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

Diplomarbeit Nr. 3680

Der Weg in die Cloud: Entwicklung einer Migrations-Methodologie für Desktop- und Server-Anwendungen hin zu einer Software-as-a-Service Anwendung

Jakub Sabaciński

Studiengang:	Informatik
Prüfer:	Jun.-Prof. Dr.-Ing. Dimka Karastoyanova
Betreuer aus der Wissenschaft:	Steve Strauch
Betreuer aus der Industrie:	Boris Wehrle
begonnen am:	10. Juli, 2014
beendet am:	9. Januar, 2015
CR-Klassifizierung:	C.2.4, D.2.1, D.2.10, D.2.11, D.2.13, H.3.4, H.3.5, K.6.3

Zusammenfassung

Cloud Computing wird, dank seiner Vorteile, wie geringe Kapitalbindung oder Skalierbarkeit der Dienste, immer beliebter. Bereits mit geringem Aufwand können Cloud-Dienste entwickelt, in der Cloud bereitgestellt und einem breitem Spektrum von Kunden angeboten werden. Bei einer Entwicklung einer Software für die Cloud kann die Architektur entsprechend den Rahmenbedingungen aufgebaut werden. Unternehmen verfügen jedoch bereits über eine umfangreiche Code-Basis und können in ihrem Portfolio zahlreiche Anwendungen aufweisen, bei deren diese nicht berücksichtigt wurden. Eine solche Anwendung in die Cloud zu migrieren stellt oft eine komplexe, multidimensionale Herausforderung dar. Die zu migrierende Anwendung erfordert oft weitreichende Anpassungen, um in der Cloud zur Verfügung gestellt werden zu können. Um den Prozess der Cloud-Migration effizienter und effektiver durchführen zu können, empfiehlt sich, eine Methodologie einzusetzen, die das zum Ziel führende Vorgehen festlegt.

Im Rahmen dieser Diplomarbeit wird der aktuelle Stand in der Wissenschaft und in der Industrie im Bereich der vorhandenen Methodologien, Techniken und Vorgehensweise für die Cloud-Migration von On-Premise-Anwendungen erforscht. Das gewonnene Wissen wird dafür genutzt, eine eigene Cloud-Migrations-Methodologie zu entwickeln. Mit der Absicht der Firma *AIT - Applied Information Technologies GmbH & Co. KG*, ihre Anwendung *TFS ASAP* auf die Microsoft Azure Plattform zu migrieren, bot sich die Gelegenheit an, die entwickelte Cloud-Migrations-Methodologie in der Praxis zu verwenden. Die Migration der Anwendung *TFS ASAP* in die Cloud erfolgte unter Einsatz der im Rahmen dieser Diplomarbeit erarbeiteten Cloud-Migrations-Methodologie. Nach der durchgeführten Cloud-Migration werden der Prozess und die Ergebnisse analysiert und bewertet. Von den währenddessen gewonnenen Erkenntnissen und Erfahrungen werden Empfehlungen für künftige Migrationen von On-Premise-Anwendungen sowie die Entwicklung von Anwendungen, die zukünftig in die Cloud migriert werden könnten, abgeleitet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste von Codeausschnitten

1 Einführung

In letzten Jahren konnte man den Siegeszug vom Cloud-Computing beobachten und inzwischen ist diese Technologie zum festen Bestandteil der IT-Welt geworden. Cloud-Computing änderte die Art und Weise, wie man über die Bereitstellung und den Verbrauch von IT-Ressourcen dachte. Hier werden sie als Dienste in virtualisierter Form angeboten und können übers Internet in Anspruch genommen werden. Die Cloud-Dienste können nach Bedarf konsumiert werden und werden verbrauchsabhängig abgerechnet, was dem Wasserkonsum oder Stromkonsum in den Haushalten ähnelt. Der Benutzer hat damit den Eindruck, die Ressourcen wären unbegrenzt. Der Anwender nutzt sie über eine Web-Schnittstelle oder definierte Protokolle. Für die Cloud-Anbieter ist das Ziel, die Dienste gewinnbringend einem großen Kunden Kreis mit möglichst kleinem Verwaltungsaufwand zur Verfügung zu stellen. Aus der Verbrauchersicht, ist das die Möglichkeit, die Anschaffungskosten für eine IT-Infrastruktur zu reduzieren und sie in Betriebskosten umzuwandeln, welches einer der wichtigsten Vorteile vom Cloud-Computing ist. IT-Unternehmen können durch den Einsatz von Cloud-Computing neuartige Geschäftsmodelle umsetzen, die Wettbewerbsvorteile für sie ermöglichen. [AFG+09]

In den nachfolgenden Abschnitten werden die Problemstellung, die Zielsetzung sowie die Motivation dieser Arbeit dargestellt.

