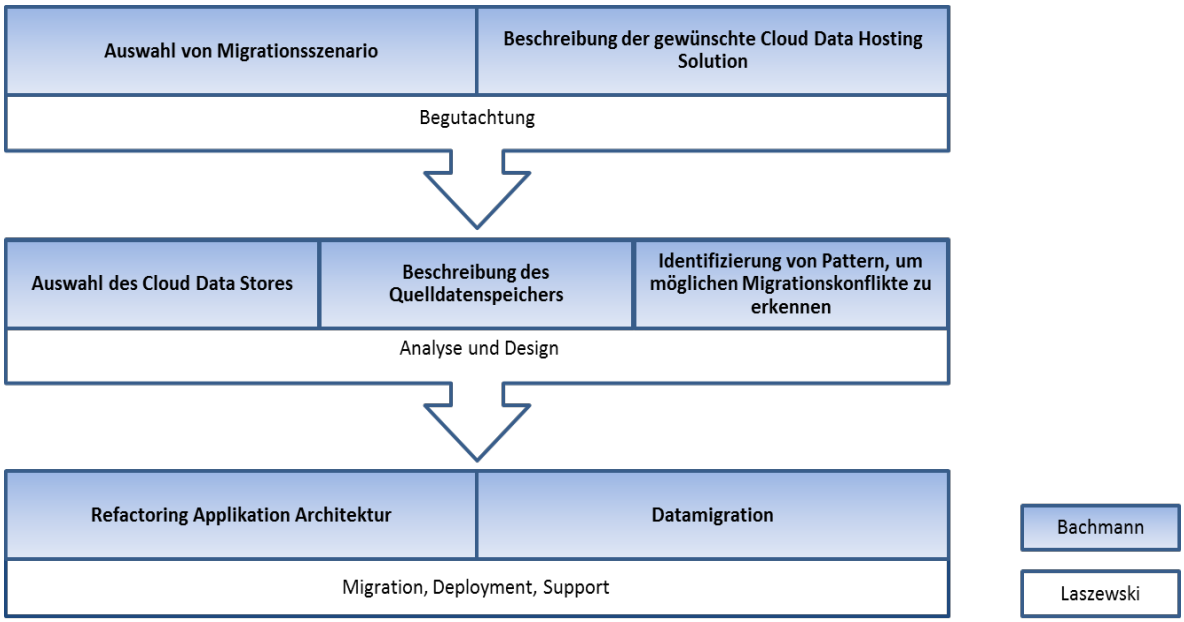

Abbildung 2.5.: Abbildung Migrationsmethodik Bachmann - Laszewski

Im Gegensatz zu Laszewski, bietet Bachmann in seiner Methodik keine Unterstürzung für die Test- und Optimierungsphase. Im Vergleich zu der Optimierungsphase, die während unserer Evaluation nicht von große Bedeutung ist (siehe Abschnitt 4.6.1), ist die Testphase sehr wichtig. Im Laufe dieser Phase wird die Funktionalität des migriertes System getestet und auftretende Fehlern behoben. Deswegen wird diese Phase während unserer Evaluation berücksichtigt und evaluiert.

2.8. Cloud Datamigrations-Tool

Das Cloud Datamigrations-Tool wurde von Thomas Bachmann im Rahmen seiner Diplomarbeit [39] entwickelt. Es unterstützt die Migration der Datenschicht in die Cloud, indem sowohl die Vorbereitung der Migration als auch die Migration selbst von dem Tool durchgeführt werden [39].

Da das Ziel unserer Arbeit, die Evaluation des Tools im Bezug auf die Gebrauchstauglichkeit ist, werden wir in diesem Abschnitt eine kurze Beschreibung der Hauptanforderungen des Tools geben. Eine detaillierte Beschreibung des Tools und seine Funktionen sind in der Arbeit von Bachmann zu finden [39].

Als Teil der Vorbereitung der Migration soll das Tool das Migrationsszenario mit Ausprägungen identifizieren, sowie eine passende Cloud Data Hosting Solution und passende Cloud Data Stores vorschlagen [39]. Des Weiteren sollen mögliche Konflikte zwischen den Anforderungen und den Spezifikationen der Dienste erkannt werden und Lösungen u.a. in Form von Cloud Data Patterns angeboten werden. Die Identifizierung der einzelnen Szenarien sowie das Erkennen der auftretenden Konflikte sind mit Hilfe eines Fragenkatalogs implementiert.

Im zweiten Schritt wird die Migration des Quelldatensystems durchgeführt. Das Tool unterstützt zwei Typen von Quellsystemen - MySQL und Virtuelle Maschine. Die Migration des ersten Quellsystems wird über das Kommandozeilenwerkzeug (mysql) unterstützt. Im Fall eines Abbilds einer virtuellen Maschine wird den Exportanweisungen der Virtualisierungssoftware gefolgt.

Wie bereits im Abschnitt 2.3 beschrieben, benutzt NovaERM eine PostgreSQL Datenbank als Quellsystem. Da diese von dem Tool von Bachmann nicht unterstützt wird, wird eine Erweiterung für das Tool implementiert, um die benötigte Funktionalität anzubieten. Die Details bezüglich der Implementierung sind im Abschnitt 5.3 zu finden.

2.9. NovaTec Cloud

Wie in unserem Evaluationsplan (siehe Abschnitt 1.3) beschrieben, findet unsere Evaluation in zwei Iterationen statt. Während der ersten Iteration werden die Datenbaken der Webapplikation NovaERM in eine Virtuelle Maschine migriert, die von der Firma NovaTec [3], auf ihrer internen Cloud zur Verfügung gestellt wurde. Die Migration erfolgt mittels des Tools von Bachmann (siehe Abschnitt 2.8) und entspricht dem im Abschnitt 4.5.1 beschriebenen Migrationsszenario. Die Spezifikation der VM ist in der Tabelle 2.2 zu finden.

Vendor ID	GenuineIntel
CPU model name	Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz
CPU MHz	2500.000
CPU cache size	15360 KB
Total Memory	3833 MB
Distribution ID	CentOS
Distribution Release	6.4

Tabelle 2.2.: NovaTec Cloud VM Spezifikation

2.10. Amazon Web Services

Um eine Datenbankschicht in die Public Cloud zu migrieren, benötigen wir einen passenden Cloud Anbieter. Basierend auf unseren Recherchen sowie auf der Empfehlung von Bachmann

in seiner Arbeit [39], haben wir uns für die Verwendung von Amazon Web Services (AWS) entschieden. Des Weiteren ist AWS sehr gut dokumentiert und bieten einen Support für auftretenden Problemen und Fragen.

2.10.1. AWS Migrationsmethodik

AWS hat eine phasengetriebene Migrationsmethodik entwickelt, die die Benutzer bei der Migration ihrer Anwendungen in die AWS Cloud unterstützen sollte. Die Migrationsmethodik beschreibt Schritte, Techniken und Methoden für die Provisionierung vorhandener Enterprise-Anwendungen in die AWS Cloud und wird in sechs Phasen durchgeführt, die in der Abbildung 2.6 zu sehen sind:

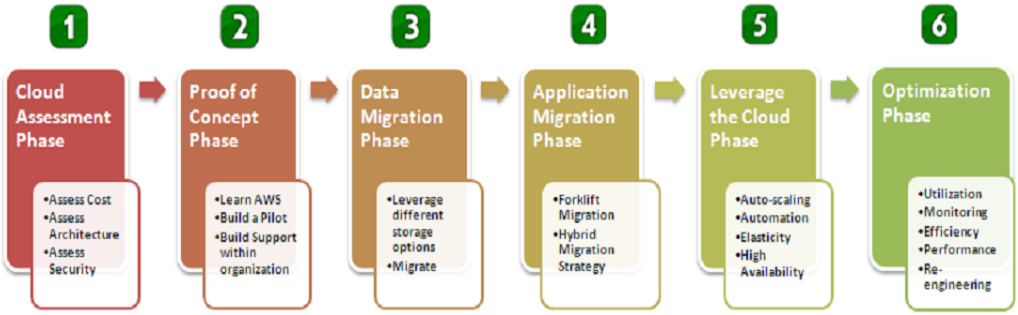

Abbildung 2.6.: AWS Phasengetriebene Migrationsmethodik [22]

Die einzelnen Phasen der Migrationsmethodik sind [22]:

- *Cloud Assessment Phase* - In der ersten Phase der Migrationsmethodik werden Informationen über die Kosten, die Architektur und die Sicherheit gesammelt und bewertet.
- *Proof of Concept Phase* - Das Ziel dieser Phase ist es, AWS kennen zu lernen, um sicherzustellen, dass die Annahmen hinsichtlich der Eignung unserer Anwendung für die Migration in die Cloud unseren Vorstellungen entsprechen.
- *Data Migration Phase* - Im laufe dieser Phase werden zwei wichtige Schritte durchgeführt. Zum Einen wird der passenden Cloud Data Speicher ausgewählt und zum Anderen werden die Daten migriert.
- *Application Migration Phase* - Das Ziel dieser Phase ist eine passende Migrationsstrategie auszuwählen sowie eine Amazon Machine Images (AMI) zu erstellen und zu konfigurieren.
- *Leverage the Cloud Phase* - Nachdem die Anwendung erfolgreich in die Cloud migriert wurde, muss die Funktionalität auf Korrektheit geprüft werden. Des Weiteren wird versucht die Elastizität sowie den Systems Development Life Cycle (SDLC) zu automatisieren.

- *Optimization Phase* - In dieser Phase werden Optimierungsschritte an der migrierten Anwendung durchgeführt, um die Kosteneinsparung zu erhöhen.

Eine detaillierte Beschreibung der einzelnen Phasen ist in [22] zu finden.

Die phasengetriebene Migrationsmethodik von AWS beschreibt eine ganzheitliche Methodik für die Migration in die Cloud, während die von Bachmann sich auf der DBL fokussiert. Da im Rahmen dieser Arbeit die Methodik von Bachmann evaluiert wird, aber für die Migration der NovaERM Webanwendung eine ganzheitliche Methodik benötigt wird, werden wir die Datenmigrationsphase von AWS mit der Migrationsmethodik von Bachmann (siehe Abschnitt 2.7) ersetzten. Somit wird überprüft, in wie weit die Methodik von Bachmann mit anderen kombinierbar ist.

2.10.2. Amazon Datenverarbeitung

In der zweiten Iteration unserer Evaluation werden wir den Cloud Anbieter wechseln und die Migration wiederholt durchführen. Wie bereits in der ersten Iteration unserer Evaluation (siehe Abschnitt 2.9), benötigen wir auch in der zweiten Rechenkapazität. Amazon Elastic Compute Cloud (Amazon EC2) stellt dem Benutzer leicht konfigurierbare virtuelle Server zur Verfügung, auf denen u.a. ein Datenbankserver betrieben werden kann [6]. Außerdem bietet Amazon EC2 die Möglichkeit unbegrenzt viele Instanzen einer virtuellen Maschine zu betreiben und somit genügend Rechenkapazität zu schaffen.

Die beschriebenen Voraussetzungen schaffen eine geeignete Grundlage für unsere Evaluation und die Durchführung der Migration. Für die Verwendung von Amazon EC2 werden auf der Seite des Cloud Anbieters [7] vier Schritten beschrieben:

- Im ersten Schritt wird als Vorlage ein vorkonfiguriertes AMI ausgewählt. Es ist auch möglich ein AMI selbst zu erstellen, welches unsere eigenen Anwendungen, Bibliotheken, Daten und zugehörigen Konfigurationseinstellungen enthält.
- Im zweiten Schritt werden die Sicherheitseinstellungen und der Netzwerkzugang der Amazon EC2 Instanz konfiguriert.
- Im vorletzten Schritt wird der Instanz-Typ ausgewählt und anschließend die Instanz gestartet. Mithilfe der Web-Service-APIs oder der verschiedenen Verwaltungstools können weitere Instanzen gestartet werden oder die vorhandenen überwacht bzw. beendet werden.
- Im letzten Schritt kann entschieden werden, ob die Instanz an mehreren Standorten ausgeführt werden soll, ob statische IP-Endpunkte für die Instanzen benutzt werden sollen oder ob ein dauerhafter Blockspeicher zu den Instanzen hinzugefügt werden soll.

3. Kenntnisstand

Dieses Kapitel gibt einen Überblick über bereits existierende Ansätze für die Durchführung einer Evaluation zur Optimierung von Prozessen und Bewertung von Softwareprodukten. Dabei soll der eigene Beitrag in den größeren Rahmen gegenwärtiger wissenschaftlicher Erkenntnisse positioniert werden.

3.1. Evaluation und Daten

Mit den steigenden Anforderungen an die Qualität der Software, um wettbewerbsfähig zu bleiben, suchen Unternehmen nach Methoden, welche die Produktivität erhöhen und die Qualität ihrer Software verbessern können. Die Evaluation hat sich als solch eine Methode in den letzten Jahren stark entwickelt und gewinnt in vielen Bereichen an Bedeutung [9]. Ihre Durchführung trägt dazu bei Ablaufprozesse effektiver zu gestalten sowie den Input effizienter einzusetzen, um den Output zu erhöhen.[34]

Die *Gesellschaft für Evaluation (DeGEval)* [12] definiert den Begriff der Evaluation als die systematische Untersuchung des Nutzens oder Werts eines Gegenstands. Die erzielten Ergebnisse, Schlussfolgerungen oder Empfehlungen müssen nachvollziehbar sein und auf empirisch gewonnenen, qualitativen oder quantitativen Daten beruhen. Des Weiteren soll jede Evaluation vier grundlegende Eigenschaften aufweisen: Nützlichkeit, Durchführbarkeit, Fairness und Genauigkeit. Die *Internationale Organisation für Normung (ISO) [1] und die Internationale elektrotechnische Kommission (IEC) [19]* definieren die Evaluation in der Norm ISO/IEC 25000 als:

> *"systematic determination of the extent to which an entity meets its specified criteria".*

Außerdem werden die Personen, die an einer Evaluation teilnehmen, die Rollen Käufer (acquirer), Entwickler (developer), unabhängiger Evaluator (independent evaluators), Lieferant (supplier), Anwender (operator) und Maintainer zugewiesen. Der *Center for Civic Partnerships* [10] definieren die Evaluation als eine Methode, um Organisationen zu helfen, die Qualität ihrer Dienstleistungserbringung zu verbessern, sowie die gewünschten Ergebnisse zu erzielen. Zusätzlich wird die Evaluation abhängig von ihrem Schwerpunkt, in "formative" und "summative" aufgeteilt [9, 10]. Die formative Evaluation hat die Verbesserung eines Evaluationsgegenstandes zum Ziel und begleitet deswegen seine Entwicklung. Dagegen ist das Ziel der summativen eine zusammenfassende Bilanz zu ziehen. Diese Evaluation wird erst nach dem Abschluss eines Projektes durchgeführt.

Wie anhand der bereits vorgestellten Definitionen zu erkennen ist, wird der Begriff Evaluation in verschiedenen Kontexten verwendet und es ist zu erkennen, dass eine allgemeine

verbindliche Definition dieses Begriffes fehlt. Außerdem wird es ersichtlich, dass es sich bei der Evaluation um eine umfassende Bewertung schon erhobener Daten handelt. Trotzdem fällt es schwer eine Evaluationsmethode zu finden, die auf die Methodik und das Tool von Bachmann [39] angewendet werden kann.

Basierend auf bewährten Methoden und Ansätzen zur Evaluation werden wir in den nächsten Abschnitten die Grundlage unserer Evaluation vorstellen. Im Abschnitt 3.2 werden die einzelnen Methoden zur Durchführung einer Evaluation vorgestellt, wobei die im Thema dieser Arbeit ausgewählte Fallstudie im Mittelpunkt steht. Des Weiteren werden auch die Informationsquellen und die Kategorien von Daten, die zum Zweck einer Evaluation benötigt werden, erläutert. Da im Rahmen dieser Arbeit, die von Bachmann entwickelte Methodik und das Tool zu evaluieren sind, werden in den Abschnitten 3.3 und 3.4 Ansätze und Methoden zur Evaluation von Prozessen und Tools ausführlich vorgestellt.

3.2. Durchführung einer Evaluation

Vor dem Beginn einer Evaluation, müssen drei wichtige Fragen beantwortet werden.

1. Welche Methode wird bei der Evaluation verwendet? (siehe Abschnitt 3.2.1)
2. Welche Quellen sind für die Datenerfassung zu benutzen? (siehe Abschnitt 3.2.2)
3. Welcher Art von Daten werden während der Evaluation erhoben? (siehe Abschnitt 3.2.3)

3.2.1. Evaluationsmethoden

Angesichts der unterschiedlichen Ansätze zur Evaluation gibt es keine einheitliche Liste oder Kategorisierung von Methoden der Datenerhebung [15]. Obwohl unsere Evaluation einer Fallstudie zugrunde liegt, ist das Verständnis der Differenzen zwischen den einzelnen Methoden sehr wichtig und deswegen werden wir sie hier kurz vorstellen.
In ihrem Artikel *Case Studies for Method and Tool Evaluation* [8] kategorisieren Kitchenham und Pickard die Evaluationsmethoden in Fallstudie, Umfrage und formales Experiment. Sie beschreiben die Fallstudie als eine Studie, die sich nur auf ein einzelnes Projekt konzentriert und leicht zu planen ist. Des Weiteren helfen Fallstudien die Auswirkungen einer neuen Technologie in einer bestimmten Situation zu beschreiben. Im Gegensatz dazu lässt sich das formale Experiment leicht generalisieren. Es wird oft klein gehalten und enthält immer ein angemessenes Maß an Replikation. Aus diesem Grund kann das formale Experiment entweder auf ein einzelnes Projekt, das mehrmals repliziert wird, oder auf mehrere Projekte gleichzeitig angewendet werden. Wenn der Umfang der Studie über mehrere Projekte und Teams hinausgeht, bezeichnet Kitchenham und Pickard dies als Umfrage.
Anderseits unterscheidet E.Taylor-Powell in *Collecting Evaluation Data: An Overview of Sources and Methods* [15] zwischen mehreren Evaluationsmethoden wie z.B. Umfrage, Fallstudie, Interview, Beobachtung usw.. Hier wird die Fallstudie als eine ausführliche Untersuchung

eines bestimmten Falls beschrieben. Außerdem stützt sich die Fallstudie auf mehrere Informationsquellen und Methoden, um ein möglichst vollständiges Bild bereitzustellen. Umfragen sind Studien, die von Angesicht zu Angesicht, telefonisch oder elektronisch durchgeführt werden. Sie sammeln standardisierte Informationen über Fragebögen, um quantitative Daten (siehe Abschnitt 3.2.3) zu erheben. Interviews, ähnlich wie die Umfragen, sammeln Daten für die Evaluation durch die Durchführung von Gesprächen mit beteiligten Personen. Die Beobachtung teilt sich in strukturierte und unstrukturierte auf. Diese sammelt Informationen durch "sehen" und "zuhören".

Wie wir aus den bereits beschriebenen Evaluationsmethoden erkennen können, hängt die Entscheidung über eine geeignete Methode von verschiedenen Faktoren wie z.B. dem Umfang und der Art des Projektes ab [8]. Aus den beiden Definitionen des Begriffes der Fallstudie wird ersichtlich, dass diese am besten zu unserem Szenario passt.

3.2.2. Datenquellen der Evaluation

Ein weiterer wichtiger Punkt bei der Durchführung jeder Evaluation ist die richtige Auswahl von Informationsquellen. Informationen können auf unterschiedliche Weise gesammelt bzw. gewonnen werden. Da unsere Arbeit auf einer Fallstudie aus der Industrie basiert, werden die benötigten Informationen während der Evaluierung gesammelt.

3.2.3. Daten in der Evaluation

Unabhängig von der Auswahl der Evaluationsmethode, können die Informationen, die beim Evaluationsprozess generiert werden, in zwei Kategorien eingeteilt werden - qualitative und quantitative [35], [16]. Qualitative Daten werden in Worten ausgedrückt und tragen zu einem tieferen Verständnis von Zusammenhängen bei Prozessen und Tools bei [16]. Demgegenüber stehen die quantitativen Daten. Diese werden dazu genutzt, um bestimmte Aspekten eines Prozesses zu messen wie z.B. Anzahl der gefundenen Fehler [16] oder zum Testen von Hypothesen. Shull et al. beschreiben in ihrem Artikel die Nutzung der beiden Typen von Daten als sinnvoll, wenn es sich um die Evaluation von Prozessen handelt [16]. In seinem Buch [11] untersucht Silverman genauer die Unterschiede zwischen qualitativen und quantitativen Recherchen. Die quantitative Recherche beginnt mit einer Hypothese, die basierend auf einer großen Anzahl von Fällen geprüft wird. Die qualitative Recherche dagegen beginnt mit einer Fallstudie, die auch als Evaluationsmethode auch unserer Arbeit zugrunde liegt.
Für die Sammlung von qualitativen Daten stehen den Evaluatoren unterschiedliche Techniken zur Verfügung wie Fragenkataloge, Dokumentation und Interviews [15]. Bei unserer Evaluation werden wir sowohl die Dokumentation als Evaluationstechnik benutzen als auch die Zeit, die wir für die einzelnen Phasen der Evaluation gebraucht haben und die Performance von NovaERM, die wir bevor und nach der Migration, messen. Obwohl die Zeit und die Performance quantitativ sind, werden wir uns im Wesentlichen auf qualitative Daten konzentrieren.

Weiterhin ist im Rahmen dieser Arbeit ein Fragenkatalog (siehe Abschnitt 4.7) entstanden, der uns bei der Entscheidung bezüglich der Datenmigration in die Cloud unterstützen soll. Die Rolle des Fragenkatalogs sowie der Fragenkatalog, sind im Abschnitt 4.7 dargestellt und näher erläutert.

3.3. Evaluationsmethoden für Prozesse

Um ihren Produkte und Dienstleistungen besser, schneller und effizienter zu gestalten, greifen heutzutage viele Unternehmen auf die Migration in die Cloud zurück. In seiner Arbeit beschreibt Bachmann [39] mehrere Techniken zur Migration einer Datenbankschicht in die Cloud. Dafür entwickelte er eine Datenmigrationsmethodik, die auf der Methodik von Laszewski [40] basiert. Diese sowie die Methodik von Laszewski basieren auf dem Wasserfallmodel. Das Wasserfallmodel ist eines der bekanntesten Modelle in der Softwareentwicklung und wurde in den frühen 1970er Jahren von Royce [14] populär gemacht. In den vergangenen Jahren wurde das Model viel kritisiert. Bereits bei seiner Veröffentlichung hat Royce das Model als fehleranfällig bezeichnet. Sommerville übt in seinem Artikel *Software Process Models* [18] auch Kritik an dem Wasserfallmodel. Er kritisiert das Fehlen an Feedback zwischen den Phasen des Modells sowie die später Entdeckung von Fehlern, die in früheren Phasen der Entwicklung entstanden sind. Dass wir in unsere Arbeit, die von Bachmann entwickelte Methodik evaluieren, werden wir uns die Kritikpunkte zu Nutze machen und die Methodik in dieser Hinsicht überprüfen.

In den letzten Jahren sind viele Prozessmodelle entstanden, die danach streben Prozesse zu optimieren und zu verbessern. Als solche hat sich im Bereich der Systementwicklung das Capability Maturity Model Integration (CMMI) und im Bereich der IT-Servicemanagement ITIL zum de facto Standard entwickelt [13].

Das CMMI ist ein Qualitätsmanagementmodell, das an dem Software Engineering Institute (SEI) [5] entstanden ist. Das primäre Ziel von CMMI ist die Beurteilung und die Verbesserung der Qualität von Produktentwicklungsprozessen in Unternehmen. Das Model wird dazu verwendet, um die Stärken und Schwächen einer Produktentwicklung zu analysieren, bei der Abschätzung von Kosten und Aufwänden und für die Effiziente Einsetzung von Ressourcen [17], [42]. Sekundär setzt sich CMMI das Ziel die Überprüfung des Reifegrades eines Unternehmens. Darauffolgend beinhaltet das Model zwei unterschiedlichen Repräsentationen - stufenweise und kontinuierliche, welche zwei verschiedene Vorgehensweisen im CMMI beschreiben. Zum Einen bietet die stufenweise Repräsentation eines Unternehmens einen strukturierten Verbesserungspfad, mit dem das Unternehmen ihre Fähigkeiten, Systeme zu entwickeln und zu integrieren, steigern kann. Dem gegenüber steht die kontinuierliche Repräsentation, die den Reifengrad eines Unternehmen bewertet, komplexe technische Systeme nach weltweit standartisierten und langjährig erprobten Methoden zu entwickeln [13]. Das CMMI ist ein sehr umfangreiches und aufwändiges Qualitätsmanagementmodell. Des Weiteren ist das CMMI Modell an der Verbesserung des Entwicklungsprozesses gerichtet und deswegen zum Zwecke unserer Evaluation nicht geeignet.

ITIL ist ein "Best Practices" - Framework, das sich als Standard-Prozessrahmen für IT Service Management (ITSM) etabliert hat [21]. Ziele von ITIL sind die Erhöhung der Kundenzufriedenheit und der Produktivität, der gezielte Einsatz von Know-How sowie die Verbesserung der Kommunikation zwischen IT-Mitarbeiter und Kunden bzw. Kollegen [32]. ITIL definiert verschiedene Prozessen, die als Leitfaden für die effektive Durchführung von ITSM dienen. In der letzte Version ITIL V3 2011 Edition sind diese Prozesse in die sogenannte Publikationen zusammengefasst:

- ITIL Service Strategy (ITIL Servicestrategie)
- ITIL Service Design
- ITIL Service Transition (ITIL Serviceüberführung)
- ITIL Service Operation (Servicebetrieb)
- ITIL Continual Service Improvement (CSI) (Kontinuierliche Serviceverbesserung)

Alle Publikationen zusammen bilden den Service Lebenszyklus ab und stellen den Kern des ITIL Modells dar. Der Lebenszyklus ist in der Abb. 3.1 dargestellt.