1.1 Problemstellung und Zielsetzung

Die AIT GmbH & Co. KG (AIT) ist ein in Stuttgart ansässiges IT-Unternehmen, das Dienstleistung im Bereich Softwareentwicklung und Beratung erbringt. Die AIT ist strategisch mit Microsoft verbunden und gehört seit Jahren zu Microsoft Gold Certified Partners (die höchste Stufe im Microsoft Partner-Programms). Aus diesem Grund werden von der AIT ausschließlich Microsoft Technologien eingesetzt. Eins der prominenten Geschäftsfelder der Firma ist die Beratung von Unternehmen während des Softwareentwicklungsprozesses unter dem Einsatz vom Team Foundation Server (TFS). Der TFS ist ein Softwarepaket, zu dessen Funktionalität unter anderem die Versionkontrolle und Projektverwaltung zählen. Die AIT hat in ihrem Portfolio ein Plug-In für den TFS, das seine Funktionalität um Automationen von Work Items erweitert, die durchgeführt werden, wenn ein Ereignis eintrifft. Ein Beispiel für eine Automation ist eine Aggregation von den Werten des Attributs "Remaining Work", die entlang der Work Item-Hierarchie propagiert werden.

Neben dem TFS bietet Microsoft seit einiger Zeit unter dem Namen Visual Studio Online (VSO) einen Cloud-Dienst an, der eine ähnliche Funktionalität wie der TFS bereitstellt. Der VSO ist mandantenfähig, dass heißt, der Dienst wird mehreren Benutzern zur Verfügung

gestellt. Für den VSO wurde auch die Visual Studio Online REST API veröffentlicht, die dem Benutzer ermöglicht, diverse Abfragen gegen den VSO über das HTTP-Protokoll durchzuführen. Mit der VSO REST API ist es auch möglich Subscriptions zu definieren. Dadurch können neue Integrationsszenarien realisiert werden. So ist es möglich, den Mechanismus der Ereignis-Notification abzubilden, den der TFS ASAP benutzt.