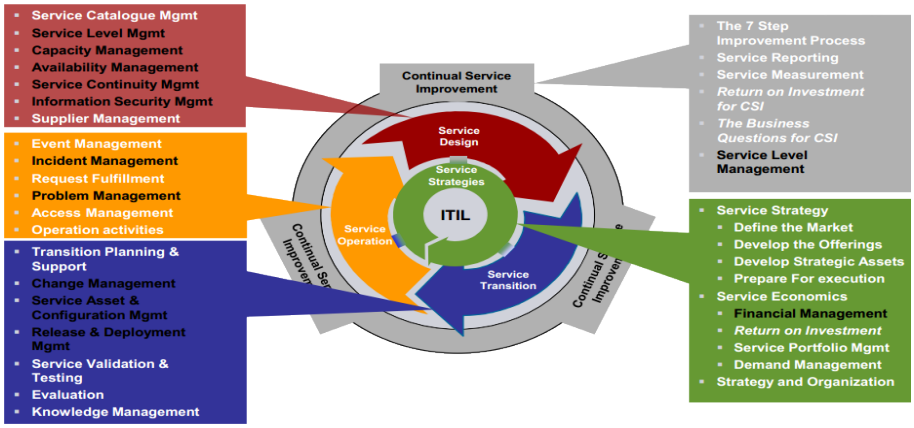

Abbildung 3.1.: ITIL Lebenszyklus [24]

Die ITIL Servicestrategie steht im Zentrum des Service Lebenszyklus und liefert die Anleitung zum Entwerfen, Entwickeln und Umsetzen des Service Management als strategisches Asset [41]. Das ITIL Service Design beschäftigt sich mit dem Entwurf und der Entwicklung von Services und ihren zugehörigen Prozessen. Das Ziel des Service Designs ist das Entwickeln von neuer oder die Anpassung von alten Services an die Produktivumgebung [41]. Die Service Transition koordiniert und managt die Prozesse und unterstützt die Überführung von Services im laufenden Betrieb. Alle definierten und umgesetzten Anforderungen in den ersten zwei Phasen werden durch die Service Transaktion effektiv umgesetzt [41], [33]. Die Service Operation Phase stellt die Effektivität und die Effizient bei der Lieferung und Unterstützung des IT-Services sicher und gewährleistet den erwarteten Nutzen [33]. Während sich die ersten vier Phasen des Lebenszyklus eher an die Entwicklung von Services und Prozessen richten, beschäftigt sich die ITIL CSI mit deren Verbesserung. CSI verbindet Prinzipien, Praktiken und Methoden aus dem Qualitätsmanagement und dem Änderungsmanagement

und strebt die kontinuierliche Verbesserung sowohl der einzelnen Phasen des Lebenszyklus als auch von bereits existierenden Prozessen und Services an. Der CSI Prozess beschreibt eine Methodik, die zur Identifizierung von Mängeln bei den Services und Prozessen und zur Implementierung von Verbesserungen verwendet werden kann. Diese Methodik besteht aus sieben Schritten und ist in Abb. 3.2 dargestellt.

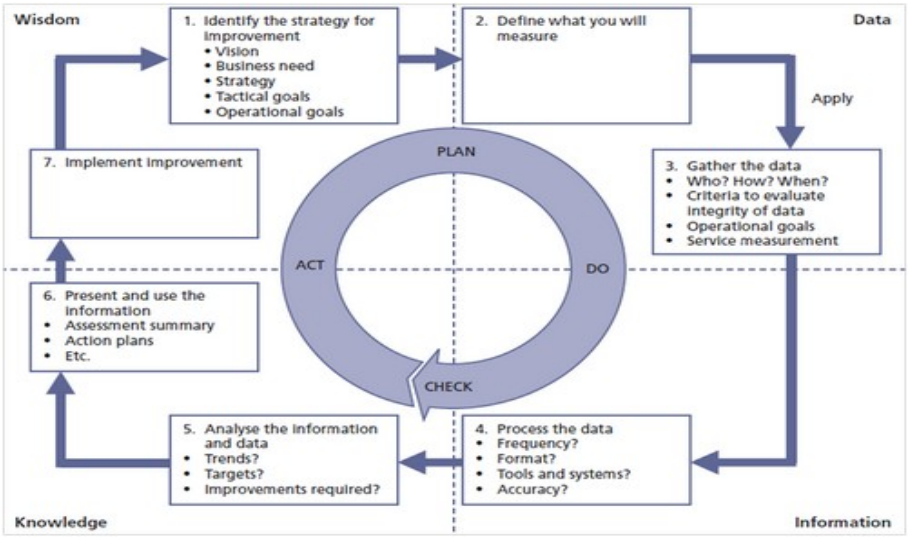

Abbildung 3.2.: ITIL Verbesserungsprozess [25]

Da die Evaluation der Migrationsmethodik (siehe Abschnitt 2.7), die in ihren einzelnen Schritten als ein Prozess angesehen werden kann, eines der Ziele unserer Arbeit ist, können wir den ITIL CSI Prozess zu diesem Zweck anpassen und wiederverwenden. Im ersten Schritt dieses Prozesses muss die Verbesserungsstrategie identifiziert werden d.h. es müssen die Gründe für die Durchführung des Prozesses festgelegt werden. Basierend auf dem Thema unserer Arbeit steht die Evaluation der Methodik und dem Tool von Bachmann als Grundlage für unsere Strategie. Der zweite Schritt legt fest was gemessen werden kann. Dieser Schritt stellt das Ziel unserer Evaluation dar und beantwortet die Frage: Welches Ziel wird angestrebt und wie erreichen wir es? Der dritte Schritt bestimmt die Art, wie die benötigten Informationen für unsere Messung gesammelt werden. Für die Sammlung von Informationen während unserer Evaluation werden wir uns zwei Tools zu Nutze machen - *inspectIT* [28] beschrieben im Abschnitt 2.5 und *seleniumHQ* [4] beschrieben im Abschnitt 2.6. Die Beschreibung von Fehlern während der Evaluation erfolgt manuell. Im vierten Schritt werden wir uns auf die Verarbeitung der erfassten Daten konzentrieren. Am Ende unserer Migration werden wir sie analysieren - Schritt fünf. Im vorletzten Schritt sechs, werden unseren Schlussfolgerungen zusammengefasst und die vorgeschlagenen Verbesserungen werden dann im letzten Schritt auf die Methodik und das Tool von Bachmann angewendet.

3.3.1. Softwareprozessmetriken

Um Prozesse oder Tools präzise beschreiben oder bewerten zu können, brauchen wir Metriken. Sie unterstützen sowohl Evaluatoren bei der Qualitätsbewertung von Produkten und

Prozessen als auch Entwickler bei ihren Entscheidungen und um Kosten und Termine besser zu prognostizieren [23]. In seinem Buch "Metrics and Models in Software Quality Engineering" [36] klassifiziert Stephen Kan die Softwaremetriken in drei Gruppen - Produktmetriken, Prozessmetriken und Projektmetriken. Da unsere Evaluation ein bereits entwickeltes Tool sowie eine existierende Methodik betrachtet, werden die Projektmetriken in dieser Arbeit nicht betrachtet.
Mit dem Ziel den Softwareentwicklungs- und Softwarewartungsprozess zu verbessern, definiert Kan vier Prozessmetriken:

1. Defect Density During Machine Testing
2. Defect Arrival Pattern During Machine Testing
3. Phase-Based Defect Removal Pattern
4. Defect Removal Effectiveness

Wie bereits beschreiben, richten sich diesen Metriken an den Softwareentwicklungs- und Softwarewartungsprozess. Wie das Thema unserer Arbeit definiert, wird eine bereits entwickelte Methodik, die als Prozess betrachtet werden kann, evaluiert. Da wir während unserer Literaturrecherche keine passenden Metriken gefunden haben, werden wir diese anpassen und wiederverwenden.

Defect Density During Machine Testing zeigt den gefundenen Fehler pro Einheit - 1000 Lines of Code (KLOC), Funtion Point oder Phase in einer Version eines Softwareproduktes. Diese Metrik ist ein guter Indikator während sich die Software noch in der Testphase befindet und ist besonders nützlich für die Überwachung der nachfolgenden Versionen eines Produkts in der gleichen Entwicklung Organisation. Damit ist der Vergleich zwischen den einzelnen Versionen nicht durch externe Faktoren beeinflusst. Um diese Metrik anwenden zu können, muss sie angepasst werden. Alle Fehler, die während der Evaluation der Methodik in einer Phase auftreten, werden erfasst. Dadurch wird die Fehlerdichte pro Phase der Methodik erfasst.

Defect Arrival Pattern During Machine Testing Metrik beschäftigt sich mit der Häufigkeit des Auftretens unterschiedlicher Typen von Fehlern sowie die Zeit zwischen den einzelnen Fehlern während der Testphase. Die Datenerhebung dauert in der Regel mehrere Wochen bis Monate. Da wir für unsere Fallstudie nur über eine bestimmten Zeit verfügen, werden wir uns auf das Auftreten der unterschiedlichen Typen von Fehlern in den einzelnen Phasen der Methodik konzentrieren und diese beschreiben.

Phase-Based Defect Removal Pattern ist eine Erweiterung der *Defect Density During Machine Testing* Metrik. Zusätzlich zu den Tests, empfiehlt die Metrik die Verfolgung von Fehlern in allen Phasen des Entwicklungsprozesses. Da die meisten Fehler in frühen Phasen der Entwicklung entstehen, haben diese später eine große Auswirkung, insbesondere in der Testphase. Deswegen empfiehlt die Metrik die regelmäßige Durchführung von Reviews. Da wir im Rahmen unserer Arbeit die Evaluation einer bereits entwickelten Methodik und Tools durchführen, ist die Nutzung dieser Metrik nicht möglich.

Defect Removal Effectiveness (DRE) Metrik misst den Prozentsatz von behobenen Fehlern während allen Entwicklungsphasen (von der Anforderungsphase bis hin zur Testphase) im Vergleich zu allen entdeckten Fehlern im Produktivbetrieb.

$$DRE = \frac{\text{Anzahl Fehler während der Entwicklungsphasen}}{\text{Anzahl entdeckter Fehler im Produktivbetrieb}} * 100\% \tag{3.1}$$

Um diese Metrik benutzen zu können, müssen wir sie an unseren Anforderungen anpassen. Wir werden den Prozentsatz der von uns behobenen Fehlern am Ende unserer Arbeit berechnen. Dafür wird die Formel wie folgt angepasst:

$$DRE = \frac{\text{Anzahl behobener Fehler}}{\text{Anzahl entdeckter Fehler während der Migration}} * 100\% \tag{3.2}$$

3.4. Evaluationsmethoden für Tools

Wenn es um die Evaluation der Qualität eines Softwareproduktes geht, wird häufig auf den ISO 9126 Standard zurückgegriffen. Diese ISO-Norm stammt aus dem Jahr 1991 und wurde im Jahr 2005 durch die *Software product Quality Requirements and Evaluation (SQuaRE)*-Normen der ISO 25000-Reihe abgelöst. In der neuen ISO Norm 25010 wird der Begriff der Qualität wie folgt definiert:

"The degree to which the system satisfies the stated and implied needs of its various stakeholders, and thus provides value."

Anhand dieser Definition wird die Unterscheidung zwischen Anforderungsorientierter Qualitätssicht und der Nutzensicht klar. Diese Trennung erfolgt durch die Definition zweier Qualitätsmodelle - Produktqualitätsmodell und Benutzungsqualitätsmodell. Jedes der beiden Modelle definiert mehrere Qualitätscharakteristika, die die Qualität des Softwareproduktes spezifizieren. Jedes Qualitätscharakterisitikum ist wiederum in mehrere Attribute aufgeteilt.

Das Benutzungsqualitätsmodel in der Abb. 3.3 besteht aus fünf Qualitätscharakteristika - Effektivität, Effizienz, Zufriedenheit, Freiheit von Risiko und Kontext Abdeckung, die sich auf das Ergebnis der Interaktion beziehen, wenn das Softwareprodukt in einem bestimmten Kontext genutzt wird.

Das Produktqualitätsmodel in der Abb. 3.4 unterscheidet zwischen acht Qualitätscharakteristika - Gebrauchstauglichkeit, Leistungseffizienz, Kompatibilität, Benutzerfreundlichkeit, Zuverlässigkeit, Sicherheit, Wartbarkeit, Portabilität und betrachtet das Softwareprodukt ohne jeglichen Kontext.

Der Zusammenhang zwischen den beiden Modellen wird anhand der Abb. 3.5 ersichtlich. Die Qualität des Softwareprodukts kann durch die Messung von internen Eigenschaften während der Entwicklungsphase, die Messung von externen Eigenschaften während der Testphase oder die Messung von der Benutzungsqualität während das Softwareprodukt im Einsatz

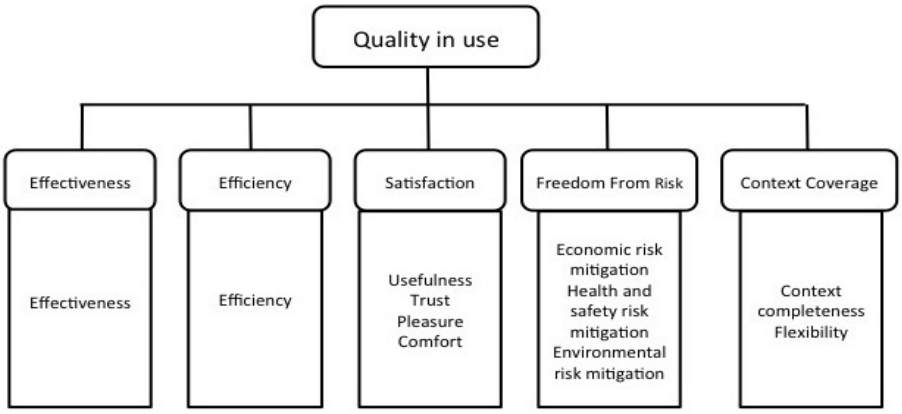

Abbildung 3.3.: Benutzungsqualitätsmodel [20]

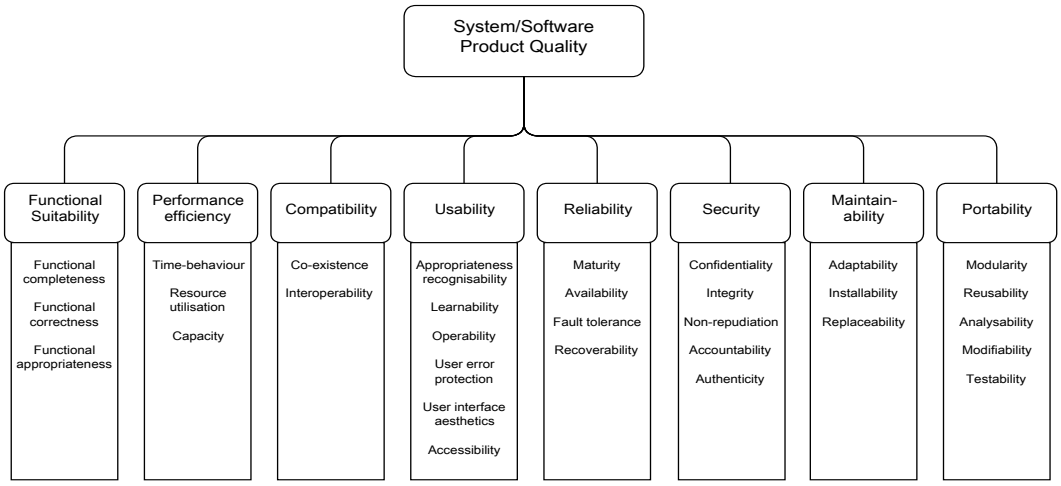

Abbildung 3.4.: Produktqualitätsmodel [20]

ist, evaluiert werden. Daran erkennen wir, dass die Prozessqualität (links in der Abbildung dargestellt) eine indirekte Auswirkung auf die Benutzungsqualität hat (ganz rechts in der Abbildung) und zu deren Verbesserung beiträgt.

Aufgrund der hohen Anzahl von Qualitätscharakteristika und des eingeschränkten Zeitrahmens werden wir uns während unserer Evaluation bei der Benutzungsqualität auf die Effektivität und Effizienz fokussieren und bei der Zufriedenheit mit Fokus auf Brauchbarkeit/Nützlichkeit. Des Weiteren werden wir uns bei der Evaluation der Produktqualität auf die Gebrauchstauglichkeit mit Schwerpunkt auf die Funktionale Vollständigkeit, Korrektheit und Angemessenheit sowie bei der Benutzerfreundlichkeit mit Schwerpunkt auf die Angemessenheit und die Lernbarkeit fokussieren.

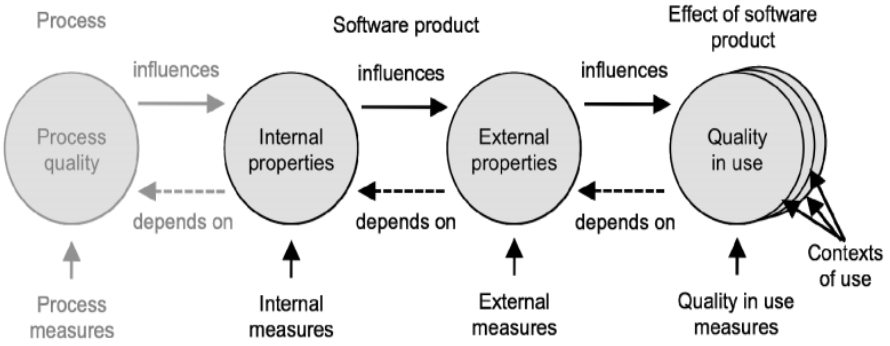

Abbildung 3.5.: Produktqualität Lebenszyklus [20]

3.4.1. Softwareproduktmetriken

Bei der Evaluation des Tools von Bachmann werden wir die Qualitätsmetriken von Kan [36] für Softwareprodukte benutzen. In seinem Buch teilt Kan die Metriken für die Softwareproduktqualität in zwei Ebenen auf: Eigentliche Produktqualität und Kundenzufriedenheit. In jede Ebene definiert er wiederum zwei Metriken:

1. Eigentliche Produktqualität
    - Mean time to failure (MTTF)
    - Fehlerdichte
2. Kundenzufriedenheit
    - Kundenprobleme
    - Kundenzufriedenheit

Die eigentliche Produktqualität wird gemessen anhand der Anzahl von Fehlern relativ zur Größe der Software Lines of Code (LOC) (Fehlerdichte) oder der Zeit zwischen dem Auftritt von zwei Fehlern (MTTF). Die MTTF Metrik wird vorwiegend bei sicherheitskritischen Systemen wie den Steuersystemen des Flugverkehrs verwendet. Dagegen wird die Metrik Fehlerdichte eher bei kommerziellen Software-Systemen verwendet. Obwohl das MTTF das Ergebnis des Auftretens eines Fehlers ist, besitzen die zwei Metriken viele Gemeinsamkeiten. Außerdem ist die Fehlerrate eines Produkts bzw. die Anzahl der zu erwartenden Defekte über einen bestimmten Zeitraum aus Kosten- und Ressourcengründe für die Wartungsphase eines Softwareprodukts sehr wichtig. Des Weiteren ist die Erfassung der Zeit zwischen zwei Fehlern sehr aufwändig und erfordert eine hohe Präzision. Um diese Metrik benutzen zu können, müssen wir sie anpassen. Deswegen werden wir die Anzahl aller auftretender Fehler der Zeit, die wir für die Evaluation des Tools in jeder Phase benötigt haben, entgegensetzen.

Wie bereits erwähnt, misst die *Fehlerdichte* die Anzahl der gefundenen Fehler relativ zur Größe der Software in KLOC. Da unserer Evaluation auf einem bereits entwickelten Tool basiert,

werden wir nur wenig Software entwickeln und deswegen diese Metrik nicht benutzen können.

Die *Kundenprobleme* Metrik wird von vielen Entwicklern benutzt und misst die Probleme, mit denen die Kunden konfrontiert werden, während sie mit dem Produkt arbeiten. Aus Kundensicht stellen alle Probleme, die während der Benutzung des Produktes auftreten, einen Fehler dar. Diese Fehler können an mangelhafter Dokumentation oder aber auch an der Benutzbarkeit des Produkts liegen. Diese Metrik wird in der Einheit Probleme pro Nutzer pro Monat (PUM) angegeben.:

$$PUM = \frac{\text{Anzahl der gemeldeten Fehler pro Periode}}{\text{Anzahl von Lizense-Monaten der Software pro Periode}} \tag{3.3}$$

Da wir im Rahmen unserer Arbeit eine Fallstudie durchführen und ein Tool und eine Methodik evaluieren, können wir diese Metrik nicht benutzen.

Kundenzufriedenheit Metrik wird oft mit Hilfe einer Umfragen gemessen und beinhaltet die folgenden fünf Punkten:

1. Sehr zufrieden
2. Zufrieden
3. Neutral
4. Unzufrieden
5. Sehr unzufrieden

Die Zufriedenheit des Kunden wird meistens in mehrere Kategorien unterteilt wie z.B. Funktionalität, Brauchbarkeit, Zuverlässigkeit, Leistung usw.. Da als Evaluationsmethode unserer Arbeit eine Fallstudie zugrunde liegt und keine Umfrage, können wir diese Metrik nicht benutzen. Als Alternative werden wir im letzten Kapitel [7] eine zusammenfassende Beschreibung geben.

4. Spezifikation und Entwurf der Evaluation

Basierend auf dem vorherigen Kapitel werden wir in diesem die Vorgehensweise bei der Durchführung der Evaluation erläutern und auf die einzelnen Schritte des ITIL CSI Prozesses abbilden. Wie bereits im Abschnitt 3.3 beschrieben, verbindet der ITIL CSI Prozess Prinzipien, Praktiken und Methoden aus dem Qualitätsmanagement und dem Änderungsmanagement und strebt die kontinuierliche Verbesserung von Prozessen und Services an. Da es sich bei der Migrationsmethodik um einen Prozess handelt, werden wir den ITIL CSI Prozess für unsere Evaluation anpassen und wiederverwenden. Die Abbildung der einzelnen Schritten des Prozesses auf die Kapitel dieser Arbeit zeigt die Tabelle 4.1.

ITIL CSI Schritt	Kapitel	Abschnitt
1. Verbesserungsstrategie identifizieren	4	4.2
2. Ziele der Evaluation	4	4.3, 4.4, 4.5, 4.6
3. Evaluation Datenerhebung	6	6.1, 6.2.1, 6.2.2, 6.3.1, 6.3.2, 6.3.4
4. Evaluation Datenverarbeitung	6	6.2.3, 6.3.3, 6.3.5
5. Evaluationsanalyse	6	7.1.1, 7.1.2
6. Evaluationsergebnisse	7	7.1.1, 7.1.2
7. Verbesserungsvorschläge	7	7.1.1

Tabelle 4.1.: Abbildung des ITIL CSI Verbesserungsprozesses auf die Evaluationsschritte

4.1. Evaluationsvorbereitung

Bevor wir mit der Evaluation der Methodik und des Tools von Bachmann [39] anfangen, muss die Arbeitsumgebung vorbereitet werden.

Es werden insgesamt zwei Schritte durchgeführt. Diese werden uns ermöglichen NovaERM aufzusetzen, um die Performance der lokal gehosteten Anwendung (on-premise) zu messen. Die nächsten zwei Abschnitte beschreiben die einzelnen Schritte in Bezug auf die Vorbereitung der Evaluation.

4.1.1. Schritt 1: Aufsetzten von NovaERM

Im ersten Schritt der Vorbereitung wird mittels *automaIT* (siehe Abschnitt 2.4), die für die Evaluation benötigte Infrastruktur aufgesetzt. Diese beinhaltet sowohl die Webapplikation

und die dazugehörige Datenbank als auch einen Applikation Server. Wie die einzelnen Komponenten zusammenhängen und mit automaIT installiert werden, wird im Abschnitt 5.1.1 näher erläutert.

4.1.2. Schritt 2: Performance Diagnose von NovaERM

Im zweiten Schritt werden wir eine Performance Diagnose von NovaERM durchführen. Als Application Performance Management (APM) Werkzeug werden wir das Tool inspectIT (siehe Abschnitt 2.5) benutzen. Das Tool ermöglicht eine Performance Diagnose parallel zur Laufzeit durchzuführen, während wir mittels seleniumHQ (siehe Abschnnitt 2.6) einen vollständigen Graphical User Interface (GUI) Test durchführen. Diese Diagnose wird quantitative Daten zur Performance von NovaERM liefern, während die Anwendung lokal gehostet wird. Diese Daten werden einen Performance Vergleich nach einer erfolgreichen Migration ermöglichen.

4.2. Verbesserungsstrategie identifizieren

Im ersten Schritt des ITIL CSI Prozesses sind die Gründe für dessen Durchführung festzulegen. In unserem Fall wird die Strategie durch die Evaluation der Migrationsmethodik und des Tools von Bachmann definiert.

Im Rahmen der Evaluation wird die Applikation NovaERM in die Cloud migriert, die wie im Abschnitt 2.3 beschrieben, aus insgesamt drei Komponenten besteht. Deshalb findet der Migrationsprozess in zwei Phasen statt. Während der ersten Phase werden mit Hilfe von Bachmanns Methodik und Tools die zwei Datenbanken von NovaERM in zwei aufeinanderfolgenden Schritten migriert. Nach einer erfolgreichen Migration, wird im zweiten Schritt die dazugehörige Webapplikation migriert. Eine detaillierte Beschreibung der einzelnen Migrationsszenarien ist im Abschnitt 4.5 zu finden.

4.3. Ziele der Evaluation

Nachdem im vorherigen Abschnitt die Verbesserungsstrategie definiert wurde, werden nun im zweiten Schritt des ITIL CSI Prozesses die Ziele dieser Arbeit erläutert. Als Grundlage dafür dient die Formulierung der Diplomarbeit.

Basierend auf der bereits entwickelten Methodik und dem Tool von Bachmann [39] wird sich diese Arbeit auf deren Evaluierung und Verbesserung konzentrieren. Einer der wichtigen Aspekte der Evaluation ist die Eignung des Tool zur Migration eines Datenbanksystems sowie die Unterstützung der Entscheidungsfindung zur Auswahl eines Cloud Anbieters. Zum anderen soll festgelegt werden in wieweit die Methodologie und das Tool sich zur Anpassung der Architektur einer Anwendung eignen. Diese Punkte werden anhand der in den Abschnitten 3.3 und 3.4 beschriebenen Kriterien evaluiert. Bereits im Abschnitt 3.3 wurde

Kritik an dem Migrationsmodel geübt im Hinsicht auf das Fehlen an Feedback zwischen den Phasen sowie der spätere Entdeckung von Fehlern, die in früheren Phasen entstanden sind. In [16] beschreiben Shull et al. drei wichtige Gründe zur Nutzung von Iteration während einer Evaluierung [16]:

1. besseres Verständnis und Nachvollziehbarkeit der Faktoren, die Einfluss nehmen auf den Prozess,
2. sicherstellen, dass grundlegende Fragen angesprochen werden, bevor ein Prozess an einer Umgebung abgestimmt wird,
3. effizientere Nutzung von Ressourcen, um einen großen Profit zu erzielen.