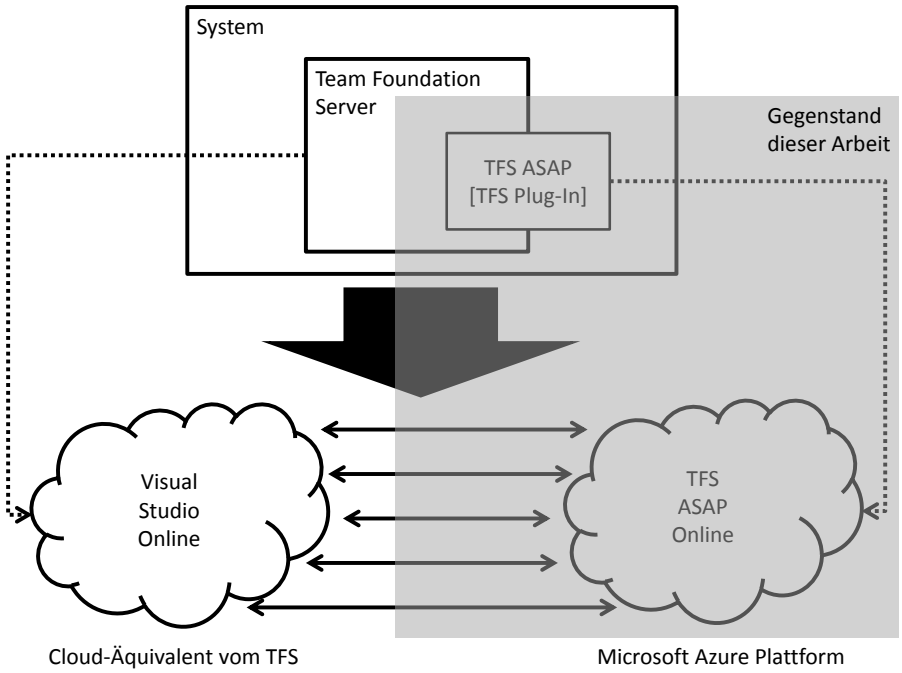

Abbildung 1.1: Schematische Darstellung der Migration vom TFS ASAP

Sowohl die oben erwähnten technischen Bedingungen, als auch die Chance für die AIT, in dem Cloud-Umfeld neue Lösungen für ihr Portfolio zu sammeln und Know-How in diesem Bereich aufzubauen, waren die Treiber für die Entscheidung, den TFS ASAP in die Cloud zu migrieren und mit dem VSO auf diese Art und Weise zu integrieren, wie TFS ASAP mit dem TFS integriert ist. Dabei ist auch zu gewährleisten, dass die TFS ASAP Online auch Mandantenfähigkeit aufweist und die Module, die für Abarbeitung der Ereignisse verantwortlich sind, wegen nicht vorhersehbarer Anzahl von Benutzern und zu bearbeitenden Ereignissen skalierbar sein sollen. Die Migration von TFS ASAP ist graphisch in der Abbildung 1.1 dargestellt.

1.2 Motivation

Die Vorteile des Cloud-Computing, wie geringe Kapitalbindung und schnelle Einstiegsmöglichkeit, locken IT-Unternehmen, Cloud-Anwendungen zu entwickeln. Eine explizit für die Cloud gedachte Neuentwicklung einer Applikation stellt eine gewisse Herausforderung dar, wie das auch jedes herkömmliche Softwareentwicklungsprojekt tut. In solchem Fall wird die

Anwendung bewusst für die Cloud entwickelt, von Anfang an unter Berücksichtigung der Fähigkeiten, die mit Cloud-Computing möglich sind.

Es kann durchaus sein, dass ein Unternehmen bereits über On-Premise-Lösungen in seinem Portfolio verfügt und die Entscheidungsträger ein Potential daran erkannt haben, eine solche bestehende Lösung oder ihre Teile in die Cloud zu migrieren. Eine Cloud-Migration einer existierenden Applikation bringt eine Reihe von Herausforderungen mit sich, welche adressiert werden müssen. So zum Beispiel die Auswahl des Cloud Anbieters oder die Anpassung der Architektur und des Quellcodes. Eine weitere Herausforderung ist, die Anwendung so anzupassen, dass sie sich auch durch die für die Cloud spezifischen Merkmale wie Mandantenfähigkeit oder Skalierbarkeit auszeichnet. [ABLS13]

Für eine erfolgreiche Durchführung einer Cloud-Migration kann eine Cloud-Migration-Methodologie eine besondere Unterstützung leisten. Sowohl in der Wissenschaft, als auch von den einzelnen Cloud-Anbietern, wurden bereits einige Methodologien erarbeitet, die sich mit unterschiedlichen Aspekten der Cloud-Migration auseinandersetzen. Das Ziel dieser Diplomarbeit ist es, eine von den bestehenden Methodologien auszuwählen, gegebenenfalls zu erweitern, an einem prototypischen Beispiel anzuwenden und darauffolgend nach einer gewählten Methode zu validieren und zu evaluieren.

Als die zu migrierende Software soll im Rahmen dieser Diplomarbeit der TFS ASAP[1] dienen. Der Urheber vom TFS ASAP ist die Firma AIT & Co. KG[2] . Da die AIT strategisch mit Microsoft verbunden ist, können für die Migration in die Cloud vom TFS ASAP nur Cloud Dienste von Microsoft berücksichtigt und eingesetzt werden.