Basierend auf den Eintritt- bzw. Austrittsbedingungen jeder Phase die Bachmann in seiner Methodik [39] definiert hat, werden wir am Ende unserer Arbeit, entsprechend Schritt 7 des ITIL CSI Prozess Verbesserungsvorschläge und ein Model zur iterative Durchführung der Methodik entwickeln.

Weitere Aspekte, die in Betracht gezogen werden, sind neben den Metriken, wie in den Abschnitten 3.3.1 und 3.4.1 beschrieben, auch die Qualitätscharakteristika für die Benutzungsqualität und für die Produktqualität, beschrieben im Abschnitt 3.4. Abschließend werden alle Fehler während der Evaluation aufgezeichnet und dokumentiert.

Es ist auch wichtig zu erwähnen, dass Bachmann in seiner Arbeit mehreren Szenarien für die Migration der Datenschicht vorstellt und wir nicht alle in Betracht ziehen werden.

Ein weiterer wichtiger Aspekt unserer Arbeit ist die Untersuchung von relevanten Fragen bei der Migration einer Anwendung. Im Rahmen dieser Arbeit ist ein Fragenkatalog entstanden, der im Abschnitt 4.7 vorgestellt wird. Mit Hilfe dieses Fragenkataloges werden wichtige Fragen beantwortet, die zu einer erfolgreichen Migration einer Anwendung beitragen.

4.4. Sammlung von Daten

Die Sammlung der richtigen Daten unterstützt das Ziehen der richtigen Schlussfolgerung und dementsprechend auch die Empfehlung der korrekten Verbesserungen sowohl in Bezug auf der Migrationsmethodik als auch des Tools. Wie wir im Abschnitt 3.2.3 beschrieben haben, können die Daten in der Evaluation unabhängig von der Evaluationsmethode in qualitativ und quantitativ kategorisiert werden. Obwohl wir zum großen Teil unserer Arbeit uns die qualitativen Daten zu Nutzte machen, werden wir uns für die Problem-, und Fehlerbeschreibungen sowie für manche Metriken (Kundenprobleme Metrik 3.4.1, Defect Density During Machine Testing, Phase-Based Defect Removal Pattern 3.3.1) auf die quantitativen Daten stützen. Die Ergebnisse aus der Messung mit *inspectIT* [28] und *seleniumHQ* [4] werden auch in quantitativen Daten erfasst. Um ein besseres Verständnis bezüglich des Evaluationsprozesses zu bekommen, werden alle auftretenden Fehler dokumentiert und für deren Beschreibung einer Vorlage in der Tabelle 4.2 vorgestellt. Die Vorlage für die Fehlerbeschreibung trägt zum besseres Verständnis des Evaluationsprozesses sowie zu einer leichteren Datenverarbeitung und Analyse bei. Jede Fehlerbeschreibung wird mit einer laufenden ID (1, 2 usw.) versehen.

Value	Key
ID	MD 3
Name	Datenbankmigrationsfehler
Fehlerzugehörigkeit	Methodik
Schweregrad	kritisch
Priorität	hoch
Reproduzierbarkeit	Ja, Beschreibung der dazugehörigen Schritte...
Beschreibung	Die Migration der Datenbank benötigt einen zusätzlichen Benutzer mit ausreichenden Rechten
Fehlerbehandlung	Wir haben einen Administrator Benutzer für die Datenbank eingerichtet
Fehlerlösung	Bevor die Migration der Datenbank beginnt, muss einen Benutzer mit Administratorrechten eingerichtet werden.
Kommentar	Kommentar

Tabelle 4.2.: ITIL CSI Verbesserungsprozess

Eine ID hilft bei der Nachvollziehbarkeit in welchem Schritt der Methodik dieser Fehler aufgetreten ist. Der Wert einer ID besteht aus einem Bezeichner für ein konkretes Migrationsszenario. Die Bezeichner sind somit an die Migrationsszenarien gebunden (siehe Abschnitt 4.5) und definieren das Szenario in dem der Fehler aufgetreten ist. Die möglichen Bezeichner sind:

- **MD**: Migration Datenbank
- **MW**: Migration Webanwendung
- **MG**: Migration Gesamtanwendung

Zusätzlich wird jeder Fehler durch einen Namen benannt. Die Fehlerzugehörigkeit beschreibt den Ursprung eines Fehlers. Ein Fehler kann mehrere unterschiedliche Fehlerzugehörigkeiten besitzen - Methodik, Tool oder Anderer. Des Weiteren wird bei dem Tool Fehlerzugehörigkeit noch das entsprechende Qualitätscharakteristika (siehe Abschnitt 3.4) hinzugefügt. Bei dem Schweregrad eines Fehlers kann zwischen *leicht, mittel, hoch und kritisch* unterschieden werden. Der Schwergrad beschreibt den Grad der Fehlerauswirkung. Dagegen wird durch die Priorität eine Empfehlung für die Notwendigkeit der Korrektur eines Fehlers im Tool gegeben. Bei der Methodik wird dieser Punkt ignoriert. Hier kann wiederum zwischen *niedrig, mittel und hoch* unterschieden werden. Die Reproduzierbarkeit gibt an, ob ein Fehler reproduziert werden kann oder nicht. Falls der Fehler reproduzierbar ist, wird eine genauere Beschreibung gegeben. Zu jedem Fehler gehört ebenfalls eine detaillierte Beschreibung sowie ein Lösungsansatz der zur Behebung des Fehlers führen kann. Die Beschreibung der Lösung wird in der Zeile "Fehlerlösung" gegeben. An letzter Stelle in der Fehlervorlage ist es noch möglich einen Kommentar anzugeben.

4.5. Migrationsplan und Evaluationsstruktur

Nachdem die Evaluation vorbereitet ist und die Ziele definiert sind, wird in diesem Abschnitt der Migrationsplan beschrieben und erläutert.

Wie bereits im Abschnitt 2.3 vorgestellt, baut NovaERM auf drei Komponenten auf - Aktiviti Datenbank, NovaERM Datenbank und NovaERM Webapplikation. Als Grundlagen für die Evaluation wird die Migration der drei Komponenten in drei unterschiedliche Migrationsszenarien aufgeteilt. Des Weiteren wird in jedem Szenario, abhängig von der Komponente (Datenbank oder Webapplikation) die migriert wird, eine andere Methodik (Bachmann [39] oder Amazon Web Services [22]) verwendet. Die Migrationsszenarien und die dazu verwendeten Methodiken werden in den folgenden Abschnitten detailliert dargestellt.

4.5.1. Migration Datenbank (MD)

Wie der Name des Abschnittes beschreibt, werden in diesem Migrationsszenario jeweils die beiden Datenbanken in die Cloud migriert, wobei die Webapplikation lokal gehostet wird. Der Plan dieses Szenarios ist in Abbildung 4.1 dargestellt:

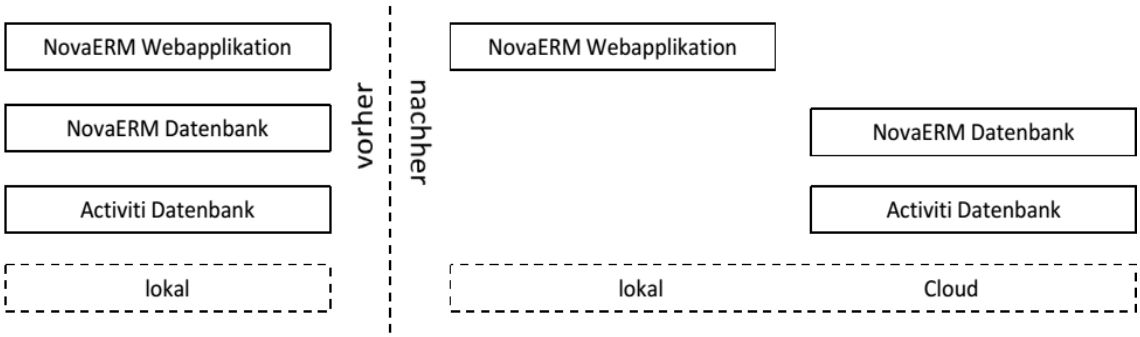

Abbildung 4.1.: Migrationsszenario Datenbank

Die Migration der Datenbanken erfolgt mittels der Methodik und des Tools von Bachmann. Zusätzlich zu jeder Phase der Methodik werden wir die Probleme sowie deren Lösungsansätze kurz zusammenfassen.

4.5.2. Migration Webanwendung (MW)

Bei diesem Migrationsszenario wird nur die NovaERM Webapplikation in die Cloud migriert. Die beiden Datenbanken - Activiti Datenbank und NovaERM Datenbank, bleiben lokal gehostet. Der Plan des Migrationsszenarios ist in Abbildung 4.2 zu sehen:

Da innerhalb dieses Szenarios nur die Webapplikation zu migrieren ist, werden wir uns nur auf die phasengetriebene Cloud Migrationsmethodik von AWS (siehe Abschnitt 2.10)

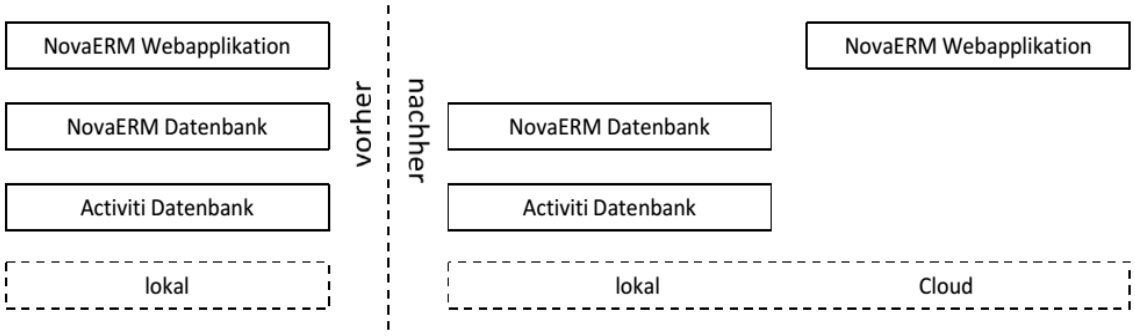

Abbildung 4.2.: Migrationsszenario Webapplikation

stützen. Des Weiteren entfällt der Datenmigrationsschritt aus der Migrationsmethodik von Amazon.

4.5.3. Gesamtmigration (MG)

Bei dem Gesamtmigrationsszenario handelt es sich um eine Migration der gesamten Anwendung. Diese Migration beinhaltet sowohl die Webapplikation als auch die zwei Datenbanken und ist in Abbildung 4.3 zu sehen.

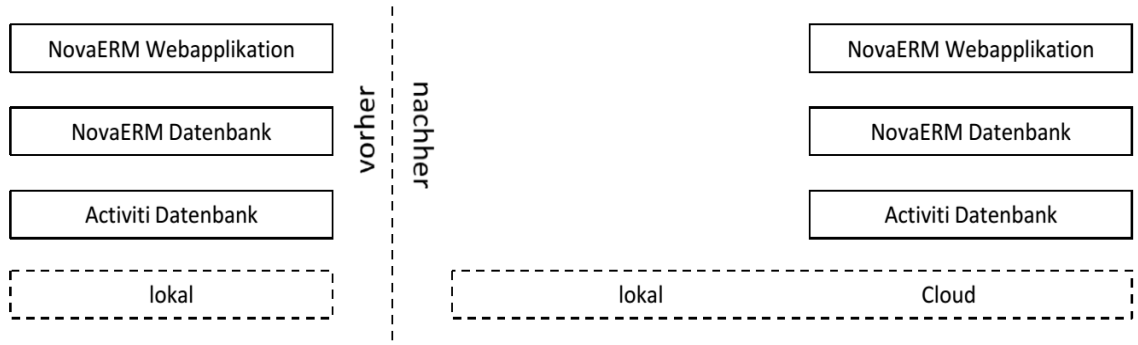

Abbildung 4.3.: Migrationsszenario der gesamten Anwendung

Da im Laufe dieses Szenarios sowohl die Webapplikation als auch die beiden Datenbanken migriert werden, brauchen wir einen gesamtheitlichen Ansatz, um die Migration durchzuführen. Wie bereits Bachmann [39] in seiner Arbeit beschrieben hat, kann seine Cloud Datenmigrationsmethodik innerhalb der Cloud Anwendungsmigrationsmethodik von AWS verwendet werden. Diesen Ansatz werden wir in unserer Arbeit weiter verwenden und eine Kombination aus beiden Methodiken zu einer Gesamtmigration kombinieren, wie in Abbildung 4.4 dargestellt.

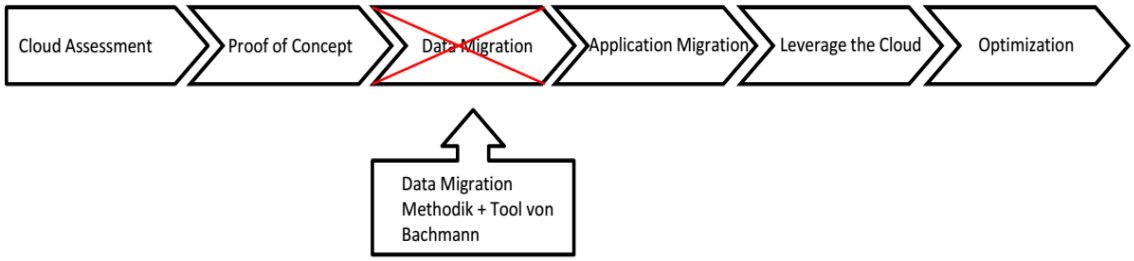

Abbildung 4.4.: Kombinationsmigrationsszenario AWS und Bachmann

4.5.4. Auswahl von Migrationsszenarien

Da das Thema unserer Arbeit durch die Evaluation der Migrationsmethodik von Bachmann [39] definiert ist und sich auf die Migration der Datenbank in die Cloud fokussiert, werden wir nur die Migrationsszenarien **MD** 4.5.1 und **MG** 4.5.3 durchführen. Im Szenario **MW** 4.5.2 wird nur eine Migration der Webanwendung durchgeführt und ist deshalb nicht als Teil dieser Arbeit zu betrachten.

4.6. Evaluationsstruktur

In diesem Abschnitt unserer Arbeit wird die Struktur für die Durchführung der Evaluation detailliert beschrieben und erläutert.

Wie bereits im Evaluationsplan (siehe Abschnitt 1.3) beschrieben, findet unsere Evaluation in zwei Iterationen statt. Während sich die erste Iteration nur auf die Migration der zwei Datenbanken von NovaERM beschränkt (siehe Abschnitt 4.5.1), werden in der zweiten Iteration die in den Abschnitten 4.5.1 und 4.5.3 beschriebenen Migrationsszenarien durchgeführt. Abschließend wird zu jedem Migrationsszenario eine Performance Messung durchgeführt. Außerdem werden alle auftretenden Fehler mit Hilfe der im Abschnitt 4.4 definierten Vorlage erfasst. Eine detaillierte Beschreibung der zwei Iterationen wird in den folgenden Abschnitten gegeben.

4.6.1. Iteration 1 - NovaTec Cloud

In der ersten Iteration werden wir uns auf die Migrationsmethodik von Bachmann [39] stützen und in zwei aufeinander folgenden Schritten, die NovaERM DB und die Activiti DB in die NovaTec Cloud migrieren. Dieses Vorgehen entspricht dem Migrationsszenario **MD** (siehe Abschnitt 4.5.1). Da sich NovaERM immer noch in Entwicklung befindet und der Fokus dieser Arbeit auf der Methodik und dem Tool von Bachmann liegt, ist die Optimierungsphase der Migrationsmethodik in unserer Evaluation nicht anwendbar. Folgende Schritte, die bei der Migration eine Rolle spielen, sind:

- Auswahl des Migrationsszenarios

- Beschreibung der gewünschten Cloud Data Hosting Solution
- Auswahl des Cloud Data Stores
- Beschreibung des Quelldatenspeichers
- Identifizierung von Pattern, um mögliche Migrationskonflikte zu erkennen
- Refactoring der Applikation Architektur
- Datenmigration
- Test

4.6.2. Iteration 2 - Amazon Web Services

In der zweiten Iteration unseres Evaluationsplans werden wir sowohl das Migrationsszenario **MD** (siehe Abschnitt 4.5.1) als auch das Migrationsszenario **MG** (siehe Abschnitt 4.5.3) durchführen und folglich die beiden Datenbanken und die NovaERM Webapplikation in die Amazon Web Services Cloud migrieren. Für die Durchführung des ersten Szenarios **MD** gelten dieselben Bedingungen wie im vorherigen Abschnitt beschrieben. Für das zweite Szenario **MG** werden wir die phasengetriebene Migrationsmethodik von AWS benutzen, wobei die Datenmigrationsphase durch die Migrationsmethodik von Bachmann ersetzt wird (siehe Abbildung 4.4). Außerdem wird auf die beiden letzten Schritte der AWS Migrationsmethodik verzichtet, da sie für unsere Fallstudie nicht relevant sind. Folgende Schritte der AWS Migrationsmethodik sind durchzuführen:

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

4.7. Fragenkatalog

Wie bereits in den Abschnitten 3.2.3 und 4.3 beschrieben, wurde im Rahmen dieser Arbeit ein Fragenkatalog erstellt. Dieser ist von großer Bedeutung bei der Beantwortung von relevanten Fragen bezüglich der Migration von NovaERM in die Cloud. Des Weiteren soll mit Hilfe des Fragenkataloges ermöglicht werden auf Grund der Applikation Architektur den passenden Cloud Anbieter auszuwählen sowie herauszufinden wie dieser anzupassen ist.

Der Fragenkatalog wird für die Migrationsszenarien, die in den Abschnitten 4.5.1 und 4.5.3 vorgestellt wurden, ausgefüllt (siehe Anhang A, B). Der ausgefüllte Katalog ist im Anhang dieser Arbeit beigefügt.

5. Implementierung

Dieses Kapitel stellt die Implementierungsaspekte dieser Arbeit vor.

5.1. Implementierung eines Plugins in automaIT

Im Abschnitt 2.4 haben wir das Provisionierungswerkzeug automatIT [26] vorgestellt. Im Rahmen unserer Fallstudie wird dieses Werkzeug sowohl zum Aufsetzen des Grundsystems als auch während der Durchführung der einzelnen Migrationsszenarien zum Aufsetzen der lokal gehosteten Komponenten und auch zur Bestimmung der Baseline verwendet.

Um dies zu ermöglichen, werden in automaIT die Modelle gebildet. Jedes Modell setzt sich aus mehreren Teilen zusammen und wird in Form eines Plugins auf dem automaIT Server installiert. Die einzelnen Teile eines Plugins sind in der folgenden Abbildung 5.1 dargestellt:

```
/
├─ components/
├─ plans/
├─ resources/
└─ plugin-descriptor.xml
```

Abbildung 5.1.: automaIT Modell Komponenten

Das Hauptelement jedes Plugins ist der *Plug-In Descriptor*. Dieser beschreibt die Eigenschaften sowie die einzelnen Teile aus denen jedes Plugins besteht. Komponenten, Pläne oder Ressourcen, die in der *Plug-In Descriptor* nicht beschrieben sind, werden vom automaIT Server nicht erkannt.

Eine Komponente ist vergleichbar mit einer Klasse in einer objektorientierten Sprache und besteht aus Variablen, einem Konstruktor, einer oder mehreren Methoden und einem Destruktor. Ähnlich wie in der objektorientierten Programmierung, können mit Hilfe von Komponenten, Software-Artefakten wie Datenbank Instanzen, Web Server, Web Container usw. abgebildet werden. Die Komponenten können mittels Plänen auf die Agents installiert werden (siehe Abschnitt 2.4). Pläne enthalten Anweisungen, die in einer zuvor bestimmten Reihenfolge ausgeführt werden können. Diese ermöglichen nicht nur die Installation und Deinstallation von Komponenten, sondern erlauben die Ausführung von Kontrollmethoden auf den Agents. Die Methoden müssen in den jeweiligen Komponenten definiert sein. Des Weiteren enthält

jedes Plugin auch Ressourcen. Die Ressourcen werden durch Dateien oder Verzeichnisse repräsentiert und werden in Komponenten referenziert[1].

Im Rahmen unserer Arbeit wurde auch ein automaIT Plugin entwickelt, mit dessen Hilfe wir die einzelne Komponente von NovaERM provisionieren können. Wie bereits beschrieben besteht jedes Plugin aus Komponenten, Plänen, Ressourcen und einem *Plug-In Descriptor*. In den folgenden Abschnitten werden wir die einzelnen Komponenten unseres Plugins detailliert beschreiben.

5.1.1. automaIT NovaERM Plugin - Komponente

Wie bereits beschrieben ist die Komponente mit einer Klasse in einer objektorientierten Sprache vergleichbar. Für unser Plugin sind insgesamt elf einzelne Komponenten entstanden:

1. **ActivitiDBSchema**: Diese Komponente referenziert die Ressource *activitiPostgresDBSchema* (siehe Abschnitt 5.1.3) und stellt das Datenbank Schema der Actitiviti Datenbank dar.

2. **ActivitiDBSchemaWithData**: Die Komponente referenziert eine weitere Ressource *activitiPostgresDBSchemaWithData* (siehe Abschnitt 5.1.3) und stellt das Datenbank Backup der Actitiviti Datenbank dar.

3. **DeployWebApp**: Mit Hilfe der *DeployWebApp* Komponente wird es ermöglicht, dass die Webapplikation NovaERM in den Web Container deployt oder undeployt wird.

4. **Glassfish**: Bei dieser Komponente handelt es sich um den GlassFish Web Container[2]. In dieser sind alle wichtigen Methoden enthalten, um den Web Container zu verwalten wie z.B. eine Start- und Stopmethode, Abfragen des Web Container Domain Status, usw.

5. **ImportActivitiDB**: Diese Komponente ermöglicht den Import des Activiti Datenbank (DB) Schemas bzw. Backups in die laufende Instanz des PostgreSQL Datenbank Servers.

6. **ImportNovaERMDB**: Diese Komponente ermöglicht den Import des NovaERM DB Schemas bzw. Backups in die laufende Instanz des PostgreSQL Datenbank Servers.

7. **JavaDevelopmentKit**: Die Java Development Kit (JDK) Komponente ermöglicht die Installation bzw. die Deinstallation des JDK.

8. **NovaERMDBSchema**: Diese Komponente referenziert die Ressource *novaermPostgresDBSchema* (siehe Abschnitt 5.1.3) und stellt die Datenbank Schema der NovaERM Datenbank dar.

9. **NovaERMDBSchemaWithData**: Die Komponente referenziert eine weitere Ressource *novaermPostgresDBSchemaWithData* (siehe Abschnitt 5.1.3) und repräsentiert die Datenbank Backup der NovaERM Datenbank.

[1] automaIT: http://www.automait.de/downloads/
[2] GlassFish: https://glassfish.java.net/

10. **NovaERMWar**: Diese Komponente referenziert die Ressource *novaermweb* (siehe Abschnitt 5.1.3) und stellt die NovaERM Webapplikation dar.
11. **Postgresql**: Bei dieser Komponente handelt es sich um den PostgreSQL Datenbank Server[3]. In dieser Komponente sind alle wichtigen Methoden enthalten, um den Datenbank Server zu verwalten wie z.B. eine Start-, Stop- und eine Restartmethode, Abfragen des Datenbank Server Status, Benutzer Verwaltung, usw.

5.1.2. automaIT NovaERM Plugin - Pläne

Um die einzelnen Komponente auf den Agents zu einem gesamten System zusammen zu bauen, werden Pläne benötigt. Durch diese wird die Reihenfolge bestimmt, in welcher die Komponenten installiert werden. Um alle möglichen Szenarien, die während unserer Evaluation benötigt werden, abzubilden, sind insgesamt vier Pläne entstanden:

1. **InstallNovaERM**: Dieser Plan ermöglicht die Installation der gesamten NovaERM Webapplikation Umgebung. Von dem JDK, über den GlassFish Web Container, bis hin zu den PostgreSQL Datenbank Servern und dem Import der benötigten Testdaten in die jeweilige Datenbank.
2. **InstallNovaERMWithoutDB**: Dieser Plan begrenzt sich auf die Installation der NovaERM Webapplikation. Wie bereits bei dem vorherigen Plan beschrieben, werden alle Komponenten installiert, ausgenommen des PostgreSQL Datenbank Servers.
3. **InstallPostgresDB**: Mit Hilfe dieses Plans lässt sich eine Installation des PostgreSQL Datenbank Servers durchführen.
4. **InstallPostgresDBExtended**: Mit Hilfe dieses Plans lässt sich eine Installation des PostgreSQL Datenbank Servers durchführen. Des Weiteren werden Betriebssystem spezifische Einstellungen wie z.B. den Namen des Paketmanager auszuwählen, Name des PostgreSQL Services zu definieren oder z.B. den Installationspfad des Datenbank Servers zu bestimmten, ermöglicht. Diese Einstellungen erlauben eine Installation auf unterschiedlichen Betriebssystemen wie Ubuntu, Debian, CentOS, Red Hat Enterprise Linux (RHEL) usw.

5.1.3. automaIT NovaERM Plugin - Ressourcen

Ressourcen werden durch Dateien oder Verzeichnisse in einem automaIT Plugin repräsentiert und in Komponenten referenziert. Um eine erfolgreiche Installation von NovaERM zu ermöglichen benötigen wir fünf Ressourcen.

1. **activitiPostgresDBSchema**: Diese Ressource repräsentiert das Activiti Datenbank Schema, das für die Erstellung der Activiti Datenbank zuständig ist. Das Schema beschreibt insgesamt 22 Tabellen, dabei bestehen zwischen diesen sowohl 1:n als auch n:m Beziehungen. Die Größe dieser Ressource beträgt 37,2 KB.

[3]PostgreSQL: http://www.postgresql.org/

2. **activitiPostgresDBSchemaWithData**: Diese Ressource repräsentiert das Activiti Datenbank Backup und enthält Testdaten. Die Größe dieser Ressource beträgt 251 KB.
3. **novaermPostgresDBSchema**: Diese Ressource stellt das NovaERM Datenbank Schema dar, das für die Erstellung der Novaerm Datenbank zuständig ist. Das Schema beschreibt insgesamt 66 Tabellen, dabei bestehen zwischen diesen sowohl 1:n als auch n:m Beziehungen. Die Größe dieser Ressource beträgt 92,4 KB.
4. **novaermPostgresDBSchemaWithData**: Diese Ressource repräsentiert das NovaERM Datenbank Backup und enthält Testdaten. Die Größe dieser Ressource beträgt 1,89 MB.
5. **novaermweb**: Diese Ressource stellt die NovaERM Webapplikation dar. Die Größe der Webapplikation beträgt 35,8 MB.

5.1.4. automaIT NovaERM Plugin - *Plug-In Descriptor*

Mittels des *Plug-In Descriptor* werden die einzelnen Komponenten im Plugin beschrieben und somit sichtbar gemacht. Außerdem können Abhängigkeiten definiert und Ressourcen einzelnen Komponenten zugewiesen werden.

5.1.5. automaIT NovaERM Plugin - Entwicklung

Die Einarbeitung und die Entwicklung des Plugins für die Provisionierung von NovaERM wurde zur Beginn dieser Arbeit durchgeführt. Die Dauer dieser Phase beträgt insgesamt vier Wochen.

5.2. Durchführung von Performance Test mit inspectIT und seleniumHQ

Um eine Performance Messung während unserer Evaluation durchzuführen und um das Ergebnis der Migration zu überprüfen, werden im Rahmen dieser Arbeit das APM Tool *inspectIT* und das Tool *seleniumHQ* verwendet. Eine Kombination der beiden Werkzeuge ermöglicht die Erzeugung von Last mit dem Ziel den Betrieb von NovaERM zu simulieren und gleichzeitig eine Perfomance Messung durchzuführen. Die wichtigsten Einstellungen, um beiden Tools zu verwenden, sind im Folgenden detailliert beschrieben.

5.2.1. inspectIT Aufsetzen

Wie bereits im Abschnitt 2.5 beschrieben, besteht *inspectIT* aus drei Komponenten. Da der Client nur die Benutzeroberfläche darstellt und keine besonderen Einstellungen erfordert, wird dieser nicht weiter betrachtet. Das CMR stellt eine Serverkomponente dar und ist für die Sammlung aller Messdaten zuständig, die während der Messung von dem *inspectIT*

Agent erfasst werden. Die Verbindung zwischen dem *inspectIT* Client und dem *inspectIT* CMR erfolgt indem das CMR in dem Client hinzugefügt wird. Eine detaillierte Beschreibung ist auf der Webseite der *inspectIT* Dokumentation zu finden[4].

Der Agent stellt die dritte Komponente von *inspectIT* dar und ist für die Erhebung der Daten zuständig. Die Integration des Agent in die überwachte Anwendung erfolgt mittels Startup Parametern, die an die Java Virtual Machine (JVM) übergeben werden. Im Rahmen unserer Arbeit muss der *inspectIT Agent* in den GlassFish Web Container, in dem die NovaERM Webapplikation deployt ist, integriert werden. Da der GlassFish Web Container über mehreren Domains[5] verfügen kann, müssen die Startup Parameter in der entsprechenden Domain Konfiguration eingebunden werden. Diese befinden sich in dem Konfigurationsverzeichnis der NovaERM Domain - *glassfish/domains/novaerm/config/domain.xml*. Die benutzte Domain Konfiguration mit den *inspectIT Agent* Startup Parametern ist in Listing 5.1 zu sehen. Darüber hinaus verfügt jeder *inspectIT Agent* über eine Konfigurationsdatei in der alle genutzten Sensoren und die Adresse des CMR definiert werden. Aufgrund der umfassenden Einstellungsmöglichkeiten bezüglich der Sensoren verzichten wir auf eine detaillierte Beschreibung innerhalb unserer Arbeit und verweisen auf die Dokumentationsseite von *inspectIT*[6].

```
1  <java-config debug-options="-Xdebug␣-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address
       =9009" debug-enabled="true" system-classpath="" classpath-suffix="">
2    <jvm-options>-XX:MaxPermSize=192m</jvm-options>
3    <jvm-options>-XX:PermSize=64m</jvm-options>
4    <jvm-options>-client</jvm-options>
5    <jvm-options>-Djava.awt.headless=true</jvm-options>
6    <jvm-options>-Djavax.management.builder.initial=com.sun.enterprise.v3.admin.
         AppServerMBeanServerBuilder</jvm-options>
7    ...
8    <jvm-options>-Dinspectit.config=PATH-TO-INSPECTIT-AGENT/config</jvm-options>
9    <jvm-options>-Xbootclasspath/p:PATH-TO-INSPECTIT-AGENT/inspectit-agent.jar</jvm-options>
10   <jvm-options>-javaagent:PATH-TO-INSPECTIT-AGENT/inspectit-agent.jar</jvm-options>
11 </java-config>
```

Listing 5.1: *inspectIT Agent* Startup Parameter

Die während des Installationsvorganges der einzelnen *inspectIT* Komponenten aufgetretenen Herausforderungen und Probleme und wie diese gemeistert wurden ist im Abschnitt 7.2 beschrieben.

5.2.2. seleniumHQ Aufsetzen

Für die Durchführung der UI Tests im Rahmen dieser Arbeit wird das Tool *seleniumHQ* (siehe Abschnitt 2.6) benutzt. Die einzelnen *seleniumHQ* UI Tests wurden von dem Entwicklungs-

[4] *inspectIT* Adding a CMR: https://documentation.novatec-gmbh.de/display/INSPECTIT/Adding+a+CMR
[5] GlassFish Domains: http://docs.oracle.com/cd/E19798-01/821-1751/ggnop/
[6] *inspectIT* Agent Configuration: https://documentation.novatec-gmbh.de/display/INSPECTIT/Agent+Configuration

team von NovaERM zur Verfügung gestellt. Diese werden mit Hilfe des Build-Management-Tool *maven*[7] gestartet. Der Befehl zum Starten der UI Test ist in Listing 5.2 zu sehen.

```
1 mvn clean install -Duitests
```

Listing 5.2: UI Tests Start Befehl

Die UI Tests beinhalten insgesamt 36 Testfälle. Diese decken das Anlegen einer Bewerbung, die Durchführung eines Bewerbungsprozesses, das Erstellen eines Vertragsangebotes sowie die Nutzung der Suchfunktionen in NovaERM ab. Eine Auflistung der einzelnen Testfälle ist im Folgenden aufgelistet, wobei zu beachten ist, dass der erste Testfall mehrere einzelne Testfälle beinhaltet und einen vollständigen Bewerbungsprozess simuliert.

- *DoProzessSequenziell*
- *BewerbungAnlegen*
- *BewerbungVorqualifizieren*
- *BewerbungVervollstaendigen*
- *MeinungAbgeben*
- *BewerbungNachqualifizieren*
- *TerminPlanen*
- *TerminVereinbaren*
- *BewerbungsgespraechNachbearbeiten*
- *VAErstellenGraduand*
- *VAFinalisierenGraduand*
- *VAErstellenPracticalSchool*
- *VAFinalisierenPracticalSchool*
- *VAErstellenPracticalTrainee*
- *VAFinalisierenPracticalTrainee*
- *VAErstellenRegularEmployment*
- *VAFinalisierenRegularEmployment*
- *VAErstellenTempStaff*
- *VAFinalisierenTempStaff*
- *VAErstellenTrainee*

[7] Apache Maven: http://maven.apache.org/

- *VAFinalisierenTrainee*
- *VAErstellenTraineeCE*
- *VAFinalisierenTraineeCE*
- *VAErstellenWorkingStudent*
- *VAFinalisierenWorkingStudent*
- *AblehnenInAnlegen*
- *AblehnenInVorqualifizieren*
- *AblehnenInNachqualifizieren*
- *AblehnenInTerminVereinbaren*
- *AblehnenInTerminPlanen*
- *AblehnenInGespraechNachbearbeiten*
- *AblehnenInVAErstellen*
- *AblehnenMehrereBewerbungenInUebersicht*
- *AblehnenInVAFinalisieren*
- *FindMitarbeiterInUebersicht*
- *FindBewerberInUebersicht*

5.3. Erweiterung Tool von Bachmann

Wie bereits im Abschnitt 2.8 beschrieben, eignet sich das Tool von Bachmann nicht für die Migration einer PostgreSQL Datenbank, da kein Source und Target Adapter für PostgreSQL existiert. Damit diese ermöglicht wird, mussten wir im Rahmen unserer Arbeit die Funktionalität des Tools erweitern. Wie Bachmann in seiner Arbeit beschreibt, ist das Cloud Datamigrations-Tool modular und erweiterbar aufgebaut und besteht aus zwei Komponenten: Decision Support Tool und Migration Tool. Eine Abbildung des erweiterten Tools ist in Abbildung 5.2 zu sehen. Die zwei in rot gekennzeichneten Adapter ermöglichen die Migration der PostgreSQL Datenbank. Eine detaillierte Beschreibung erfolgt in den nächsten zwei Abschnitten.

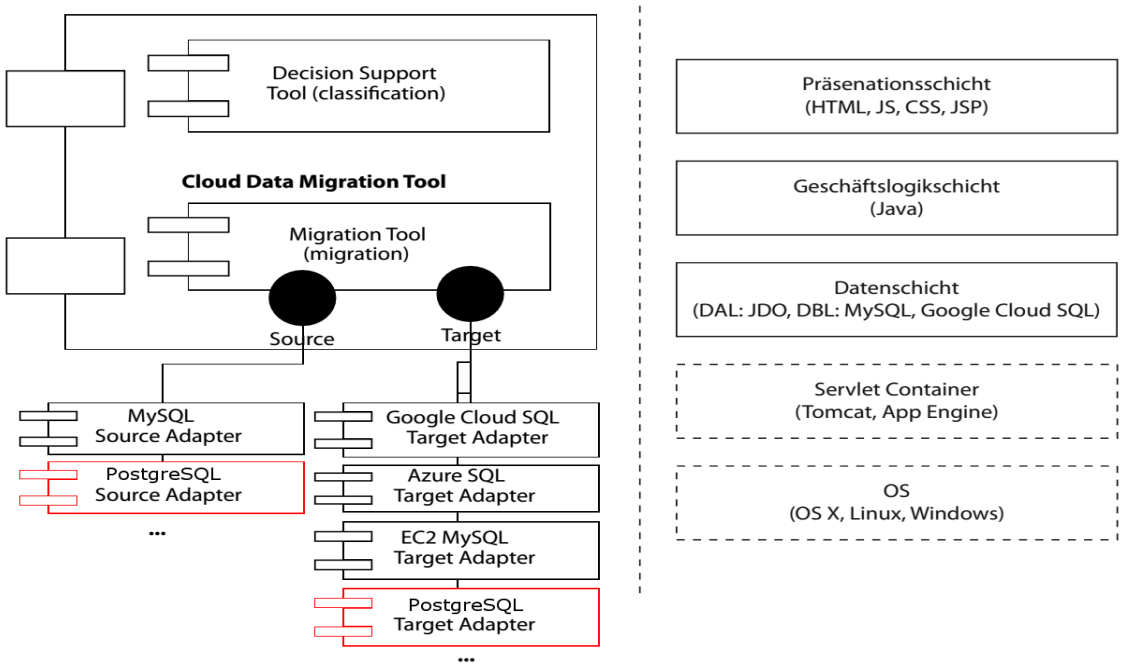

Abbildung 5.2.: UML Komponenten- und Schichtendiagramm des Cloud Data Migration Tools Erweitert [39]

5.3.1. PostgreSQL Source Adapter

Der PostgreSQL Source Adapter erweitert die Funktionalität des Cloud Datamigrations-Tools und ermöglicht die Migration einer PostgreSQL Datenbank. Das Tool benutzt das Kommandozeilenwerkzeug "psql", um eine Verbindung mit dem Quelldatenbankserver herzustellen. Falls diese erfolgreich ist, wird mittels des Kommandozeilenwerkzeugs "pg_dump" einen Datenbank Dump erstellt.

Der Befehl zum Testen der Verbindung mit dem Quelldatenbankserver ist in Listing 5.3 zu sehen.

```
1 $ PGPASSWORD=Password PfadZuPostgreSQLBinaries/psql -h Hostname -p Port -U Username -d
      Datenbankname -c"\\conninfo"
```

Listing 5.3: Test der Verbindung mit dem PostgreSQL Datenbank Server

Der Befehl zum Erstellen eines Datenbank Dumps ist in Listing 5.4 zu sehen.

Als *ExportOption* sind zwei Werte möglich. Entweder *–schema-only* oder *–data-only*. Dieser Wert kann leer sein. Somit wird es ermöglicht, dass nur bestimmte Teile der Datenbank migriert werden - entweder das Datenbank Schema oder nur die Datenbank Daten ohne das Schema. Falls der Wert leer ist, wird ein gesamter Dump der Datenbank erstellt.

```
1 $ PGPASSWORD=Password PfadZuPostgreSQLBinaries}/pg_dump -h Hostname -p Port -U Username
      ExportOption -E Encoding Datenbankname
```

Listing 5.4: Erstellen eines Datenbank Dumps

5.3.2. PostgreSQL Target Adapter

Der PostgreSQL Target Adapter dient dem Import der Daten in eine Zieldatenbank. Dieser nutzt wiederum das Kommandozeilenwerkzeug "psql", um die Verbindung mit dem Zieldatenbankserver zu testen und falls erfolgreich, den Datenbank Dump einzuspielen. Da im Moment nur eine Kompatibilität mit dem PostgreSQL Source Adapter besteht, werden keine weitere Anpassungen durchgeführt. Der Befehl zum Abspielen des Datenbank Dumps ist in Listing 5.5 zu sehen. Abhängig von dem Sicherheitseinstellungen des Datenbank Servers

```
1 $ PGPASSWORD=Password PGCLIENTENCODING=Encoding PfadZuPostgreSQLBinaries/psql -h Hostname -p
      Port -U Username -d Datenbankname -q -f PfadZuPostgreSQLDumpFile
```

Listing 5.5: Importieren eines Datenbank Dumps

kann das Passwort sowohl bei dem Source als auch bei dem Target Adapter leer sein. Falls dieser Fall zutrifft, wird in jeden Befehl zusätzlich der Parameter *–no-password* übergeben.

6. Evaluation Datenerhebung und Datenverarbeitung

Dieses Kapitel stellt die Durchführung des Evaluationsprozesses dar, basierend auf dem Evaluationsplan beschrieben im Abschnitt 1.3. Zum Beginn der Evaluation als auch zu den einzelnen Iterationen des Evalutionsplanes wird die Baseline immer durch das Tool *automaIT* bestimmt. Im zweiten Schritt der beiden Iterationen werden die zwei PostgreSQL Datenbanken von NovaERM mit Hilfe der Methodik und dem Tool von Bachmann jeweils in die NovaTec und Amazon Cloud migriert. Dabei werden alle Phasen der Migrationsmethodik von Bachmann (siehe Abschnitt 2.7) evaluiert. Die Migration in die NovaTec Cloud wird in Abschnitt 6.2 beschrieben. Dagegen wird im Abschnitt 6.3 die Migration in die Amazon Cloud in Form von Abweichungen zum Vorgehen bei der NovaTec Cloud Migration erläutert. Beide Migrationen realisieren das Migrationsszenario **MD** beschrieben im Abschnitt 4.5.1.

Im zweiten Schritt der Iteration 2 (siehe Abschnitt 1.3.2) wird die gesamte NovaERM Webapplikation in die Amazon Cloud migriert. Im Laufe dieses Schrittes wird die phasengetriebene Migrationsmethodik von AWS in Kombination mit der Migrationsmethodik von Bachmann verwendet. Die Zusammensetzung der beiden Methodiken zu eine ganzheitlichen Methodik ist im Abschnitt 4.5.3 erläutert.

Eine Spezifikation der einzelnen Software Komponenten, die während der Evaluation Durchführung verwendet werden, ist in der Tabelle 6.1 zu sehen. Eine Abweichung der Version während der Migrationsszenarien wird in den einzelnen Abschnitten gegeben.

Software	Version
GlassFish	3.2.1
JDK	JDK 7u25
PostgreSQL	9.1.9
NovaERM	beta0.10.21
automaIT	1.2
inspectIT	1.4
seleniumHQ	2.34
activiti	5.12.1

Tabelle 6.1.: Evaluation Vorbereitung Software Komponenten

6.1. Evaluation Vorbereitung

Vor dem Beginn der eigentlichen Evaluation, muss die Evaluationsumgebung vorbereitet werden. Die Durchführung der Vorbereitung erfolgt in zwei aufeinanderfolgenden Schritten. Im ersten Schritt wird mittels *automaIT* die Webapplikation NovaERM und der dazugehörige PostgreSQL Datenbank Server lokal installiert. Der zweite Schritt beinhaltet eine Performance Messung mit Hilfe von *instectIT*. Die einzelnen Schritten werden in den folgenden Abschnitten detailliert beschrieben.

6.1.1. Aufsetzen von NovaERM

Im ersten Schritt der Vorbereitung wird mittels *automaIT* (siehe Abschnitt 2.4) die Webapplikation samt der beiden dazugehörigen Datenbanken auf eine lokale Maschine aufgesetzt. Die Installation von NovaERM wird durch den zuvor implementierten Plan - *InstallNovaERM* durchgeführt (siehe Abschnitt 5.1.2). Dieser Plan installiert in mehreren aufeinanderfolgenden Schritten die benötigten Komponenten (siehe Abschnitt 5.1.1).

Der gesamte Installationsvorgang benötigt 08:58 Minuten, wobei zu beachten ist, dass die Komponente wie z.B. den GlassFish Web Container (Download Zeit: 04:45 Min.) und das JDK (Download Zeit: 01:10 Min.), vom Internet heruntergeladen werden. Die Dauer des Starts- und Deployvorgangs der NovaERM Webapplikation beträgt 01:44 Minuten von der gesamte Zeit. Der Import des Datenbank Schemas und der Daten für die beiden Datenbanken hat insgesamt 16 Sekunden von der gesamte Zeit in Anspruch genommen, wobei die Zeit zwischen der Activiti und der NovaERM Datenbank wie folgt aufgeteilt ist: 6 Sek. Activiti Datenbank, 10 Sek. NovaERM Datenbank.

6.1.2. NovaERM - Performance Diagnose

Nach der erfolgreichen Installation von NovaERM, wird im zweiten Schritt der Evaluationsvorbereitung, eine Perfomance Diagnose durchgeführt. Wie im Abschnitt 4.1.2 beschrieben, wird als APM Werkzeug inspectIT verwendet. Um Last zu erzeugen und damit den Betrieb von NovaERM zu simulieren, führen wir mit Hilfe von seleniumHQ (siehe Abschnitt 2.6), einen vollständigen GUI Test durch. Es werden insgesamt 36 Testfälle getestet (siehe Abschnitt 5.2.2). Diese decken das Anlegen einer Bewerbung, die Durchführung eines Bewerbungsprozesses, das Erstellen eines Vertragsangebots sowie die Nutzung der Suchfunktionen in NovaERM ab.

Die Dauer des gesamten Testdurchlaufs mit Hilfe von *inspectIT* und *seleniumHQ* beträgt 29:46 Minuten. Dabei wurden insgesamt 211 Unique SQL Statements durchgeführt, wobei die Dauer eines Statements zwischen 1 Millisekunden und 24 Sekunden in Anspruch genommen hat. Die Unique SQL Statements beinhalten 6 *DELETE*, 35 *UPDATE*, 50 *INSERT* und 120 *SELECT* Befehle. Des Weiteren beträgt die Anzahl aller durchgeführten Statements 50093.

Weiterhin wurden 43 Unique Adressen innerhalb der NovaERM Webapplikation aufgerufen, wobei die gesamte Anzahl aller HTTP Anrufe 3946 beträgt. Die Dauert der einzelnen Aufrufe beträgt zwischen 1 Millisekunden und 144 Sekunden.

Die Zeiten für die Durchführung der einzelnen Testdurchläufe beschrieben im Abschnitt 5.2.2 sind in der Tabelle E.1 im Anhang zu sehen.

6.2. Evaluationsplan - Iteration 1

In diesem Abschnitt werden wir die erste Iteration der Evaluation durchführen, indem wir dem Plan vorgestellt im Abschnitt 1.3.1 folgen. In zwei aufeinanderfolgenden Schritten werden wir die beiden Datenbanken von NovaERM unter der Verwendung der Migrationsmethodik und des Tools von Bachmann in die NovaTec Cloud (siehe Abschnitt 2.9) migrieren.

Um ein besseres Verständnis für den Migrationsprozess zu erzielen, werden wir jede Phase der Migrationsmethodik durchführen und in den folgenden Abschnitten näher erläutern.

6.2.1. Iteration 1: Baseline

Bevor wir mit der Migration beginnen, müssen wir die Baseline bestimmen. Diese wird uns einen umfassenden Vergleich ermöglichen, nachdem wir die Datenbanken erfolgreich migriert haben.

Ähnlich wie bei der Vorbereitung (siehe vorherigen Abschnitt 6.1) der Evaluation, wird bei der Bestimmung der Baseline der PostgreSQL Datenbank Server mit Hilfe des Tools *automaIT* auf der NovaTec Cloud VM installiert und die Testdaten für die zwei NovaERM Datenbanken importiert. Die Installation des Datenbankservers wird durch den Plan *InstallPostgresDBExtended* durchgeführt. Dieser Plan ermöglicht Datenbankspezifische Einstellungen vorzunehmen und folglich eine Vorbereitung für die spätere Migration durch das Tool von Bachmann, vorzunehmen. Die wichtigsten Einstellungen, die bei der Installation des PostgreSQL Servers vorzunehmen sind, sind in den folgenden Punkten aufgelistet:

- *Host-Based Authentication*[1]: Diese Einstellung beschränkt bzw. ermöglicht den Zugriff auf eine Datenbank durch einen bestimmten Benutzer von einem definierten Server. Außerdem ist es möglich, die Authentifizierungsmethode des Benutzers festzulegen. Alle diese Informationen werden in einer PostgreSQL spezifischen Datei abgelegt. Die Syntax sieht wie folgt aus:

  ```
  hostname DATABASE USER ADDRESS AUTH_METHOD [OPTIONS]
  ```

[1]The pg_hba.conf File: http://www.postgresql.org/docs/9.1/static/auth-pg-hba-conf.html

- *PostgreSQL User Password*: Eine weitere Einstellung, die im Laufe der Installation der Datenbankserver vorgenommen wird, ist die Änderung des Administratorpassworts der Datenbank. Diese Änderung wird mit Hilfe einer SQL Query durchgeführt:

  ```
  ALTER USER postgres PASSWORD 'password';
  ```

Der gesamte Installationsvorgang benötigt 01:55 Min., wobei der Import des Datenbank Schemas und der Daten für die beiden Datenbanken 42 Sekunden benötigt. Die Zeit zwischen der Activiti und der NovaERM Datenbank ist wie folgt aufgeteilt: 13 Sek. Activiti Datenbank, 29 Sek. NovaERM Datenbank.

6.2.2. Migrationsmethodik von Bachmann

Auswahl des Migrationsszenarios

Wie bei der Vorstellung der Migrationsmethodik im Abschnitt 2.7 beschrieben, ist die erste Phase in zwei Schritte unterteilt. Im ersten Schritt dieser Phase muss ein Migrationsszenario ausgewählt werden. Da wir uns am Anfang dieser Arbeit für *Reines Outsourcing des Datenbank Schicht* als Migrationsszenario entschieden haben, wählen wir dieses Szenario aus. Während des zweiten Schrittes kann die Migrationsstrategie definiert werden. Diese besteht aus einzelnen Eigenschaften, die mögliche Konflikte zu dem ausgewählten Szenario identifizieren können. Zu unserer Migrationsstrategie definieren wir folgenden Eigenschaften[2] [3]:

- *Migration Strategie*: Non-Live (with downtime)
- *Local DBL*: Moved
- *Migration Degree*: Complete
- *Source Data Store Type*: Relational Database Management System (RDBMS)
- *Target Data Store Type*: RDBMS
- *Product/Version Change*: Same product, same version
- *Direction of Data Movement*: From local datacenter to cloud
- *Utilization Increase*: Slow

Abhängig von den definierten Eigenschaften bei der Migrationsstrategie werden am Ende dieser Phase Konflikte zu dem ausgewählten Migrationsszenario erkannt. Die Konflikte sowie deren mögliche Lösungen werden in Form einer kategorisierten Liste im Tool angezeigt.

Fazit: Im Laufe dieser Phase traten keine Probleme auf und die Strukturierung der einzelnen Schritte ist verständlich und nachvollziehbar.

Zeitaufwand: 5 Minuten.

[2] Die Namen der einzelnen Eigenschaften sind von dem Tool direkt übernommen.
[3] Wegen der großen Zahl von Eigenschaften, sind nur die Wichtigsten aufgelistet

Beschreibung der gewünschten Cloud Data Hosting Solution

In der zweiten Phase der Migrationsmethodik können wir funktionale und nicht-funktionale Anforderungen bezüglich des Zieldatenspeichers identifizieren. Die ausgewählten Anforderungen während dieser Phase sind in der Tabelle 6.2 zu sehen[3].

Fazit: Im Laufe dieser Phase trat nur ein Problem auf (siehe Tabelle 6.3). Die Strukturierung der einzelnen Schritte ist verständlich und nachvollziehbar.

Zeitaufwand: 7 Minuten.

Kategorien	Eigenschaft	Werte
Scalability	Degree of Automation Automated Scalability	Manual None
Security	Storage Encryption Transfer Encryption Firewall Authentication Confidentiality Integrity Authorization	No No No Yes Yes Yes No
Location	Cloud Location	On-Premise
Compatibility	Product including Version	PostgreSQL, Version 9.1.9
Storage	Storage Type	RDBMS
Cloud Computing	Service Model Deployment Model	IaaS Private Cloud
Management and Maintenance effort	Degree of Automation	Self-Service
Monitoring	Supported KPIs	CPU\RAM Load, Data Transfer - Network\Disk (I/O)

Tabelle 6.2.: Auswahl von funktionalen und nicht-funktionalen Anforderungen während der zweiten Migrationsphase in Iteration 1

Auswahl des Cloud Data Stores

Nachdem wir im zweiten Schritt der Methodik die Anforderungen an den Zieldatenspeicher erfasst haben, können wir in dieser Phase den passenden Cloud Data Store auswählen. Da dieser nicht in der Liste der empfohlenen Stores vorhanden war, mussten wir vorher eine entsprechenden Store anlegen. Als Cloud Data Store haben wir *PostgreSQL Server* ausgewählt.

Fazit: Im Laufe dieser Phase traten mehrere Probleme auf (siehe Tabellen 6.4 und 6.5). Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 10 Minuten.

ID	MD 1
Name	Fehler beim Ausfüllen von Textfeldern
Fehlerzugehörigkeit	Tool (funktionale Korrektheit)
Schweregrad	mittel
Priorität	mittel
Reproduzierbarkeit	Der Fehler ist reproduzierbar. Ein Textfeld zum Ausfüllen auswählen.
Beschreibung	In der zweiten Phase der Migrationsmethodik, in der Kategorie *Compatibility*, kann die Version des Quelldatenbanksystem definiert werden. Beim Versuch das Textfeld auszufüllen, verliert das Feld den Fokus.
Fehlerbehandlung	Es besteht die Möglichkeit einen Doppelklick auszuführen, um den richtigen Fokus auf dem Textfeld zu setzen.
Fehlerlösung	Bei diesem Fehler handelt es sich um einen CSS Fehler. Eine detaillierte Beschreibung ist in Abschnitt 7.1.1 zu finden.
Kommentar	-

Tabelle 6.3.: Fehler MD 1

ID	MD 2
Name	Fehlender Cloud Data Store
Fehlerzugehörigkeit	Andere (Brauchbarkeit, Angemessenheit, Lernbarkeit)
Schweregrad	leicht
Priorität	niedrig
Reproduzierbarkeit	Das Problem ist reproduzierbar.
Beschreibung	In der dritten Phase der Migrationsmethodik kann kein Cloud Data Store ausgewählt werden, wenn noch keine passenden angelegt sind.
Fehlerbehandlung	Es besteht die Möglichkeit im Tool zu jedem Zeitpunkt einen neuen und entsprechenden Cloud Data Store anzulegen.
Fehlerlösung	Cloud Data Store anlegen
Kommentar	-

Tabelle 6.4.: Fehler MD 2

Beschreibung des Quelldatenspeichers

Um mögliche Konflikte zu dem Zieldatenspeicher zu erkennen, können in dieser Phase die Eigenschaften des Quelldatenspeichers spezifiziert werden. Eine Übersicht über die ausgewählten Eigenschaften ist in Tabelle 6.6 zu sehen.

Fazit: Im Laufe dieser Phase traten keine Fehler auf. Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 3 Minuten.

ID	MD 3
Name	Langsames Laden von vorhandenen Cloud Data Stores
Fehlerzugehörigkeit	Tool (Effizienz)
Schweregrad	leicht
Priorität	niedrig
Reproduzierbarkeit	Das Problem ist reproduzierbar. Dritte Phase in der Migrationsmethodik auswählen.
Beschreibung	Anhand der großen Anzahl von Kriterien zu einem Cloud Data Store, wird die Seite im Tool sehr langsam geladen.
Fehlerbehandlung	Warten bis alle vorhandene Stores geladen sind.
Fehlerlösung	Siehe Abschnitt 7.1.1
Kommentar	-

Tabelle 6.5.: Fehler MD 3

Identifizierung von Pattern, um möglichen Migrationskonflikte zu erkennen

In dieser Phase der Migrationsmethodik werden Konflikte zwischen dem Quell- und Zieldatenspeicher erkannt und Pattern als mögliche Lösung vorgeschlagen.

Das Cloud Datamigration Tool hat in dieser Phase, drei Konflikte erkannt (siehe Abbildung 7.1) und für jeden ein Pattern vorgeschlagen. Anhand der vorgeschlagenen Pattern wurde das erste *#1* umgesetzt.

Fazit: Im Laufe dieser Phase trat ein Verständlichkeitsproblem auf (siehe Tabelle 6.7). Die Strukturierung dieser Phase ist verständlich.

Zeitaufwand: 3 Minuten.

Refactoring der Applikation Architektur

Abhängig von der Auswahl des Migrationsszenarios im ersten Schritt der Methodik, werden in diesem Schritt, Empfehlungen zur Anpassung höherer Anwendungsschichten von dem Tool gemacht. Die einzelne Vorschläge sind in Kategorien unterteilt.

Im Folgenden werden die Kategorien sowie die umgesetzten Empfehlungen vorgestellt. Eine vollständige Auflistung der einzelnen Empfehlungen ist im Anhang C zu finden.

Kategorie - *Network Layer*

- *Firewall (Open Ports)*: Diese Empfehlung bezieht sich auf die Änderung der Zugriffsbeschränkungen des Zieldatenbank Systems. Diese sind bei der Bestimmung der Baseline (siehe Abschnitt 6.2.1) durch *automaIT* geändert worden und benötigen daher keine weiteren Anpassungen.

Kategorien	Eigenschaft	Werte
Characterization	Location of the source system	On-premise
	Deployment model	None (not hosted in the cloud)
	Service model	None (not hosted in the cloud)
	Store Type	RDBMS
	Accessibility over the network	Yes
RDBMS	Does the source system support stored procedures or trigger and are these used by the application?	No
	Does the source system support transactions and are these used by the application?	No
	Does the source system support Joins and are these used by the application?	No

Tabelle 6.6.: Auswahl von Eigenschaften während der vierten Phase

Kategorie - *Data Access Layer*

- *Connection String*: Dieser Schritt empfiehlt die Änderung der Verbindungsadresse in der Webapplikation, die auf die Datenbank zugreift.

Fazit: Im Laufe dieser Phase sind keine Fehler aufgetreten. Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 3 Minuten.

Datenmigration

In der letzten Phase der Migrationsmethodik wählen wir das Quell- und Zieldatenbank System aus und geben die Zugangsdaten ein. Sowohl als Quell- als auch Zieldatenbank ist PostgreSQL ausgewählt. Nachdem wir alle Zugangsdaten vollständig ausgefüllt haben, starten wir die Migration.

Fazit: Während dieser Phase sind mehrere Fehler aufgetreten. Die Beschreibung der einzelnen Fehler ist in den Tabellen 6.8, 6.9 und 6.10 zu finden. Die Strukturierung dieser Phase ist verständlich.

Zeitaufwand: 3 Minuten.

Test

Für die Testphase wurden insgesamt drei Durchläufe ausgeführt und die Mittelwerte berechnet. Die Dauer der einzelnen Testdurchläufe mit Hilfe von *inspectIT* und *seleniumHQ* beträgt zwischen 29:34 Minuten und 30:05 Minuten.

ID	MD 4
Name	Mehrmalige Erkennung desselben Konflikts.
Fehlerzugehörigkeit	Tool (funktionale Korrektheit, Angemessenheit)
Schweregrad	leicht
Priorität	niedrig
Reproduzierbarkeit	Das Problem ist reproduzierbar. Bei der Beschreibung des Quelldatenspeichers, die Frage *"Is the source system addressed on the network level via an URL that can be resolved by a DNS?"* positiv beantworten.
Beschreibung	Derselbe Konflikt tritt mehrmals in der Liste von möglichen Konflikten auf.
Fehlerbehandlung	Beide Konflikte besitzen unterschiedliche Lösungen.
Fehlerlösung	Siehe Abschnitt 7.1.1
Kommentar	Obwohl die beiden Konflikte unterschiedliche Lösungen besitzen, führt die mehrfache Auflistung von desselben Konflikts zu Unübersichtlichkeit und möglicher falscher Interpretation der empfohlenen Lösung.

Tabelle 6.7.: Fehler MD 4

ID	MD 5
Name	Erkennung von Sonderzeichen im Passwort.
Fehlerzugehörigkeit	Tool (Effektivität, funktionale Korrektheit, Brauchbarkeit)
Schweregrad	kritisch
Priorität	hoch
Reproduzierbarkeit	Das Problem ist reproduzierbar. Im Passwort sollen Sonderzeichen wie z.B. "\" oder "|" vorkommen.
Beschreibung	Im Passwort für den Zieldatenbank Server sind keine Sonderzeichen zugelassen.
Fehlerbehandlung	Das Passwort auf dem Zieldatenbank Server anpassen.
Fehlerlösung	Das Passwort auf dem Zieldatenbank Server ändern.
Kommentar	Eine detaillierte Beschreibung der Lösung ist im Abschnitt 7.1.1 zu finden.

Tabelle 6.8.: Fehler MD 5

Wie bereits im Abschnitt 2.6 beschrieben, werden die UI Test mit Hilfe des Tools *seleniumHQ*, basierend auf den vom NovaERM Entwicklungsteam zur Verfügung gestellten Testcases, durchgeführt. Dies verhindert die Verfälschung der Tests, da sie immer in derselben Reihenfolge ausgeführt werden. Deswegen werden wir die Ergebnisse dieser Phase nur in Form von Abweichungen zur Baseline, beschrieben im Abschnitt 6.1.2, darstellen.

Wie bei der Bestimmung der Baseline im Abschnitt 6.1.2 wurden während dieser Testphase

ID	MD 6
Name	Migrationsprozess Status
Fehlerzugehörigkeit	Tool (Effizienz, funktionale Angemessenheit)
Schweregrad	mittel
Priorität	mittel
Reproduzierbarkeit	Das Problem ist reproduzierbar. Den Migrationsprozess starten.
Beschreibung	Bei der Durchführung des Migrationsprozesses wird kein Status angezeigt.
Fehlerbehandlung	Es muss gewartet werden bis die *Migration Log Import* Konsole eine Nachricht zum Beenden des Prozesses ausgibt.
Fehlerlösung	-
Kommentar	Eine detaillierte Beschreibung einer möglichen Lösung ist im Abschnitt 7.1.1 zu finden.

Tabelle 6.9.: Fehler MD 6

ID	MD 7
Name	Button *Cancel* Funktionalität
Fehlerzugehörigkeit	Tool (funktionale Korrektheit, Angemessenheit, Brauchbarkeit)
Schweregrad	mittel
Priorität	mittel
Reproduzierbarkeit	Das Problem ist reproduzierbar. Sowohl in den einzelnen Schritten der Migrationsmethodik als auch am Ende der Migration, auf dem Button *Cancel* drücken.
Beschreibung	Das Drücken der *Cancel* Button, hat nicht den erwarteten Effekt. Während der einzelnen Schritten der Migrationsmethodik, anstatt zurück zur vorherige Seite zu wechseln, wird direkt zur allgemeinen Projekt Übersicht gewechselt. Am Ende des Migrationsprojektes führt ein Klick auf den *Cancel* Button zum erneuten Laden der Seite.
Fehlerbehandlung	Navigieren durch die Browser Buttons.
Fehlerlösung	Siehe Abschnitt 7.1.1
Kommentar	-

Tabelle 6.10.: Fehler MD 7

insgesamt 211 Unique SQL Statements ausgeführt. Die Dauer der einzelnen Statements beträgt zwischen 1 Millisekunden und 21 Sekunden.

Die Dauert der einzelnen HTTP Aufrufe beträgt zwischen 1 Millisekunden und 140 Sekunden.

Die Zeiten für die Durchführung der einzelnen Testfälle beschrieben im Abschnitt 5.2.2, sind

in der Tabelle E.2 im Anhang zu sehen.

6.2.3. Migration Datenverarbeitung

Im Laufe der ersten Iteration unseres Evaluationsplans sind keine kritischen Fehler aufgetreten. Deshalb sind für die weitere Verarbeitung der gesammelten Informationen während der Migration keine weitere Schritten notwendig.

Eine Zusammenfassung der genutzten Zeit pro Migrationsphase ist in der Abbildung 6.1 zu finden.

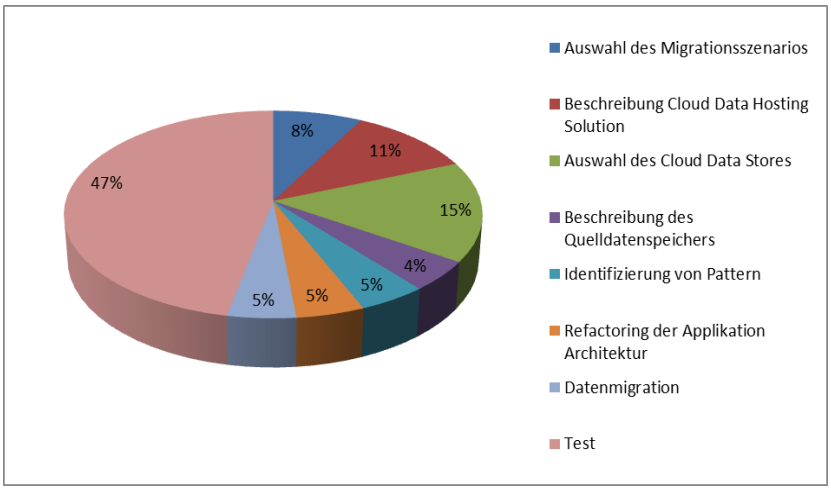

Abbildung 6.1.: Zusammenfassung genutzter Zeit während der ersten Iteration

6.3. Evaluationsplan - Iteration 2

Im Gegensatz zu der ersten Iteration, werden in diesem Abschnitt die Migrationsszenarien, beschrieben in den Abschnitten 4.5.1 und 4.5.3, durchgeführt. Da der Ablauf des ersten Szenarios bereits in der erste Iteration detailliert dargestellt ist, wird dieser im folgenden Abschnitt in Form von Abweichungen zu dem ersten erläutert. Dagegen wird bei der Durchführung des zweiten Migrationsszenarios eine ausführliche Beschreibung der einzelnen Schritten erfolgen.

6.3.1. Iteration 2: Baseline

Zu Beginn der zweite Iteration werden wir die Baseline wiederholt bestimmten. Um eine umfassende Basis zum Vergleich zu der ersten Iteration zu schaffen werden wir während der Zweiten in zwei aufeinanderfolgenden Schritten die Webapplikation und den PostgreSQL Datenbank Server in die Amazon Cloud installieren.

Wie bereits in der ersten Iteration werden wir für die Installation des PostgreSQL Datenbank Servers den Plan *InstallPostgresDBExtended* verwenden. Nach der erfolgreichen Installation des Datenbank Servers, installieren wir im zweiten Schritt mit Hilfe des Plans *InstallNovaERM-WithoutDB* die NovaERM Webapplikation.

Der Installationsvorgang der NovaERM Webapplikation ist in drei Schritte unterteilt. Nachdem im ersten und zweiten Schritt das JDK und der GlassFish Web Container installiert werden, wird im dritten Schritt die NovaERM Webapplikation in den Web Container deployt. Die wichtigsten Einstellungen, die während der Installation vorgenommen werden, sind in den folgenden Punkten näher erläutert:

- Einstellung des *JAVA_HOME* Pfads: Nachdem das JDK und der GlassFish Web Container installiert sind, wird der *JAVA_HOME* Pfad in der NovaERM Domain Konfigurationsdatei gespeichert[4].

- Verbindungsdaten für den PostgreSQL Datenbank Server: Bevor die NovaERM Webapplikation in den Web Container deployt wird, müssen die Verbindungsdaten der zwei Datenbanken in der Domain Konfigurationsdatei eingetragen werden[5].

Der Installationsvorgang des Datenbank Servers benötigte 01:09 Min., wobei der Import der Datenbank Schemas und der Daten für die Datenbanken 29 Sekunden von der Zeit in Anspruch genommen hat. Die Zeit zwischen der Activiti und der NovaERM Datenbank ist wie folgt aufgeteilt: 14 Sek. Activiti Datenbank, 15 Sek. NovaERM Datenbank. Weiterhin dauerte die Installation der NovaERM Webapplikation 10:41 Min. Diese Zeit beinhaltet die Downloadzeiten für das JDK 7 Sek. und der GlassFish Web Container 5:13 Min. sowie die Zeit für das Starten und Deployen der Webapplikation (4:51 Min.).

Die gesamt Installationszeit setzt sich aus der Zeit für die Installation des Datenbank Servers sowie die Zeit für die Installation der NovaERM Webapplikation und beträgt somit 11:49 Minuten.

6.3.2. Migrationsmethodik von Bachmann

In diesem Abschnitt wird die Durchführung der Migration der Activiti und NovaERM Datenbanken in die Amazon Cloud, in Form von Abweichungen zum Abschnitt 6.2.2 vorgestellt. Weiterhin wird in jedem Schritt der Migrationsphase die benötigte Zeit angegeben. Die Migration erfolgt mittels der Migrationsmethodik und des Tools von Bachmann (siehe Abschnitten 2.7, 2.8).

[4]Oracle GlassFish Server Administration Guide: http://docs.oracle.com/cd/E19798-01/821-1751/ghmbn/
[5]Oracle GlassFish Server 3.1 Application Deployment Guide: http://docs.oracle.com/cd/E18930_01/html/821-2417/giyil.html

Auswahl des Migrationsszenarios

In dieser Phase bestehen keine Unterschiede zum Abschnitt 6.2.2.

Fazit: Im Laufe dieser Phase sind keine Fehler aufgetreten. Die Strukturierung der Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 4 Minuten.

Beschreibung der gewünschten Cloud Data Hosting Solution

Die ausgewählten Eigenschaften zur Beschreibung der Cloud Data Hosting Solution sind in der Tabelle 6.11 zu sehen[6].

Fazit: Im Laufe dieser Phase trat nur ein Fehler auf (siehe Tabelle 6.3). Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 7 Minuten.

Kategorien	Eigenschaft	Werte
Availability	Replication	Yes (distributed infrastructure)
	Automatic Failover	Yes
Security	Storage Encryption	Yes
	Transfer Encryption	Yes
	Firewall	Yes
	Authentication	Yes
	Confidentiality	Yes
	Integrity	Yes
	Authorization	Yes
Location	Choice	No - Single Location
	Geogpraphic Location	Ireland
	Cloud Location	Off-Premise
Cloud Computing	Deployment Model	Public Cloud

Tabelle 6.11.: Auswahl von funktionalen und nicht-funktionalen Anforderungen während der zweiten Migrationsphase in Iteration 2

Auswahl des Cloud Data Stores

In dieser Phase bestehen keinen Unterschiede zum Abschnitt 6.2.2.

Fazit: Im Laufe dieser Phase sind keine Fehler aufgetreten. Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 1 Minuten.

[6]Wegen der großen Anzahl von Eigenschaften, sind nur die Unterschiede zur Phase 2 in Abschnitt 6.2.2 aufgelistet

Beschreibung des Quelldatenspeichers

In dieser Phase bestehen keinen Unterschiede zum Abschnitt 6.2.2.

Fazit: Im Laufe dieser Phase sind keine Fehler aufgetreten. Die Strukturierung dieser Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 2 Minuten.

Identifizierung von Pattern, um mögliche Migrationskonflikte zu erkennen

Während dieser Phase wurden zwei mögliche Konflikte zwischen dem Quell- und Zieldatenspeicher erkannt und für jeden eine Lösung in Form von Pattern vorgeschlagen. Die Pattern sind in der Abbildung 6.2 dargestellt. Anhand der vorgeschlagenen Pattern wurde das zweite #2 umgesetzt

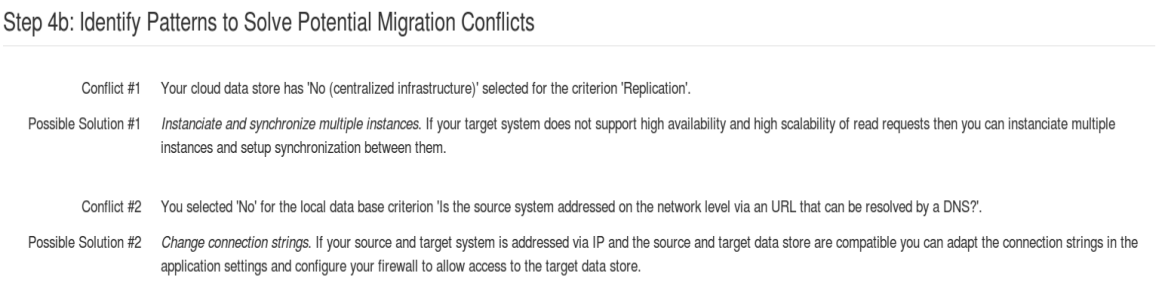

Abbildung 6.2.: Iteration 2: Migrationspattern

Fazit: Im Laufe dieser Phase traten keine Fehler auf. Die Strukturierung der einzelnen Schritten während der Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 2 Minuten.

Refactoring der Applikation Architektur

In dieser Phase bestehen keine Unterschiede zum Abschnitt 6.2.2.

Fazit: Im Laufe dieser Phase trat kein Problem auf. Die Strukturierung der Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 3 Minuten.

Datenmigration

In der letzten Phase der Migrationsmethodik haben wir als Quell- und Zieldatenspeicher den PostgreSQL Datenbank Server ausgewählt. Nachdem wir alle Zugangsdaten vollständig ausgefüllt haben, starten wir die Migration.

Fazit: Die Fehler, die während dieser Phase aufgetreten sind, sind in den Tabellen 6.8, 6.9 und 6.10 beschrieben. Die Strukturierung der Phase ist verständlich und nachvollziehbar.

Zeitaufwand: 3 Minuten.

Test

Für die Testphase wurden insgesamt zwei Durchläufe ausgeführt und von diesen immer die mittleren Werten berechnet. Die Dauer der einzelnen Testdurchläufe mit Hilfe von *inspectIT* und *seleniumHQ* beträgt zwischen 39:35 Minuten und 39:56 Minuten.

Wegen Netzwerk-Latenzen dauerten die beiden Testdurchläufe länger im Vergleich zu der ersten Iteration und der Vorbereitungsphase der Migration. Des Weiteren konnten nicht alle Testfälle durchgeführt werden. Eine Liste der UI Tests, die nicht durchgeführt werden können, ist im Folgenden zu sehen:

- *TerminPlanen*
- *TerminVereinbaren*
- *AblehnenInVorqualifizieren*
- *AblehnenInNachqualifizieren*
- *AblehnenInTerminVereinbaren*
- *AblehnenInTerminPlanen*
- *AblehnenInGespraechNachbearbeiten*
- *AblehnenInVAErstellen*
- *AblehnenMehrereBewerbungenInUebersicht*
- *AblehnenInVAFinalisieren*

Von den insgesamt 211 Unique SQL Statements während der ersten Iterationsphase und der Migrationsvorbereitungsphase, wurden während dieser Testphase nur 200 Unique SQL Statements ausgeführt. Die Dauer eines Statements hat zwischen 1 ms und 19 Sekunden in Anspruch genommen. Die Unique SQL Statements beinhalten 6 *DELETE*, 30 *UPDATE*, 50 *INSERT* und 114 *SELECT* Befehle. Des Weiteren beträgt die Anzahl aller durchgeführten Statements 62098.

Weiterhin wurden 43 Unique Adressen innerhalb der NovaERM Webapplikation aufgerufen, wobei die gesamte Anzahl aller HTTP Anrufe 4663 beträgt. Die Dauert der einzelnen Aufrufe beträgt zwischen 1 ms und 186 Sekunden.

Die Zeiten für die Durchführung der einzelnen Testdurchläufe beschrieben, im Abschnitt 5.2.2, sind in der Tabelle E.3 im Anhang zu sehen.

6.3.3. Datenverarbeitung Bachmann

Bei der durchgeführten Migration der DBL in die Amazon Cloud sind keine kritischen Fehler aufgetreten. Weiterhin ist eine Verarbeitung der gesammelten Informationen während der Migration nicht notwendig.

Eine Zusammenfassung der genutzten Zeit pro Migrationsphase ist in der Abbildung 6.3 zu finden.

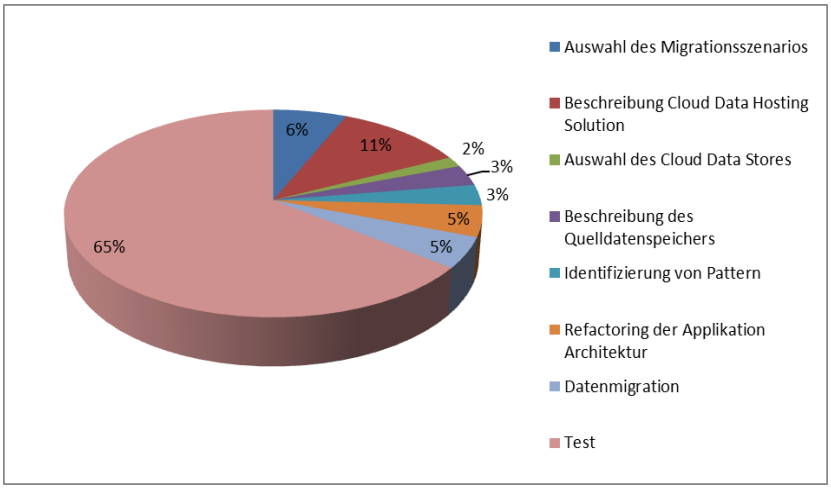

Abbildung 6.3.: Zusammenfassung genutzter Zeit während der zweiten Iteration

6.3.4. Migrationsmethodik AWS

In diesem Abschnitt wird die Durchführung des Migrationsszenarios **MG** (siehe Abschnitt 4.5.3) vorgestellt. Die Migration erfolgt mit Hilfe der phasengetriebenen Migrationsmethodik von AWS in Kombination mit der Migrationsmethodik und dem Migrationstool von Bachmann (siehe Abschnitt 4.5.3).

Cloud Assessment Phase

Die erste Phase der Migrationsmethodik von AWS beinhaltet Schritte zur Sammlung von Informationen für die Entscheidungsfindung anhand von Kriterien wie Sicherheit, Compliance, Wirtschaftlichkeit usw.. Weiterhin werden die Komponenten, die in die Cloud migriert werden sollen, identifiziert [22]. Diese Phase teilt sich in folgende Schritte:

- *Financial Assessment*[7]: Während des ersten Schrittes dieser Phase sind alle Wirtschaftlichkeitsfaktoren zu beurteilen, um die finanziellen Erwartungen bezüglich der Migration zu überprüfen. Da unserer Arbeit eine Fallstudie zu Grunde liegt, wird davon ausgegangen, dass die Entscheidung für die Migration in die Cloud bereits getroffen wurde.
- *Security and Compliance Assessment*[7]: Der zweite Schritt bewertet alle Sicherheitsrisiken der Migration und hilft bei der Identifizierung von potentiellen Bedrohungen bezüglich der Offenlegung von vertraulichen Informationen. Da sich NovaERM immer noch in Entwicklung befindet, werden während der Migration nur Testdaten in die Cloud migriert.
- *Technical Assessment*[7]: In diesem Schritt werden die technischen Aspekte einer Migration bewertet. Während dieses Schrittes sollen alle Abhängigkeiten der einzelnen Komponenten von NovaERM identifiziert und dargestellt werden. Des Weiteren sollen anhand dieser Abbildung die zu migrierenden Komponenten erkannt werden.

  Eine Abbildung der einzelnen Komponenten von NovaERM wurde bereit im Abschnitt 2.3 dargestellt. Weiterhin sind die zu migrierenden Komponenten in dem Migrationsszenario **MG** (siehe Abschnitt 4.5.3) identifiziert.
- *Identify the Tools That You Can Reuse*[7]: Während dieses Schrittes soll eine Entscheidung bezüglich der genutzten Cloud Dienste und Tools getroffen werden. Wie im Abschnitt 2.10.2 beschrieben, benötigen wir für unsere Zwecke Rechenkapazität. Deshalb haben wir uns für Amazon EC2 entschieden.
- *Migrating Licensed Products*[7]: Dieser Schritt betrachtet die genutzten Lizenzen. Eine Auflistung der einzelnen Lizenzen der migrierten Software Komponenten ist in der Tabelle 6.12 zu sehen.

Software	License
GlassFish	CDDL Version 1.0[8]
JDK	Oracle Binary Code License[9]
PostgreSQL	OpenSource[10]
activiti	Apache License[11]

Tabelle 6.12.: Lizenz der Software Komponenten

- *Create a Roadmap and a Plan*[7]: Das Ziel dieses Schrittes ist ein Plan für die Migration der einzelnen Komponenten zu erstellen. Unser Plan ist in der Abbildung 1.3 dargestellt. Die Erstellung eines neuen Planes während dieses Migrationsschrittes ist nicht notwendig.

Fazit: Während dieser Phase der Migration sind keine Fehler aufgetreten.

Zeitaufwand: 10 Minuten.

[7]Die Namen der einzelnen Phasen sind direkt von [22] übernommen.

Proof of Concept Phase

Die zweite Phase der Migrationsmethodik hat die Einarbeitung in der von AWS zur Verfügung gestellten Dienste zum Ziel.

Während dieser Phase haben wir von der Firma NovaTec eine Amazon EC2 Instanz zur Verfügung gestellt bekommen. Die Spezifikation der Instanz in der Tabelle 6.13 dargestellt. Als Vorbereitung für die Migration haben wir noch eine *Security Group*[12] erstellt. Somit wurde

Instance Size	m1.medium
Vendor ID	GenuineIntel
CPU model name	Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz
CPU MHz	1795.672
CPU cache size	20480 KB
Total Memory	3750 MB
Distribution ID	Ubuntu
Distribution Release	12.04.2 LTS
AMI-ID	ami-ddfae2a9

Tabelle 6.13.: Spezifikation der Virtuellen Maschine Amazon EC2

der Zugriff auf die neue Amazon EC2 Instanz ermöglicht. Die eingerichtete *Security Group* beinhaltet Regeln für die folgenden Ports:

- *SSH* Port: 22
- *HTTP** Port: 8080
- *HTTPS** Port: 8443
- *GlassFish Admin Console* Port: 4848
- *PostgreSQL Datenbank Server* Port: 5432

Fazit: Während dieser Phase der Migration sind keine Fehler aufgetreten. Die Dauer dieser Phase basiert auf der Beantragung und der Einrichtung der Amazon EC2 Instanz.

Zeitaufwand: 24 Stunden.

Data Migration Phase

Diese Phase der Migration erfolgt durch die Anwendung der Migrationsmethodik und des Tools von Bachmann. Eine detaillierte Beschreibung dieser Phase ist im Abschnitt 6.3.2 zu finden.

[12] Amazon Security Groups: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html

Fazit: Während dieser Phase der Migration sind keine Fehler aufgetreten.

Zeitaufwand: 10 Minuten.

Application Migration Phase

In [22] sind zwei Strategien zur Migration einer Applikation beschrieben.

- *Forklift Migrationsstrategie*: Diese Migrationsstrategie eignet sich zur Provisionierung von zustandslosen, eigenständigen Anwendungen. Die Migration erfolgt indem die gesamte Applikation in die Amazon EC2 kopiert wird.
- *Hybrid Migrationsstrategie*: Die *Hybrid Migrationsstrategie* benutzt einen iterativen Ansatz und eignet sich zur Provisionierung von einzelnen Applikationskomponenten.

Da wir bereits die DBL von NovaERM in der *Data Migration Phase* provisioniert haben, werden wir die *Hybrid Migrationsstrategie* benutzen. Die Migration der NovaERM Webapplikation erfolgt mit Hilfe des Tools *automaIT*. Dieser Schritt ist im Abschnitt 6.3.1 detailliert beschrieben.

Fazit: Während dieser Phase der Migration sind keine Fehler aufgetreten.

Zeitaufwand: 20 Minuten.

6.3.5. Datenverarbeitung

Im Laufe der durchgeführte Migration der Applikation in die Amazon Cloud sind keine kritische Fehler aufgetreten. Weiterhin ist eine Verarbeitung der gesammelten Informationen während der Migration nicht notwendig.

Eine Zusammenfassung der genutzten Zeit pro Migrationsphase ist in der Abbildung 6.4 zu finden.

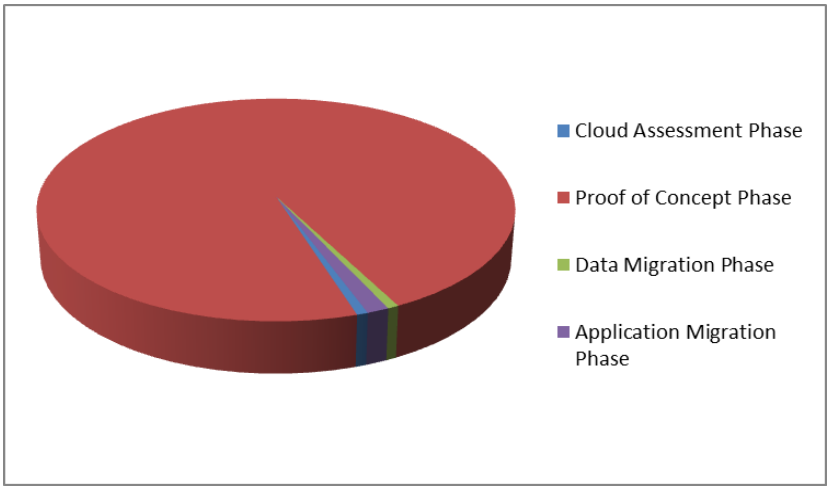

Abbildung 6.4.: Zusammenfassung genutzter Zeit während der zweiten Iteration, Schritt 2

7. Evaluationsergebnisse und Verbesserungsvorschläge

Im Laufe dieses Kapitels werden die gewonnenen Erkenntnisse sowohl in Bezug auf die Methodik als auch des Tools von Bachmann, auf Basis der durchgeführten Evaluation im Kapitel 6, vorgestellt und ausführlich beschrieben. Diese beinhalten sowohl eine Diskussion der erkannten Fehler als auch die ausführliche Darstellung der durchgeführten Verbesserungen. Außerdem werden im Abschnitt 7.2 die aufgetretenen Probleme und die damit verbundenen Herausforderungen repräsentiert.

7.1. Gewonnene Erkenntnisse

Dieser Abschnitt dient der Beantwortung von wichtigen Fragen bezüglich der Evaluation. Insbesondere werden die Fragen diskutiert, die im Abschnitt 4.3 enthalten sind. Außerdem werden die erkannten und implementierten Verbesserungsmöglichkeiten bezüglich der Methodik und des Tools von Bachmann vorgestellt und erläutert.

7.1.1. Cloud Datamigrations-Tool

In diesem Abschnitt wird eine ausführliche Analyse der Fehler und Probleme durchgeführt, die während der Evaluation des Tools erkannt wurden. Dabei werden die Fehler mit einem kritischen Schweregrad zuerst analysiert.

Analyse des Fehlers mit ID MD 5

Der Fehler mit ID MD 5 ist in Tabelle 6.8 beschrieben. Heutzutage spielt das Thema Sicherheit eine sehr wichtige Rolle, vor allem wenn es sich um den Schutz von sensiblen Firmendaten handelt. Dass das Tool von Bachmann keine Passwörter mit Sonderzeichen unterstützt, zwingt den Benutzer einfache Passwörter zu benutzen und ermöglicht die Offenlegung der migrierten Informationen.

Der Grund für die mangelhafte Unterstützung von Passwörter mit Sonderzeichen, liegt an der direkten Übergabe des Passwort als Java String im Tool. Für eine erfolgreiche Ausführung müssen die übergebenen Strings im Tool geparst werden. Dieser Fehler im Tool ist mit einer hohen Priorität zu lösen.

Analyse des Fehlers mit ID MD 6

Der Fehler mit ID MD 6 ist in Tabelle 6.9 beschrieben. In Bezug auf die Gebrauchstauglichkeit des Tools ist es empfehlenswert während der Migration der DBL einen Migrationsstatus anzuzeigen. Die Implementierung dieser Funktion kann auf unterschiedlichem Wege gelöst werden. Eine mögliche Lösung ist die einzelnen Schritte der Migration in separate Fenster aufzuteilen. Somit wird auch der Import- und Exportprozess voneinander getrennt und eine höhere Benutzerfreundlichkeit erreicht. Eine einfachere Lösung wäre das angezeigte Fenster zu sperren und einen Prozessbalken einzublenden, während die Migration im Hintergrund ausgeführt wird.

Analyse des Fehlers mit ID MD 7

Der Fehler mit ID MD 7 ist in Tabelle 6.10 beschrieben. Die Funktion des Buttons *Cancel* im Tool entspricht nicht dem erwarteten Verhalten. Während der Bearbeitung der Migrationsschritte im Tool wird mittels des Buttons *Cancel*, anstatt zurück zu vorherige Seite zu wechseln, direkt zur allgemeinen Projektübersicht des Tool gewechselt. Weiterhin ist das Verhalten dieses Buttons auch im Migrationsprojektübersicht falsch.

Lösung: Im Rahmen dieser Arbeit wurde dieser Fehler korrigiert und das gewünschte Verhalten erreicht.

Analyse des Fehlers mit ID MD 4

Der Fehler mit ID MD 4 ist in Tabelle 6.7 beschrieben. Wie bereits im Abschnitt 2.7 erläutert, werden im fünften Schritt der Migrationsmethodik von Bachmann, Konflikte zwischen dem Quell- und Zieldatenspeicher erkannt und möglichen Lösungen vorgeschlagen.

Wie anhand der Abbildung 7.1 zu erkennen ist, sind während dieser Phase insgesamt drei Konflikte erkannt worden, wobei Konflikt *#1* und *#3* dieselbe Beschreibung besitzen, aber unterschiedlichen Lösungen. Diese Auflistung der erkannten Konflikte, kann in Bezug auf die funktionale Korrektheit und Angemessenheit im Tool zu Missverständnissen führen.

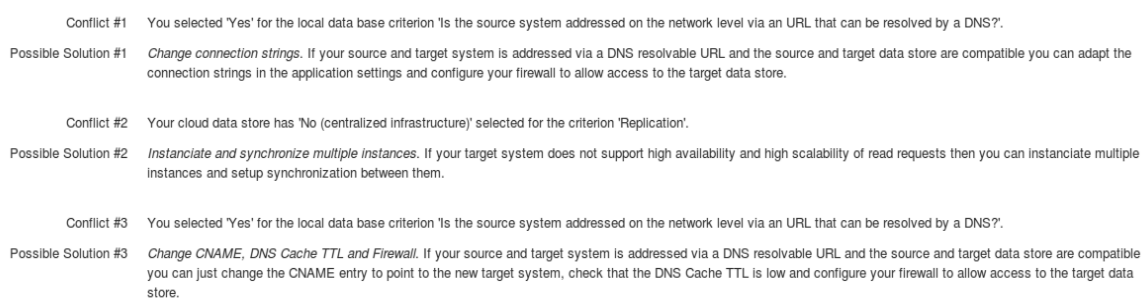

Abbildung 7.1.: Analyse Fehler MD4

Das Zusammenführen von gleichen Konflikten und die Auflistung der möglichen Lösungen als Unterpunkte (1.1, 1.2 usw.) würde die Angemessenheit der vorgeschlagenen Lösungen verbessern.

Analyse des Fehlers mit ID MD 3

Der Fehler mit ID MD 3 ist in Tabelle 6.5 beschrieben. Im Schritt drei der Migration dauert das Laden der vorgeschlagenen Cloud Data Stores sehr lange, da auch zu jedem Store eine ausführliche Beschreibung und Auflistung der unterstützen Eigenschaften angezeigt wird. Um den Ladeprozess zu beschleunigen ist es ratsam in der Darstellung nur die Namen der vorgeschlagenen Stores aufzulisten. Des Weiteren sollte es ermöglicht werden, die unterstützen Eigenschaften der Cloud Data Stores ein- oder auszublenden.

Analyse des Fehlers mit ID MD 2

Der Fehler mit ID MD 2 ist in Tabelle 6.4 beschrieben. In der dritten Phase der Migration sollte direkt ermöglicht werden, falls kein passender Cloud Data Store vorhanden ist, einen anzulegen. Ein Button *Add New Cloud Data Store* im *Select Cloud Data Store* View des Tools würde dies ermöglichen sowie die Möglichkeit eröffnen, den Migrationsprozess ununterbrochen fortzuführen.

Analyse des Fehlers mit ID MD 1

Der Fehler mit ID MD 1 ist in Tabelle 6.3 beschrieben. Das Eingeben von Text in den Textfeldern im Tool ist nur bedingt möglich, weil der Fokus des Felds wiederholt entfernt wird. Eine mögliche Lösung dieses Fehlers ist ein Doppelklick auf das Textfeld. Eine mögliche Ursache dieses Fehlers ist die Implementierung der CSS Klassen im Tool.

Tool Verbesserungsvorschlägen

Um die Benutzerfreundlichkeit und Nutzbarkeit sowie die Effizienz des Tools zu erhöhen sind folgende Verbesserungsvorschläge während der Evaluation entstanden:

- *Fehler/Konflikte hervorheben*: Um eine bessere Erkennbarkeit zu ermöglichen, ist es ratsam im Tool die identifizierten Fehler oder Konflikte während des Migrationsprozess hervorzuheben.
- *Nutzung von nativen JDBC Treiber*: Um die Qualität der angebotenen Funktionen zu verbessern, sowie das Tool betriebssystemunabhängig zu halten, ist es zu empfehlen den Import und Export der Datenbanken mit Hilfe der nativen JDBC Treiber zu ermöglichen. Somit wird die Verteilung des Cloud Datamigrations-Tools von Bachmann und der dazugehörigen MySQL[1] Datenbank auf separaten Maschinen ermöglicht.

[1] MySQL: http://www.mysql.de/

- *Abhängigkeiten zwischen Quell- und Zielsystem Adapter*: Bei der Implementierung der einzelnen Quell- und Zielsystem Adapter, soll es ermöglicht werden, Abhängigkeiten zwischen diesen zu definieren. Damit lässt sich die Auswahl eines Zielsystems nach der Spezifikation des Quellsystems nur auf die Zielsysteme beschränken für die ein kompatibler Quell- und Zieladapter vorhanden ist. Dadurch wird auch die Fehleranfälligkeit bei der Migration eines DBL verringert.
- *Migrationsprojekten/Cloud Data Stores löschen*: Das Tool soll die Funktionalität anbieten, angelegte Migrationsprojekte und Cloud Data Stores zu entfernen. Im Rahmen dieser Arbeit wurde diese Funktionalität implementiert.
- *Auswahl von Eigenschaften rückgängig machen*: Es soll ermöglicht werden die in den Schritten 1, 2 und 4 ausgewählten Eingenschaften, rückgängig zu machen. Diese sind beim Erstellen eines Projektes immer leer. Die Wiederherstellung dieses Zustandes soll ermöglicht werden.
- *Import & Export Funktion*: Zur Steigerung der Benutzerfreundlichkeit sollte sowohl der Import als auch der Export von angelegten Migrationsprojekten und Cloud Data Stores als Funktionalität ermöglicht werden. Diese Funktion wird die Migration von vorhandenen Projekten und Cloud Data Stores zu weiteren Instanzen des Tools erleichtern. Außerdem wird dadurch eine Backup Funktion des Tools ermöglicht.

7.1.2. Methodik von Bachmann

Im Laufe dieses Abschnittes werden die gewonnenen Erkenntnisse bezüglich der Migrationsmethodik von Bachmann (siehe Abschnitt 2.7) vorgestellt. Insbesondere wird die Eignung der Methodik für die Migration einer DBL, sowie ihre Rolle bei der Entscheidungsfindung zur Auswahl eines Cloud Anbieters, näher erläutert. Des Weiteren wird ein Iterativer Ansatz zur Durchführung der Migration vorgestellt. Abschließend wird die Möglichkeit die Methodik von Bachmann mit anderen zu kombinieren, beschrieben.

Analysis

Während des Migrationsprozesses hat die genutzte Migrationsmethodik von Bachmann einen ausführlichen und Datenbank unabhängigen Ansatz für die Durchführung der Migration angeboten. Weiterhin ist die Auswahl von Quell- und Zielsystem für die Migration nicht nur auf cloudbasierten Systeme und Services (off-premise) beschränkt, sondern bietet auch eine Unterstützung für lokalen Datenbanksystemen (on-premise) an.

Im ersten Schritt der Methodik wird eine große Auswahl von Migrationsszenarien angebotenen. Anhand des ausgewählten Szenarios wird der Benutzer während der Durchführung der Migration mit passenden Vorschlägen und Lösungen für möglich auftretenden Probleme unterstützt. Anhand der ausgewählten Kriterien wird ein passender Cloud Anbieter vorgeschlagen. Des Weiteren abhängig von der Auswahl des Migrationsszenarios werden möglichen Anpassungen der Applikation Architektur empfohlen.

Iterativer Ansatz

Wie bereits im Abschnitt 4.3 dargelegt, werden wir in diesem Abschnitt einen iterativen Ansatz für die Durchführung der Methodik repräsentieren. Die einzelnen Migrationsschritte sowie die möglichen Iterationen zwischen diesen sind in der Abbildung 7.2 dargestellt.

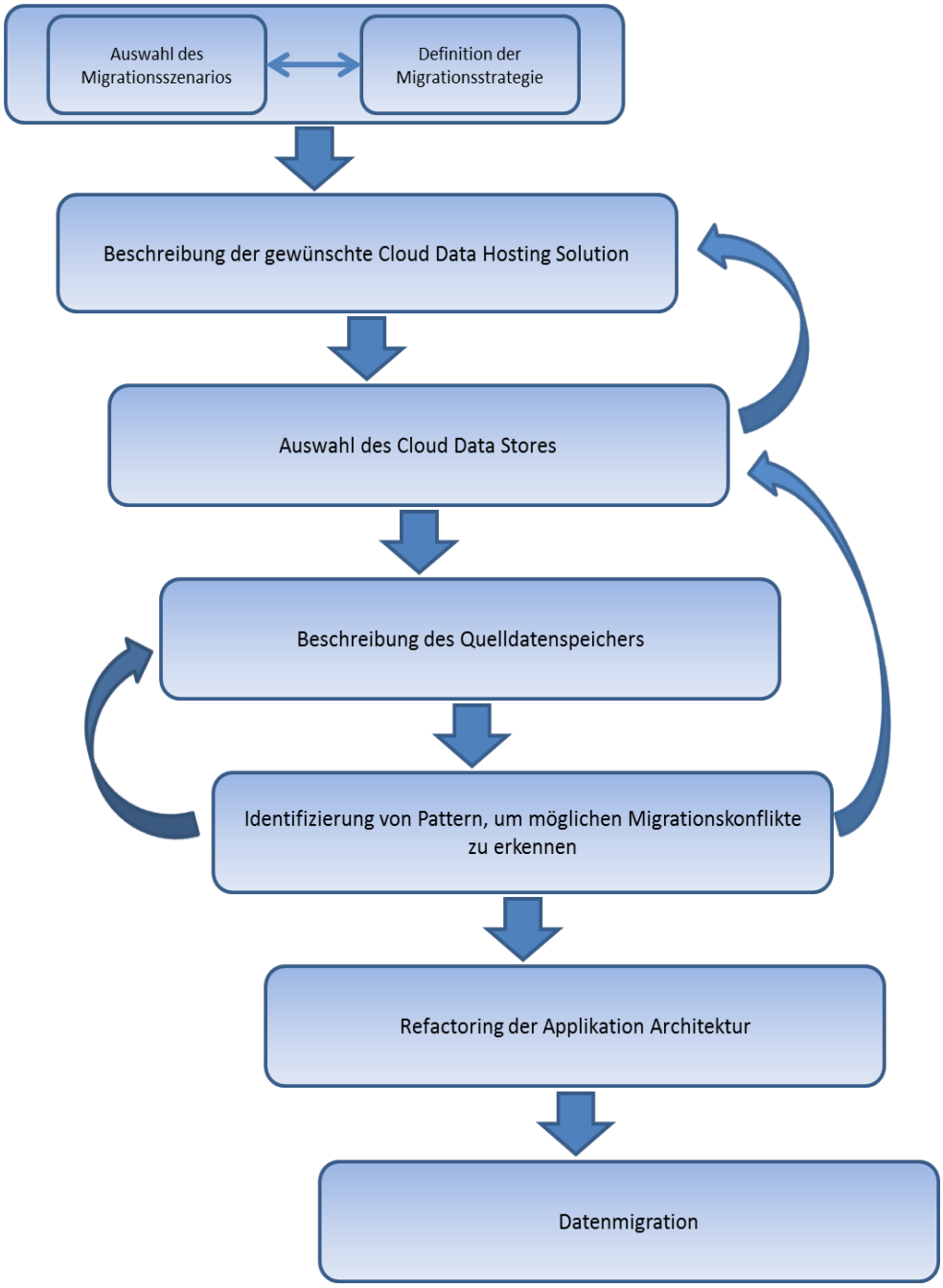

Abbildung 7.2.: Migrationsmethodik Iterativer Ansatz

Wie Anhand der Abbildung 7.2 zu erkennen ist, sind in der Migrationsmethodik vier Iterati-

onsschritte repräsentiert. Alle Schritte basieren auf der angebotenen Funktionalität im Cloud Datamigrations-Tool von Bachmann (siehe Abschnitt 2.8). Der erste Schritt findet in der ersten Phase der Methodik statt und ermöglicht eine Anpassung zwischen dem Migrationsszenario und der definierten Migrationsstrategie. In der dritten Phase der Methodik, falls der p assenden Cloud Data Store nicht als Vorschlag ausgewählt werden kann, wird ermöglicht zur zweiten Phase der Methodik zu wechseln. Diese Iteration ermöglicht die Beschreibung des gewünschten Cloud Data Stores zu ändern oder zu verfeinern, damit die entsprechende Auswahl von Cloud Data Stores in der dritten Phase zur Verfügung steht. Die anderen zwei Iterationsschritte finden in der fünften Phase der Migrationsmethodik statt. Mit Hilfe dieser Schritte wird der Wechsel des Cloud Data Stores oder die Änderung und Verfeinerung der Beschreibung des Quelldatenspeichers ermöglicht. Dadurch können mögliche Konflikte, die in der fünfte Migrationsphase erkannt werden, gelöst werden.

Kombinierbarkeit mit anderen Methodiken

Im Abschnitt 6.3.4 ist die Durchführung einer Gesamtmigration mit Hilfe der phasengetriebenen Migrationsmethodik von AWS in Kombination mit der Methodik und des Tools von Bachmann vorgestellt.

Anhand der durchgeführten Migration könnten wir erkennen, dass sich manche Phasen der beiden Methodik zum Teil überdecken. Andere Phasen der Migrationsmethodik von Bachmann ergänzen die vorgeschlagenen Phasen von AWS. Eine Kombination der beiden Methodiken liefert einen ganzheitlichen Ansatz zur Migration von Anwendungen anhand von verschiedenen Migrationsszenarien.

Im Rahmen dieser Arbeit wird nur eine Abbildung der sich zum Teil überdeckenden Phasen in Abbildung 7.3 repräsentiert. Auf eine Darstellung einer ganzheitlichen Methodik in Kombination von beiden Methodiken wird innerhalb dieser Arbeit verzichtet.

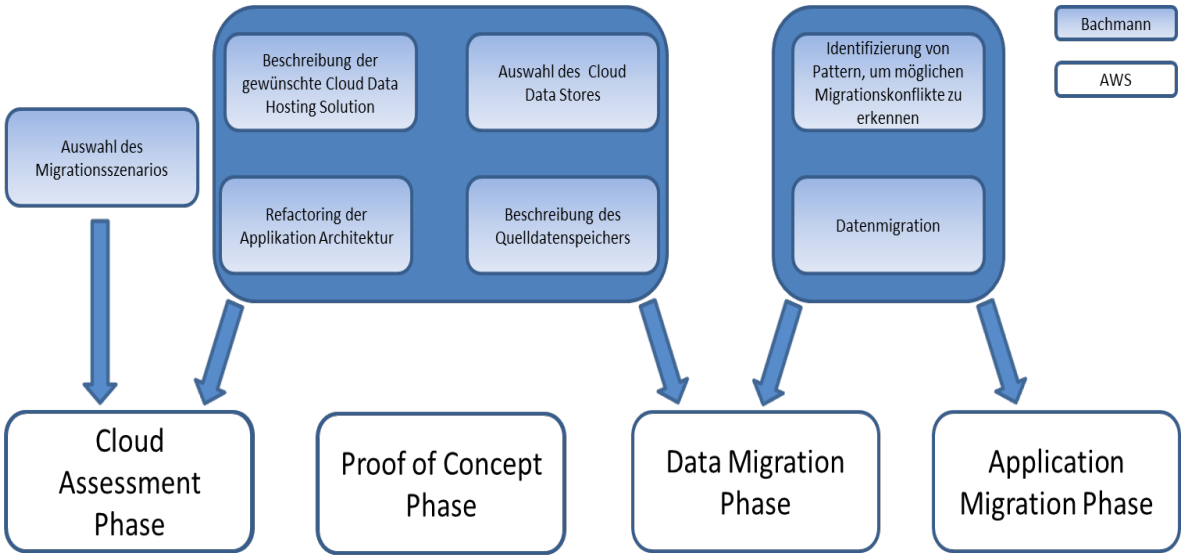

Abbildung 7.3.: Überdeckende Phasen - Bachmann und AWS

Analyse der benötigten Zeit für die Migration

Ein weiterer Aspekt der Evaluation ist die Analyse der benötigten Zeit während der Migration. Im Folgenden werden wir die Zeit für die einzelne Iterationen detailliert betrachten und diese mit der benötigten Zeit für die Baseline vergleichen.

Bei der Vorbereitung der Migration mit Hilfe des Tools *automaIT* (siehe Abschnitt 6.1) wurde für den Import der beiden Datenbanken insgesamt 16 Sekunden benötigt. Diese Zeit ist in der gesamten Installationszeit für die Vorbereitung von 08:58 enthalten. Dagegen dauerte der Import während der Bestimmung der Baseline in der ersten Iteration (siehe Abschnitt 5.1.1) 42 Sekunden. Der Unterschied von 26 Sekunden ist auf die Latenz in der Internet Verbindung zwischen der lokalen Maschine und der NovaTec Cloud zurückzuführen. Weiterhin benötigte das Cloud Datamigrations-Tool von Bachmann für die Migration der beiden Datenbanken in die NovaTec Cloud 3 Minuten, wobei zu beachten ist, dass während dieser Migration die beiden Datenbanken erst exportiert und danach in die Cloud importiert werden. Da der Export lokal stattgefunden hat, ist die dafür benötigte Zeit sehr kurz und kann somit vernachlässigt werden. Der große Unterschied zwischen der Baseline und der Migration mit Hilfe des Tools von Bachmann ist auf die Art der Datenübertragung zwischen der lokalen Maschine und der NovaTec Cloud zurückzuführen. Bei der Bestimmung der Baseline wurde für die Übertragung ein Secure Shell (SSH) Tunel genutzt, dagegen wurde bei der Migration mit dem Tool von Bachmann, das PostgreSQL Import Tool *psql* (siehe Abschnitt 5.3.2) benutzt. Des Weiteren wird bei der Migration mit dem Tool von Bachmann sowohl bei dem Export als auch bei dem Import erst die Verbindung mit dem Quell- und Zieldatenbank System überprüft.

In der zweiten Phase dauerte der Import der beiden Datenbanken bei der Baseline 29 Sekunden. Der Unterschied zur Baseline während der ersten Iteration und diese in der Migrationsvorbereitungsphase, ist auf die Latenz der Internetverbindung zurückzuführen. Die benötigte Zeit für die Migrationsphase mit Hilfe des Tools von Bachmann beträgt 3 Minuten. Somit unterscheidet sich diese nicht von Zeit für diese Phase in der ersten Iteration. Des Weiteren ist der Unterschied zur Baseline wiederum auf die Latenz in der Internetverbindung sowie auf die benutzte Übertragungsmethodik zurückzuführen.

Anhand der Durchgeführten Migrationen ist zu erkennen, dass der Import mit Hilfe des Tools *automaIT* im Vergleich der durchgeführten Migration mit Hilfe der Migrationsmethodik und des Tools von Bachmann, deutlich schneller erfolgt. Dagegen bietet *automaIT* weder Entscheidungsunterstützung bei der Auswahl eines Cloud Anbieters noch Konflikterkennung zwischen den ausgewählten Migrationseigenschaften. Diese Vorteile sowie die Unterstützung von unterschiedlichen Migrationsszenarien bieten den Benutzern eine große Hilfe bei der Migration der Datenbankschicht in die Cloud.

7.1.3. Erweiterbarkeit der Methodik und des Tools von Bachmann

Wie anhand der Durchgeführten Testphasen während unserer Evaluation zu erkennen ist, spielt die Testphase in der Migration eine sehr wichtige Rolle. Das Ziel dieser Phase ist nicht

nur den Erfolg der Durchgeführte Migration zu überprüfen sondern auch Optimierungsmöglichkeiten zu erkennen. Des Weiteren ist die Dauer dieser Phase im Vergleich zu den anderen Migrationsphasen, am längsten. Deswegen ist die Einbeziehung sowohl von der Testphase als auch von der Optimierungsphase von [40] in die Migrationsmethodik und das Tool von Bachmann (siehe Abschnitt 2.7) sehr wichtig.

7.2. Probleme und Herausforderungen

Für die erfolgreiche Durchführung der Migrationsszenarien und die damit verbundene Evaluation sowie für die Performance Messung, mussten wir uns vielen Herausforderungen stellen. Deswegen werden wir diese im Folgenden näher erläutern.

Bei der Beschreibung des Cloud Datamigrations-Tools haben wir bereits angedeutet, dass das Tool nur MySQL und Virtuelle Maschine als Quellsystem Typen unterstützt (siehe Abschnitt 2.8). Da die NovaERM Webapplikation ihre Daten in einer PostgreSQL Datenbank speichert, mussten wir für die Durchführung der Migration einen Source und Target Adapter implementieren und somit die Funktionalität des Tools erweitern. Die Einarbeitung in die Architektur des Tools sowie die Implementierung der beiden Adapter hat insgesamt vier Tage in Anspruch genommen. Die Implementierungsdetails sind im Abschnitt 5.3 beschrieben.

Eine weitere große Herausforderung während unserer Arbeit stellte die Bewerkstelligung der Performance Messung mit Hilfe der Tools *inspectIT* und *seleniumHQ* dar. Da sich NovaERM immer noch in der Entwicklungsphase befindet, werden die UI Test von *seleniumHQ* stetig an die grafische Oberfläche der Webapplikation angepasst. Dieser Vorgang führt oft zu Inkonsistenzen zwischen den UI Tests und der Oberfläche von NovaERM. Dies erforderte eine lange Testphase des Ausprobierens, um eine Version von NovaERM zu finden, die mit den *seleniumHQ* UI Test kompatibel war. Die Version, die wir für unsere Evaluation und somit für die Perfomance Messung benutzt haben, wurde am 12 August gebildet. Des Weiteren haben wir festgestellt, dass die Klassen des *inspectIT Agent*s nicht im Classloader des GlassFish Web Containers geladen werden. Um dieses Problem zu lösen, sollen die benötigten Klassen des Agents in "org.osgi.framework.bootdelegation" des Web Containers beschrieben werden. Die genutzte Konfiguration der "org.osgi.framework.bootdelegation" ist in Listing 7.1 zu sehen.

```
1 inspectit.bootdelegation= info.novatec.inspectit.agent, info.novatec.inspectit.agent.*
2
3 org.osgi.framework.bootdelegation=${eclipselink.bootdelegation}, ${inspectit.bootdelegation
      }, \
4                                   com.sun.btrace, com.sun.btrace.*, \
5                                   org.netbeans.lib.profiler, org.netbeans.lib.profiler.*
```

Listing 7.1: Konfiguration des OSGi Frameworks

Wie im Abschnitt 2.5 beschrieben, besteht *inspectIT* aus drei Komponenten. Aus Performance Gründen und um die Messergebnisse nicht zu beeinflussen, wurden diese auf unterschiedli-

chen Maschinen aufgesetzt (siehe Abshnitt 5.2). Dies hatte zu Folge, dass keine Verbindung zwischen dem *inspectIT Agent* und dem *inspectIT* CMR hergestellt werden konnte. Eine Lösung dieses Problems ist in der Dokumentationsseite von *inspectIT*[2] beschrieben. Im Laufe unserer Test konnten wir feststellen, dass das *inspectIT* CMR primär das Local Area Network (LAN) Interface erkennt und wenn dieses eingeschaltet ist, sich direkt mit diesem verbindet. Bei der Durchführung der UI Tests haben wir nur das LAN Interface genutzt.

Des Weiteren musste noch die *inspectIT Agent* Konfigurationsdatei (siehe Anhang D.1) angepasst werden, um die Funktionalität der Sensoren, beschrieben im Abschnitt 2.5, zu nutzen. Das größte Problem stellten die Enterprise JavaBeans (EJB) Komponenten Tests dar, da diese nicht die von *inspectIT* erwarteten Bedingungen erfüllt haben. In mehreren EJB Komponenten sind Java Methoden vorhanden, die als einen Rückgabewert *null* zurückgeben, anstatt den Typ, der in der Methodendefinion definiert ist. Diese Inkompatibilität verhinderte das Deployen der NovaERM Webapplikation in dem GlassFish Web Container. Aufgrund des begrenzten Zeitrahmens, der uns während unserer Arbeit zur Verfügung steht, sind diese Test nicht in unserer Performance Messung enthalten.

7.3. JDBC Connection Pools

Für die Durchführung der einzelnen Migrationsszenarien beschrieben in Abschnitten 4.5.1 und 4.5.3 muss die Verbindung zwischen den PostgreSQL Datenbanken und dem GlassFish Web Container und somit auch der NovaERM Webapplikation stets angepasst werden. Die Anpassung erfolgt in der Domain Konfiguration des Web Containers und wird mit Hilfe des Tools *automaIT* ermöglicht. Die zugehörigen *Connection Pools*[3] sind in Listing 7.2 zu finden. Wie anhand der beschriebenen *Connection Pools* zu erkennen ist, ermöglichen diese die Einstellung der Verbindungsparameter wie z.B. Username, Passwort und Port sowie Datenbankname und die dazugehörige Adresse des Datenbank Servers.

[2] *inspectIT* Dokumentation: https://documentation.novatec-gmbh.de/pages/viewpage.action?pageId=7733354
[3] GlassFish Connection Pools: http://docs.oracle.com/cd/E18930_01/html/821-2417/giyil.html

```
<jdbc-connection-pool connection-validation-method="meta-data" datasource-classname="org.
    postgresql.xa.PGXADataSource" max-pool-size="50" res-type="javax.sql.XADataSource"
    steady-pool-size="5" description="Connection Pool for Activiti" name="activitiPool" is-
    connection-validation-required="true">
  <property name="user" value="postgres"></property>
  <property name="password" value="mondschein"></property>
  <property name="portNumber" value="5432"></property>
  <property name="databaseName" value="activiti"></property>
  <property name="serverName" value="localhost"></property>
</jdbc-connection-pool>
<jdbc-resource pool-name="activitiPool" jndi-name="jdbc/activiti"></jdbc-resource>
<jdbc-connection-pool connection-validation-method="meta-data" datasource-classname="org.
    postgresql.xa.PGXADataSource" max-pool-size="50" res-type="javax.sql.XADataSource"
    steady-pool-size="5" description="Connection Pool for NovaERM" name="novaermPool" is-
    connection-validation-required="true">
  <property name="user" value="postgres"></property>
  <property name="password" value="mondschein"></property>
  <property name="portNumber" value="5432"></property>
  <property name="databaseName" value="novaerm"></property>
  <property name="serverName" value="localhost"></property>
</jdbc-connection-pool>
<jdbc-resource pool-name="novaermPool" jndi-name="jdbc/novaerm"></jdbc-resource>
```

Listing 7.2: NovaERM Connection Pools

8. Zusammenfassung und Ausblick

Dieses Kapitel stellt eine Zusammenfassung der Ergebnisse dieser Diplomarbeit dar und gibt einen Ausblick für zukünftige Erweiterungen und Verbesserungen der evaluierten Migrationsmethodik und des Tools.

Im Rahmen dieser Diplomarbeit wurden eine Methodik und ein Tool für die Migration der DBL in die Cloud evaluiert. Die Evaluation basierte auf einer Industriefallstudie, die von der Firma NovaTec zur Verfügung gestellt wurde. Das Ziel dabei war, mit Hilfe der gewonnenen Erkenntnissen während der Evaluation, die Migrationsmethodik und das Tool kontinuierlich zu erweitern und zu verbessern.

Im zweiten Kapitel dieser Arbeit wurden die Grundlagen in Bezug auf das Thema Cloud Computing und Applikationsarchitektur vorgestellt. Des Weiteren wurden sowohl alle Werkzeuge, die im Rahmen dieser Arbeit zum Einsatz kommen als auch die genutzte Methodik und das Tool für die Migration detailliert beschrieben. Ein Überblick über bereits existierende Ansätze für die Durchführung von Evaluationen von Prozessen und Tools wurde im dritten Kapitel vorgestellt. Insbesondere wurden der ITIL CSI Prozess sowie die ISO Norm 25010 hervorgehoben. Anschließend wurde unser Beitrag in Bezug auf die existierenden Arbeiten positioniert.

Im Kapitel 4 wurde die Vorgehensweise bei der Durchführung der Migration erläutert und die Ziele während der Evaluation definiert. Danach wurden die Szenarien vorgestellt, die während der Migration durchgeführt werden.

Die Implementierungsaspekte dieser Arbeit wurden im Kapitel 5 vorgestellt. Der Fokus dabei liegt auf dem implementierten *automaIT* Plugin für die Provisionierung von NovaERM, die Durchführung der Performance Messung sowie auf den erstellten Source und Target Adaptern für das Cloud Datamigrations Tool von Bachmann.

Die Durchführung der Migration mit Hilfe der im Kapitel 4 beschriebenen Migrationsszenarien und die dabei erhobenen Daten sind im Kapitel 6 beschrieben. Um die Datenverarbeitung und Analyse zu vereinfachen, wurden alle aufgetretenen Fehler mit Hilfe einer Vorlage dokumentiert. Im siebten Kapitel wurden die gewonnenen Erkenntnisse in Bezug auf die Methodik und das Tool von Bachmann vorgestellt und ausführlich beschrieben. Diese beinhalten eine Diskussion der erkannten Fehler und die implementierten und vorgeschlagenen Verbesserungen. Am Ende des Kapitels wurden noch die aufgetretenen Probleme und die damit verbundenen Herausforderungen dokumentiert.

Weiterhin wurde im Rahmen dieser Arbeit ein Fragenkatalog erstellt, mit dessen Hilfe relevante Fragen bezüglich der Migration einer Anwendung beantwortet werden können.

Dieser hilft bei der Auswahl eines passenden Cloud Anbieters sowie bei der Anpassung der Anwendungsarchitektur.

Um die Benutzerfreundlichkeit und die Angemessenheit des evaluierten Tools zu erhöhen, wurde ein Teil der aufgetretenen Fehler korrigiert. Außerdem wurden neue Funktionen implementiert, die die Funktionalität des Tools erweitern. Weitere Empfehlungen in Bezug auf mögliche neue Funktionen und Erweiterungen des Tools wurden innerhalb des siebten Kapitels gemacht. Des Weiteren wurde ein iterativer Ansatz für die Durchführung der Migration mit Hilfe der Methodik von Bachmann vorgestellt. Dieser strebt die Erhöhung der Qualität des Migrationsprozesses an und die Vermeidung von Fehlern in den Endphasen der Methodik, die in den früheren Phasen entstanden sind. Es ist vorstellbar, dass dieser Ansatz als Teil einer zukünftigen Arbeit in das Cloud Datamigrations Tool eingebaut und dessen Nutzung evaluiert wird.

Die Durchführung von weiteren Evaluationen auf Basis der Methodik und des Tools mit Einbeziehung anderer Migrationsszenarien sowie Quell- und Zielsystemen ist ebenfalls denkbar. Dabei soll die Eignung der Methodik und des Tools weiter evaluiert und verbessert werden.

Eine zusätzliche Möglichkeit der Erweiterung der Funktionalität des Tools ist die Implementierung weiterer Source und Target Adapter und die Definition von Abhängigkeitsbeziehungen zwischen diesen. Die Einbeziehung der Test- und Optimierungsphase in die Funktionalität des Tools und somit die Erweiterung der bestehende Methodik, um weiteren Phasen als Teil eine zukünftige Arbeit, ist denkbar.

Es ist auch vorstellbar, das implementierte *automaIT* Plugin um die Funktionalität für die Migration von Anwendungen und Datenbanken zu erweitern.

Anhang A.

Fragenkatalog Iteration 1

Migration

Migration Types

- ☑ Type I: Replacement of components
- ☐ Type II: Partial migration
- ☐ Type III: Migration of the whole software stack
- ☐ Type IV: Cloudify the application

Level of migration

- ☑ completely migration
- ☐ partially migration (migration of individual layouts - DB layout, application layout, business layout)

What is the replacement strategy of the migration?

- ☐ Live (without Downtime)
- ☑ Non-Live (with Downtime)

When migrating the database layer - Which is the type of source data store?

- ☑ RDBMS
- ☐ NoSQL
- ☐ BLOB

When migrating the database layer - Which is the type of target data store?

- ☑ RDBMS
- ☐ NoSQL
- ☐ BLOB

What is the direction of the migration

- ☑ Local -> Cloud
- ☐ Cloud -> Cloud
- ☐ Cloud -> Local

Durability of the migration?

- ☐ Off-loading of peak loads
- ☑ Permanent

Cloud Deployment Model

- ☑ Private Cloud
- ☐ Public Cloud
- ☐ Community Cloud
- ☐ Hybrid Cloud

Cloud Service Model

- ☑ IaaS
- ☐ PaaS
- ☐ SaaS

Are open formats supported for the migration (XML, JSON)?

- ☐ Yes
- ☑ No

Can data easily be exported?

- ☑ Yes
- ☐ No

Does the provider offer enough expertise regarding a migration?

- ☐ Yes
- ☑ No

Can the required platform be selected by the user?

- ☑ Yes
- ☐ No

Cloud Interfaces

- ☑ Does the provider offer standardized interfaces?
- ☐ Are the interfaces based on open standards?
- ☐ Is the provided access to the API interface for free?

Compatibility and Integration

Is the solution offered by the provider compatible to other providers?

- ☑ Yes
- ☐ No

Is the own infrastructure compatible with the infrastructure of the Cloud provider?

- ☑ Yes
- ☐ No

Can the architecture of the application as is be deployed on the infrastructure of the provider?

- ☑ Yes
- ☐ No

Are adaptations of the application architecture required?

- ☐ Yes, the architecture of the application should be classified as many small components
- ☑ No, the application should be migrated as a whole in the Cloud

☐ In case the architecture of the application is separated into several components. What is the relation between the components if one or more components might fail.

- ☐ The dependencies between the components should be kept as small as possible. (Loose coupling)

Which operating systems or software are provided by the Cloud providers?

CentOS 6.4

Is the Cloud provider responsible for deployment and provisioning of the OS and additional software?

- ☑ Yes
- ☐ No

What are the requirements of the processor architecture for migration of the application?

- ☐ Single-Core Prozessor
- ☐ Multi-Core Prozessor
- ☑ No Preference

Which requirements has the application with respect to disk space?

- ☑ 20 GB
- ☐ 165 GB
- ☐ 340 GB
- ☐ 850 GB
- ☐ 1890 GB

Which requirements has the application with respect to memory usage?

- ☐ 768 MB
- ☐ 1.75 GB
- ☑ 3.5 GB
- ☐ 7 GB
- ☐ 14 GB

Which requirements has the application with respect to allocated bandwidth? (Allocated Bandwidth Mbps)

- ☑ 5
- ☐ 100
- ☐ 200
- ☐ 400
- ☐ 800

Runtime environment

Which programming languages are supported by the Cloud provider?

- ☐.NET
- ☐ PHP
- ☐ Ruby
- ☐ Python
- ☐ Java

Which data formats are supported by the Cloud provider?

- ☐ XML
- ☐ JSON
- ☐ CSV

Which protocols / messaging architecture are supported by the Cloud provider?

- ☑ HTTP
- ☐ SOAP
- ☑ FTP

Which formats for importing of VM are supported by the Cloud provider?

- ☐ VMware ESX VMDK
- ☐ Citrix Xen VHD
- ☐ Microsoft Hyper-V VHD-Images für Microsoft Windows Server 2003 (R2) und 2008 (R1 und R2)

Which formats for exporting of VM are supported by the Cloud provider?

- ☐ VMware ESX VMDK
- ☐ VMware ESX OVA
- ☐ Microsoft Hyper-V VHD
- ☐ Citrix Xen VHD

Load balancing

Does the cloud provider offer "load balancing", e.g., Amazon Elastic Load Balancing and Windows Azure Traffic Manager?

- ☐ Yes
- ☑ No

Which protocols are supported for “Load Balancing”?

- ☐ HTTP
- ☐ HTTPS
- ☐ TCP
- ☐ SSL
- ☐ custom

When does the Cloud provider offer the option for "load balancing"?

- ☐ At least two instances have to be running
- ☐ On extra charge

Does the Cloud provider support error checks for the "load balancer"?

- ☐ Detection of instances having errors

Which methods are supported from the load balancer?

- ☐ Round Robin
- ☐ according performance / according application traffic and load
- ☐ Random

Is a manual selection of a method for the load balancer possible?

- ☐ Yes, it is possible to choose between the methods
- ☐ No, it is not possible to choose between the methods

Monitoring

Can be resources, applications and processes monitored during runtime (e.g., Amazon Cloud Watch, Microsoft Azure Watch)?

- ☑ Yes
- ☐ No

Does the monitoring system offer an user friendly interfaces?

- ☐ Create charts, set alarms, perform automated actions, export of data, generation of reports

Are notifications of administrators are supported (e.g., Amazon Simple Notification Service)

- ☐ Yes
- ☑ No

Can a remote connection be established with the Cloud infrastructure? Which tools are available on the side of the Cloud provider?

- ☐ Web interface
- ☐ Management Console
- ☐ Web service
- ☑ Command Line Interface
- ☑ Remote Desktop Connection

Does the cloud provider offer a detailed documentation for the use of it's infrastructure?

- ☐ Yes
- ☑ No

Concurrent Access

Multi-Tenancy

Is the application multi-tenancy aware, which I like to migrate?

- ☐ Yes, it is multi-tenancy aware
- ☐ No

If no, do I need multi-tenancy and with what effort can I refactor my application?

- ☐ Yes, with a major effort
- ☐ Yes, with a little effort

Does the Cloud provider support multi-tenancy?

- ☐ Yes, Single-Tenancy
- ☐ Yes, Multi-Tenancy
- ☑ No

If yes, in what context is the multi-tenancy offered?

- ☐ Infrastructure (multiple OS instances on the same hardware can be started using hypervisor technologies)
- ☐ Platform (Isolation of code from different applications on the same OS instance)
- ☐ Application (single instance components are shared between several clients)

What advantages and disadvantages arise for me as a customer, if I decide for single or multi-tenancy?

- ☐ Can I set my own settings without limitation?
- ☐ To which degree is my application isolated from other customers of the cloud provider, e.g. data isolation and performance isolation?
- ☐ Do I have to install and configure my application in the Cloud by myself?
- ☐ Is external support and consulting required for the maintenance?
- ☐ Is running the application in Single-tenancy mode more expensive than running it in multi-tenant aware?

Scalability

Scalability Type

- ☐ vertical
- ☐ horizontal

What is the criteria for scaling?

- ☐ system load
- ☐ latency

During an overload of my application or instance are resources automatically adjusted to the needs (e.g., Amazon Auto Scaling, Microsoft Azure - WASABi)?

- ☐ Yes
- ☐ No
- ☐ partially automated

If yes, can I define my own scalability rules?

- ☐ Yes
- ☐ No

Are the scalability rules stored in a "human readable" form? (e.g., XML)

- ☐ Yes
- ☐ No

Can I change or adjust the scalability rules on the fly, without any impact?

- ☐ Yes
- ☐ No

If No, which impact and faults are expected?

Can the resources be scaled manually, so you can adapt it to your own needs?

- ☐ Yes
- ☐ No

In case of horizontal scalability, how fast can be a new instance started (Resource Provisioning Delay)?

- ☐ min. 10 Min.
- ☐ max. 30 Min.

What are the limits for scalability set by the Cloud provider?

- ☐ Horizontal scalability - has the user an unlimited number of instances?
- ☐ Vertical scalability - to which degree performance can be increased?

Can the scaling be defined based on a time schedule?

- ☐ Yes, in certain time intervals instances can be started or shut down.
- ☐ Yes, the Cloud user himself can define time intervals, in which instances are started or shut down.
- ☐ No

Can the parameters for scaling be defined by the users themselves?

- ☐ Yes, at vertical scaling e.g. number of CPU cores, memory size, disk space, allocated bandwidth
- ☐ Yes, at horizontal scaling e.g. when a certain load is reached
- ☐ No

Transaction and Consistency

Transaction

For a Storage - is the ACID (Atomicity, Consistency, Isolation, Durability) property guaranteed**?**

- ☑ Yes
- ☐ No

For a Storage - is the BASE (Basically Available, Soft state, Eventual consistency) property guaranteed?

- ☑ Yes
- ☐ No

Consistency

What type of consistency is provided by the Cloud provider?

- ☐ Strong (Windows Azure Storage)
- ☐ Weak
- ☐ Eventual Consistency (e.g., Amazon SimpleDB)

Is there a time interval set by the Cloud provider, after all instances must be consistent?

- ☐ Yes
- ☐ No

Can be assured that during Read operations always the "latest version" is returned (Consistent read vs. eventually consistent read)?

- ☑ Yes
- ☐ No

Responsibilities

Which tasks the Cloud user is responsible for when using the Cloud service?

- ☐ System Updates
- ☐ Security Patches
- ☐ Running application
- ☐ Data
- ☑ Access restrictions
- ☐ Compliance with legal requirements

Which tasks the Cloud provider is responsible for?

Which model of responsibilities is offered by the cloud provider?

- ☐ shared responsibility - a responsibility between the customer and the Cloud provider

Does the Cloud provider offer a Service Level Agreement (SLA)?

- ☐ Yes
- ☑ No

Are the services offered and ensured in the SLAs sufficient?

- ☐ Yes
- ☐ No

Does the Cloud providers specify availability guarantee with respect to the Cloud services offered?

- ☐ Yes
- ☑ No

Which liability does the Cloud provider take for costs and damage caused by technical failure?

Does the Cloud provider inform his customers about scheduled maintenance periods?

- ☑ Yes
- ☐ No

Does the provider perform a regular, independent audit of its IT security status?

- ☑ Yes
- ☐ No

Anhang B.

Fragenkatalog Iteration 2

Migration

Migration Types

- ☑ Type I: Replacement of components
- ☐ Type II: Partial migration
- ☐ Type III: Migration of the whole software stack
- ☐ Type IV: Cloudify the application

Level of migration

- ☑ completely migration
- ☐ partially migration (migration of individual layouts - DB layout, application layout, business layout)

What is the replacement strategy of the migration?

- ☐ Live (without Downtime)
- ☑ Non-Live (with Downtime)

When migrating the database layer - Which is the type of source data store?

- ☑ RDBMS
- ☐ NoSQL
- ☐ BLOB

When migrating the database layer - Which is the type of target data store?

- ☑ RDBMS
- ☐ NoSQL
- ☐ BLOB

What is the direction of the migration

- ☑ Local -> Cloud
- ☐ Cloud -> Cloud
- ☐ Cloud -> Local

Durability of the migration?

- ☐ Off-loading of peak loads
- ☑ Permanent

Cloud Deployment Model

- ☐ Private Cloud
- ☑ Public Cloud
- ☐ Community Cloud
- ☐ Hybrid Cloud

Cloud Service Model

- ☑ IaaS
- ☐ PaaS
- ☐ SaaS

Are open formats supported for the migration (XML, JSON)?

- ☐ Yes
- ☑ No

Can data easily be exported?

- ☑ Yes
- ☐ No

Does the provider offer enough expertise regarding a migration?

- ☑ Yes
- ☐ No

Can the required platform be selected by the user?

- ☑ Yes
- ☐ No

Cloud Interfaces

- ☑ Does the provider offer standardized interfaces?
- ☐ Are the interfaces based on open standards?
- ☐ Is the provided access to the API interface for free?

Compatibility and Integration

Is the solution offered by the provider compatible to other providers?

- ☑ Yes
- ☐ No

Is the own infrastructure compatible with the infrastructure of the Cloud provider?

- ☑ Yes
- ☐ No

Can the architecture of the application as is be deployed on the infrastructure of the provider?

- ☑ Yes
- ☐ No

Are adaptations of the application architecture required?

- ☐ Yes, the architecture of the application should be classified as many small components
- ☑ No, the application should be migrated as a whole in the Cloud

☐ In case the architecture of the application is separated into several components. What is the relation between the components if one or more components might fail.

- ☐ The dependencies between the components should be kept as small as possible. (Loose coupling)

Which operating systems or software are provided by the Cloud providers?

Ubuntu 12.04 LTS

Is the Cloud provider responsible for deployment and provisioning of the OS and additional software?

- ☑ Yes
- ☐ No

What are the requirements of the processor architecture for migration of the application?

- ☐ Single-Core Prozessor
- ☐ Multi-Core Prozessor
- ☑ No Preference

Which requirements has the application with respect to disk space?

- ☑ 20 GB
- ☐ 165 GB
- ☐ 340 GB
- ☐ 850 GB
- ☐ 1890 GB

Which requirements has the application with respect to memory usage?

- ☐ 768 MB
- ☐ 1.75 GB
- ☑ 3.5 GB
- ☐ 7 GB
- ☐ 14 GB

Which requirements has the application with respect to allocated bandwidth? (Allocated Bandwidth Mbps)

- ☑ 5
- ☐ 100
- ☐ 200
- ☐ 400
- ☐ 800

Runtime environment

Which programming languages are supported by the Cloud provider?

- ☐.NET
- ☐ PHP
- ☐ Ruby
- ☐ Python
- ☐ Java

Which data formats are supported by the Cloud provider?

- ☐ XML
- ☐ JSON
- ☐ CSV

Which protocols / messaging architecture are supported by the Cloud provider?

- ☑ HTTP
- ☐ SOAP
- ☑ FTP

Which formats for importing of VM are supported by the Cloud provider?

- ☐ VMware ESX VMDK
- ☐ Citrix Xen VHD
- ☐ Microsoft Hyper-V VHD-Images für Microsoft Windows Server 2003 (R2) und 2008 (R1 und R2)

Which formats for exporting of VM are supported by the Cloud provider?

- ☐ VMware ESX VMDK
- ☐ VMware ESX OVA
- ☐ Microsoft Hyper-V VHD
- ☐ Citrix Xen VHD

Load balancing

Does the cloud provider offer "load balancing", e.g., Amazon Elastic Load Balancing and Windows Azure Traffic Manager?

- ☑ Yes
- ☐ No

Which protocols are supported for “Load Balancing”?

- ☐ HTTP
- ☐ HTTPS
- ☐ TCP
- ☐ SSL
- ☐ custom

When does the Cloud provider offer the option for "load balancing"?

- ☐ At least two instances have to be running
- ☐ On extra charge

Does the Cloud provider support error checks for the "load balancer"?

- ☐ Detection of instances having errors

Which methods are supported from the load balancer?

- ☐ Round Robin
- ☐ according performance / according application traffic and load
- ☐ Random

Is a manual selection of a method for the load balancer possible?

- ☐ Yes, it is possible to choose between the methods
- ☐ No, it is not possible to choose between the methods

Monitoring

Can be resources, applications and processes monitored during runtime (e.g., Amazon Cloud Watch, Microsoft Azure Watch)?

- ☑ Yes
- ☐ No

Does the monitoring system offer an user friendly interfaces?

- ☑ Create charts, set alarms, perform automated actions, export of data, generation of reports

Are notifications of administrators are supported (e.g., Amazon Simple Notification Service)

- ☑ Yes
- ☐ No

Can a remote connection be established with the Cloud infrastructure? Which tools are available on the side of the Cloud provider?

- ☑ Web interface
- ☑ Management Console
- ☐ Web service
- ☑ Command Line Interface
- ☑ Remote Desktop Connection

Does the cloud provider offer a detailed documentation for the use of it's infrastructure?

- ☑ Yes
- ☐ No

Concurrent Access

Multi-Tenancy

Is the application multi-tenancy aware, which I like to migrate?

- ☐ Yes, it is multi-tenancy aware
- ☐ No

If no, do I need multi-tenancy and with what effort can I refactor my application?

- ☐ Yes, with a major effort
- ☐ Yes, with a little effort

Does the Cloud provider support multi-tenancy?

- ☐ Yes, Single-Tenancy
- ☐ Yes, Multi-Tenancy
- ☐ No

If yes, in what context is the multi-tenancy offered?

- ☐ Infrastructure (multiple OS instances on the same hardware can be started using hypervisor technologies)
- ☐ Platform (Isolation of code from different applications on the same OS instance)
- ☐ Application (single instance components are shared between several clients)

What advantages and disadvantages arise for me as a customer, if I decide for single or multi-tenancy?

- ☐ Can I set my own settings without limitation?
- ☐ To which degree is my application isolated from other customers of the cloud provider, e.g. data isolation and performance isolation?
- ☐ Do I have to install and configure my application in the Cloud by myself?
- ☐ Is external support and consulting required for the maintenance?
- ☐ Is running the application in Single-tenancy mode more expensive than running it in multi-tenant aware?

Scalability

Scalability Type

- ☐ vertical
- ☐ horizontal

What is the criteria for scaling?

- ☐ system load
- ☐ latency

During an overload of my application or instance are resources automatically adjusted to the needs (e.g., Amazon Auto Scaling, Microsoft Azure - WASABi)?

- ☐ Yes
- ☐ No
- ☐ partially automated

If yes, can I define my own scalability rules?

- ☐ Yes
- ☐ No

Are the scalability rules stored in a "human readable" form? (e.g., XML)

- ☐ Yes
- ☐ No

Can I change or adjust the scalability rules on the fly, without any impact?

- ☐ Yes
- ☐ No

If No, which impact and faults are expected?

Can the resources be scaled manually, so you can adapt it to your own needs?

- ☐ Yes
- ☐ No

In case of horizontal scalability, how fast can be a new instance started (Resource Provisioning Delay)?

- ☐ min. 10 Min.
- ☐ max. 30 Min.

What are the limits for scalability set by the Cloud provider?

- ☐ Horizontal scalability - has the user an unlimited number of instances?
- ☐ Vertical scalability - to which degree performance can be increased?

Can the scaling be defined based on a time schedule?

- ☐ Yes, in certain time intervals instances can be started or shut down.
- ☐ Yes, the Cloud user himself can define time intervals, in which instances are started or shut down.
- ☐ No

Can the parameters for scaling be defined by the users themselves?

- ☐ Yes, at vertical scaling e.g. number of CPU cores, memory size, disk space, allocated bandwidth
- ☐ Yes, at horizontal scaling e.g. when a certain load is reached
- ☐ No

Transaction and Consistency

Transaction

For a Storage - is the ACID (Atomicity, Consistency, Isolation, Durability) property guaranteed**?**

- ☑ Yes
- ☐ No

For a Storage - is the BASE (Basically Available, Soft state, Eventual consistency) property guaranteed?

- ☑ Yes
- ☐ No

Consistency

What type of consistency is provided by the Cloud provider?

- ☐ Strong (Windows Azure Storage)
- ☐ Weak
- ☐ Eventual Consistency (e.g., Amazon SimpleDB)

Is there a time interval set by the Cloud provider, after all instances must be consistent?

- ☐ Yes
- ☐ No

Can be assured that during Read operations always the "latest version" is returned (Consistent read vs. eventually consistent read)?

- ☑ Yes
- ☐ No

Responsibilities

Which tasks the Cloud user is responsible for when using the Cloud service?

- ☐ System Updates
- ☐ Security Patches
- ☐ Running application
- ☐ Data
- ☐ Access restrictions
- ☐ Compliance with legal requirements

Which tasks the Cloud provider is responsible for?

Which model of responsibilities is offered by the cloud provider?

- ☐ shared responsibility - a responsibility between the customer and the Cloud provider

Does the Cloud provider offer a Service Level Agreement (SLA)?

- ☑ Yes
- ☐ No

Are the services offered and ensured in the SLAs sufficient?

- ☑ Yes
- ☐ No

Does the Cloud providers specify availability guarantee with respect to the Cloud services offered?

- ☑ Yes
- ☐ No

Which liability does the Cloud provider take for costs and damage caused by technical failure?

Does the Cloud provider inform his customers about scheduled maintenance periods?

- ☐ Yes
- ☐ No

Does the provider perform a regular, independent audit of its IT security status?

- ☑ Yes
- ☐ No

Anhang C.

Refactoring der Applikation Architektur

Plain Outsourcing - Network Layer

CNAME	If the source and target system are compatible and the source system is currently addressed via an URL, a change of the CNAME entry to point to the new target system is sufficient to allow all applications to use the new target system.
DNS Cache TTL	If the CNAME entry can be adapted to point to the new target system, make sure the DNS cache of all applications that use the old source system have a short time to live (TTL).
Firewall (Open Ports)	If the new target system is available under a different IP address or port then firewall settings need to be updated to allow all applications to access the new target system. It can also be useful to setup a Virtual Private Cloud (VPC). Make sure that not only the firewall settings for applications that use the old source system are adapted, but also the firewall settings for failover configurations, monitoring scripts and backup scripts.

Plain Outsourcing - Data Access Layer

Connection Strings	If the source and target system are compatible it can be sufficient to change the connection strings of all applications using the old source system to point to the address of the new target system. Make sure that not only applications that use the old source system are adapted, but also failover configurations, monitoring scripts and backup scripts.
Cloud Data Pattern	If the source and target system are not compatible, Cloud Data Patterns can be used to simulate missing functionality, e.g. non-standard SQL commands.
Semantic (Schema vs. DB Name)	Some data stores have a different semantic what a database and schema is (e.g. Oracle and SQL Server). This can be translated by the data access layer without adapting any upper application layers.

Naming Conventions	Most data stores have different naming conventions, therefore short, uppercase identifiers (table names, row names, constraint names) without special characters and potentially reserved words should be used. The data access layer can transform denied identifiers on the fly to allowed ones without the need to adapt upper application layers.
Data Types	Most data stores support different sets of data types which the data access layer can transform on the fly without the need to change upper application layers (e.g. BOOLEAN to BIT or CHAR(1)).
Semantics ("" vs. Null)	Some databases don't support null values for data entries, if the previous data store supported these, the data access layer can in some cases resolve that conflict if the applications itself don't differentiate between 'null' and empty values. If the applications differentiate between 'null' and empty values, the upper application layers have to be adopted.
Sorting	If the target system is a NoSQL datastore and supports sorting of result sets, it may be necessary to add zero-padding for numbers and an offset to columns with negative numbers if the ordering is done in lexicographical order.
Attribute Size	If the target system is a NoSQL datastore and allows only small values for attributes (e.g. max. 1kB), then larger values have to be stored in a Blob store and a pointer to the location in the Blob store can work as an intermediary attribute.

Plain Outsourcing - Application Logic Layer

Cloud Data Pattern Effects	If Cloud Data Patterns for Confidentiality are used in the data access layer, the application layer has to be aware that it might not have a full view on the actual data, since it could be anonymized, filtered or pseudonymized.
Product Change (Data Types, etc.)	If the product of the data store changes during the migration the semantics of e.g. comparison operators could have changed which cannot be translated by the data access layer and requires the application layer to be adopted.
Semantics (Comparisions, Locks)	Despite SQL being a standardised language, most vendors add product specific SQL commands or differ in the semantics of e.g. locks to allow dirty reads or more fine grained locks. These differences have to be identified during the migration and applied to the application layer.

Anhang D.

inspectIT Agent Konfiguration

```
## repository <IP> <port> <Agent Name>
##############################################
repository localhost 9070 inspectIT

## method-sensor-type <name> <fully-qualified-name> <priority> [<additional options>]
#############################################################################
# method-sensor-type average-timer info.novatec.inspectit.agent.sensor.method.averagetimer.AverageTimerSensor HIGH stringLength=100
method-sensor-type timer info.novatec.inspectit.agent.sensor.method.timer.TimerSensor MAX stringLength=100
method-sensor-type isequence info.novatec.inspectit.agent.sensor.method.invocationsequence.InvocationSequenceSensor INVOC stringLength=100
method-sensor-type jdbc-connection info.novatec.inspectit.agent.sensor.method.jdbc.ConnectionSensor MIN
method-sensor-type jdbc-prepared-statement info.novatec.inspectit.agent.sensor.method.jdbc.PreparedStatementSensor MIN stringLength=300
method-sensor-type jdbc-prepared-statement-parameter info.novatec.inspectit.agent.sensor.method.jdbc.PreparedStatementParameterSensor MIN
method-sensor-type jdbc-statement info.novatec.inspectit.agent.sensor.method.jdbc.StatementSensor MIN stringLength=300

## exception-sensor-type <fully-qualified-name> [<additional options>]
#####################################################################
exception-sensor-type info.novatec.inspectit.agent.sensor.exception.ExceptionSensor mode=simple stringLength=500

## platform-sensor-type <fully-qualified-name> [<additional options>]
####################################################################
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.ClassLoadingInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.CompilationInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.MemoryInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.CpuInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.RuntimeInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.SystemInformation
platform-sensor-type info.novatec.inspectit.agent.sensor.platform.ThreadInformation

## send-strategy <fully-qualified-name>
#######################################
send-strategy info.novatec.inspectit.agent.sending.impl.TimeStrategy time=5000
# send-strategy info.novatec.inspectit.agent.sending.impl.ListSizeStrategy size=10

## buffer-strategy <fully-qualified-name>
```

```
35 ########################################
36 buffer-strategy info.novatec.inspectit.agent.buffer.impl.SimpleBufferStrategy
37 #buffer-strategy info.novatec.inspectit.agent.buffer.impl.SizeBufferStrategy size=12
38
39 ## Ignore classes settings
40 ########################################
41 $include common/exclude-classes.cfg
42 exclude-class *$Proxy$_$$_*
43
44 ## SQL tracing
45 ##############
46 $include common/sql.cfg
47 ## Uncomment configuration file under to enable parameter binding for SQL queries. This
       feature allows to have
48 ## prepared Statements filled with the concrete parameter value that it was executed with,
       instead of just "?" values.
49 ## Bear in mind that activating this feature will affect performance in a negative way as
       more methods need to be instrumented.
50 # $include common/sql-parameters.cfg
51
52 ## Common technologies (please uncomment wanted)
53 ################################################
54 $include common/ejb.cfg
55 $include common/http.cfg
56 $include common/hibernate.cfg
57 # $include common/struts.cfg
58 $include common/jsf.cfg
59 $include common/jpa.cfg
60 # $include common/jta.cfg
61 #$include common/sql.cfg
62
63
64 #NovaERM Sensors
65 sensor timer * * @javax.ejb.Stateless modifiers=pub
66 sensor timer info.novatec.bpm.usertask.UserTaskService * interface=true
67 sensor isequence info.novatec.novaerm.web.tasks.UTBaseBean * superclass=true
68 sensor jdbc-prepared-statement org.postgresql.jdbc2.AbstractJdbc2Statement executeQuery()
69 sensor jdbc-prepared-statement org.postgresql.jdbc2.AbstractJdbc2Statement executeUpdate()
70 sensor jdbc-prepared-statement org.postgresql.jdbc2.AbstractJdbc2Statement execute()
71 #End NovaERM Sensors
```

Listing D.1: inspectIT Agent Konfiguration

Anhang E.

Testphasen

E.1. *seleniumHQ* Testergebnisse während der Evaluation Vorbereitung

Testname	Testdauer (Sek.)
DoProzessSequenziell	188
BewerbungAnlegen	34
BewerbungVorqualifizieren	37
BewerbungVervollstaendigen	28
MeinungAbgeben	24
BewerbungNachqualifizieren	20
TerminPlanen	18
TerminVereinbaren	15
BewerbungsgespraechNachbearbeiten	11
VAErstellenGraduand	14
VAFinalisierenGraduand	16
VAErstellenPracticalSchool	10
VAFinalisierenPracticalSchool	9
VAErstellenPracticalTrainee	14
VAFinalisierenPracticalTrainee	19
VAErstellenRegularEmployment	126
VAFinalisierenRegularEmployment	43
VAErstellenTempStaff	16
VAFinalisierenTempStaff	20
VAErstellenTrainee	13
VAFinalisierenTrainee	21

VAErstellenTraineeCE	17
VAFinalisierenTraineeCE	20
VAErstellenWorkingStudent	20
VAFinalisierenWorkingStudent	36
AblehnenInAnlegen	12
AblehnenInVorqualifizieren	11
AblehnenInNachqualifizieren	12
AblehnenInTerminVereinbaren	12
AblehnenInTerminPlanen	17
AblehnenInGespraechNachbearbeiten	12
AblehnenInVAErstellen	30
AblehnenMehrereBewerbungenInUebersicht	17
AblehnenInVAFinalisieren	13

Tabelle E.1.: Evaluation Vorbereitung *seleniumHQ* Testergebnisse

E.2. *seleniumHQ* Testergebnisse in der ersten Iteration

Testname	Erster Durchlauf (Sek.)	Zweiter Durchlauf (Sek.)	Dritter Durchlauf (Sek.)
DoProzessSequenziell	228	196	201
BewerbungAnlegen	34	34	34
BewerbungVorqualifizieren	37	34	34
BewerbungVervollstaendigen	34	34	34
MeinungAbgeben	19	19	18
BewerbungNachqualifizieren	20	18	16
TerminPlanen	19	15	14
TerminVereinbaren	18	13	12
BewerbungsgespraechNachbearbeiten	11	18	9
VAErstellenGraduand	14	21	15
VAFinalisierenGraduand	16	17	17
VAErstellenPracticalSchool	9	9	10

VAFinalisierenPracticalSchool	8	8	8
VAErstellenPracticalTrainee	15	15	14
VAFinalisierenPracticalTrainee	18	18	18
VAErstellenRegularEmployment	127	128	126
VAFinalisierenRegularEmployment	44	44	44
VAErstellenTempStaff	15	15	15
VAFinalisierenTempStaff	20	20	22
VAErstellenTrainee	13	15	13
VAFinalisierenTrainee	21	21	20
VAErstellenTraineeCE	17	18	17
VAFinalisierenTraineeCE	19	20	20
VAErstellenWorkingStudent	21	20	20
VAFinalisierenWorkingStudent	36	35	36
AblehnenInAnlegen	11	12	12
AblehnenInVorqualifizieren	11	12	11
AblehnenInNachqualifizieren	12	12	12
AblehnenInTerminVereinbaren	12	12	12
AblehnenInTerminPlanen	17	17	16
AblehnenInGespraechNachbearbeiten	12	12	12
AblehnenInVAErstellen	28	33	28
AblehnenMehrereBewerbungenInUebersicht	17	22	18
AblehnenInVAFinalisieren	14	14	14

Tabelle E.2.: *seleniumHQ* Testergebnisse in der ersten Iteration

E.3. *seleniumHQ* Testergebnisse in der zweiten Iteration

Testname	Erster Durchlauf (Sek.)	Zweiter Durchlauf (Sek.)
DoProzessSequenziell	190	198
BewerbungAnlegen	37	39
BewerbungVorqualifizieren	31	32
BewerbungVervollstaendigen	23	24
MeinungAbgeben	16	17
BewerbungNachqualifizieren	14	15
TerminPlanen	-	-
TerminVereinbaren	-	-
BewerbungsgespraechNachbearbeiten	-	-
VAErstellenGraduand	14	15
VAFinalisierenGraduand	18	24
VAErstellenPracticalSchool	10	10
VAFinalisierenPracticalSchool	10	11
VAErstellenPracticalTrainee	14	14
VAFinalisierenPracticalTrainee	20	20
VAErstellenRegularEmployment	130	131
VAFinalisierenRegularEmployment	54	46
VAErstellenTempStaff	15	15
VAFinalisierenTempStaff	22	24
VAErstellenTrainee	13	13
VAFinalisierenTrainee	21	24
VAErstellenTraineeCE	17	17
VAFinalisierenTraineeCE	21	23
VAErstellenWorkingStudent	19	19
VAFinalisierenWorkingStudent	22	23
AblehnenInAnlegen	13	14
AblehnenInVorqualifizieren	-	-
AblehnenInNachqualifizieren	-	-

AblehnenInTerminVereinbaren	-	-
AblehnenInTerminPlanen	-	-
AblehnenInGespraechNachbearbeiten	-	-
AblehnenInVAErstellen	-	-
AblehnenMehrereBewerbungenInUebersicht	-	-
AblehnenInVAFinalisieren	-	-

Tabelle E.3.: *seleniumHQ* Testergebnisse in der zweiten Iteration

Literaturverzeichnis

[1] ISO - International Organization for Standardization. http://www.iso.org/iso/home.html.

[2] National Institute of Standards and Technology - Computer Security Division - Computer Security Resource Center. http://csrc.nist.gov/.

[3] NovaTec Holding GmbH - Startseite. http://novatec-gmbh.de/.

[4] SeleniumHQ: Browser Automation. http://docs.seleniumhq.org/.

[5] Software Engineering Institute. http://www.sei.cmu.edu/.

[6] Amazon Web Services, Inc. Amazon AWS Produkte. http://aws.amazon.com/de/products/.

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

[8] Barbara Kitchenham, Lesley Pickard. Case Studies for Method and Tool Evaluation. http://www.idi.ntnu.no/grupper/su/courses/eit2004/CaseStudiesForMEthodAndToolEvaluation.pdf.

[9] Wolfgang Beywl. *Standards für Evaluation*. Geschäftsstelle DeGEval, Köln, 2002.

[10] Center for Civic Partnerships. Evaluation. http://www.civicpartnerships.org/docs/tools_resources/Evaluation%209.07.htm.

[11] David Silverman. *Interpreting Qualitative Data*. Sage Publications Ltd., 4th Revised edition. (1. November 2011).

[12] DeGEval - Gesellschaft für Evaluation e.V. DeGEval-Standards. http://www.degeval.de/degeval-standards.

[13] Dr. Thomas Greb. Prozessoptimierung mit CMMI und ITIL. http://www.thomas-greb-consulting.com/leistungen/prozessoptimierung-cmmi.html.

[14] Dr. Winston W. Royce. Managing the Development of Large Software Systems.

[15] E. Taylor-Powell, S. Steele. Collecting Evaluation Data: An Overview of Sources and Methods. http://learningstore.uwex.edu/assets/pdfs/G3658-4.pdf, 1996.

[16] Forrest Shull, Jeffrey Carver, Guilherme H. Travassos. An Empirical Methodology for Introducing Software Processes. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.7881&rep=rep1&type=pdf.

[17] gebeComGmbH. Übersicht der Prozessmodelle CMMI, SPICE, ITIL. `http://www.gebecom.de/Prozess.pdf`.

[18] Ian Sommerville. Software Process Models. `http://pdf.aminer.org/000/833/109/software_process_models.pdf`, 1996.

[19] IEC - International Electrotechnical Commission. Welcome to the IEC - International Electrotechnical Commission. `http://www.iec.ch/`.

[20] International Organization for Standardization ISO/ International Electrotechnical Commission IEC. Systems and software engineering – Systems and software product Quality Requirements and Evaluation (SQuaRE) – System and software quality models.

[21] ITIL. ITIL® Home. `http://www.itil-officialsite.com/`.

[22] Jinesh Varia. Migrating your Existing Applications to the AWS Cloud: A Phase-driven Approach to Cloud Migration.

[23] Jochen Ludewig, Horst Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.verlag GmbH, 2007.

[24] Gunter Kröber. ITIL v3 Überblick: Eine Übersicht.

[25] Michał Florys. ITIL® 2011 EDITION. `http://www.ctpartners.pl/page/111030/item,1079/ITILreg_2011_EDITION.xhtml`.

[26] NovaTec Holding GmbH. automaIT - Home. `http://www.automait.de/home/`.

[27] NovaTec Holding GmbH. InspectIT - Documentation. `https://documentation.novatec-gmbh.de/display/INSPECTIT/Home`.

[28] NovaTec Holding GmbH. InspectIT - Home. `http://www.inspectit.eu/home/`.

[29] NovaTec Holding GmbH. NovaERM - Home. `https://confluence.novatec-gmbh.de/display/NOVAERM/Home`.

[30] NovaTec Holding GmbH. TECH Paper: automaIT: Stabilität und Konsistenz im Betrieb von IT-Services durch modellbasierte Provisionierung.

[31] Peter Mell, Timothy Grance. The NIST Definition of Cloud Computing: Recommendations of the National Institute of Standards and Technology.

[32] Prof. Dr. Hans-Jürgen Scheruhn. *Einsatz von ITIL zur Prozessoptimierung im Rechenzentrum der Hochschule Harz am Beispiel des Release Managements*.

[33] Prof. Dr. N. Kratzke. IT-Management und Enterprise Architectures: Band: Enterprise Architectures und ITIL.

[34] Prof. Dr. Stockmann. Was ist eine gute Evaluation? Einführung zu Funktionen und Methoden von Evaluationsverfahren. `http://www.ceval.de/typo3/fileadmin/user_upload/PDFs/workpaper9.pdf`, 2004.

[35] Stefanie Winter. Quantitative vs. Qualitative Methoden. `http://imihome.imi.uni-karlsruhe.de/nquantitative_vs_qualitative_methoden_b.html`.

[36] Stephen H. Kan. *Metrics and Models in Software Quality Engineering: Second Edition*. 2004.

[37] Steve Strauch, Vasilios Andrikopoulos, Thomas Bachmann, and Frank Leymann. Migrating Application Data to the Cloud Using Cloud Data Patterns. In *Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER'13)*, pages 0–11. SciTePress, Mai 2013.

[38] Steve Strauch, Oliver Kopp, Frank Leymann, and Tobias Unger. A Taxonomy for Cloud Data Hosting Solutions. In *Proceedings of the International Conference on Cloud and Green Computing (CGC '11)*, pages 577–584. IEEE Computer Society, Dezember 2011.

[39] Thomas Bachmann. Entwicklung einer Methodik für die Migration der Datenbankschicht in die Cloud. Diploma thesis no. 3360, Universität Stuttgart, 2012.

[40] Tom Laszewski, Prakash Nauduri. *Migrating to the cloud: Oracle Client/Server Modernization*. SYNGRESS, 2011.

[41] J. van Bon. *IT Service Management basierend auf ITIL V3: Das Taschenbuch*. Van Haren Publishing, 2008.

[42] wibas GmbH. CMMI: Capability Maturity Model Integration. `http://www.wibas.de/publikationen/cmmi/cmmi_einfuehrung/index_de.html`.

Alle Links wurden am 30. August 2013 letztmalig überprüft.

Erklärung

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

Stuttgart, 30. August, 2013 _____

(Nikolay Nachev)