Bei der Cloud-Migration einer Anwendung kann man vier grundsätzliche Arten unterscheiden [ABLS13]:

- Replace (Ersatz): Hier wird ein oder mehrere Komponenten durch einen Cloud-Dienst ersetzt. Diese Art von Migration kann noch bestimmter Anpassungen bedürfen.
- Partially migrate (Partielle Migration): Hier werden einzelne Funktionalitäten in die Cloud migriert.
- Migrate the whole software stack (Migration des gesamten Anwendung-Stack): Die Applikation wird ganz in eine Virtuelle Maschine ausgerollt, ohne vorher angepasst werden zu müssen.
- Cloudify : Die Anwendung wird als ganze in die Cloud migriert, indem bestimmte Architekturschichten auf einzelne Cloud-Dienste abgebildet werden, dass sich die migrierte Anwendung aus einer Menge von in der Cloud laufenden Diensten zusammensetzt.

Im Rahmen dieser Diplomarbeit wird aufgrund der Charakteristika vom TFS ASAP und der Anforderungen an das Endprodukt nur der vierte Typ der Cloud-Migration “Cloudify“ in Erwägung gezogen.

[1] TFS ASAP: http://www.tfsasap.com
[2] AIT GmbH & Co. KG.: http://www.aitgmbh.de

1.3 Abkürzungen

Im folgenden Abschnitt werden die in dieser Arbeit verwendeten Abkürzungen aufgelistet.

ADMC	Durchschnittliche Meilenstein-Verzögerung (engl. Average Delay of Milestone Completion)
AIT	AIT GmbH & Co. KG
AWS	Amazon Web Services
EF	Entity Framework
IaaS	Infrastructure as a Service
IIS	Internet Information Services
KT	Kalendertag
KW	Kalenderwoche
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
RDBMS	Relational Database Management System
SaaS	Software as a Service
TFS	Team Foundation Server
TTO	Prozesszeitplan-Einhaltung (engl. Time Table Observance)
VSO	Visual Studio Online

1.4 Gliederung

Die Diplomarbeit ist folgendermaßen gegliedert:

- **Grundlagen, Kapitel 2:** beschreibt die in der Arbeit benutzte Konzepte, Technologien und die zu migrierende Software.
- **Kenntnisstand, Kapitel 3:** zeigt den aktuellen Kenntnisstand bezüglich Cloud-Migration-Methodologien und Evaluierungsmethoden auf.
- **Methodologie für die Cloud-Migration, Kapitel 4:** präsentiert die in der Arbeit entwickelte angewandte Migrationsmethodologie und ihre Evaluierungsmethodologie.
- **Anwendung der Methodologie an TFS ASAP, Kapitel 5:** Hier wird Schritt für Schritt der Einsatz der entwickelten Methodologie während der Migration beschrieben.

- **Diskussion der Ergebnisse, Kapitel 6:** In diesem Kapitel werden die Ergebnisse der Migration diskutiert sowie die Empfehlungen basierend auf diesen Ergebnissen abgeleitet.
- **Zusammenfassung und Ausblick, Kapitel 7:** Hier werden das Fazit der Arbeit und zukünftige Untersuchungsmöglichkeiten dargestellt.

2 Grundlagen

In diesem Kapitel werden Technologien und Konzepte beschrieben, die für diese Diplomarbeit relevant sind.

2.1 Cloud-Computing

Die Cloud-Computing Technologie kann als Folge der Entwicklung im Bereich der Hardware und Software gesehen werden und stellt eine neuartige Form der Bereitstellung der Ressourcen und Dienste über das Internet dar. Cloud-Computing hat seinen Weg in die moderne IT geschafft und ist in diesem Kontext nicht mehr wegzudenken. Um den Begriff näherzubringen, kann die Definition von NIST Hilfe leisten. National Institute of Standards and Technology (NIST) definiert Cloud-Computing auf folgende Art und Weise:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [NIS11].

Laut NIST weist Cloud-Computing fünf Merkmale auf:

- On-demand self-service. Der Benutzer kann automatisch bei Bedarf benötigte Ressourcen in Anspruch nehmen.
- Broad network access. Auf die Ressourcen wird über das Internet mit Standardmechanismen zugegriffen.
- Resource pooling. Die Ressourcen werden in einem Pool in virtualisierter Form verwaltet und können durch mehrere Benutzer parallel benutzt werden.
- Rapid elasticity. Dem Benutzer können nach Bedarf zusätzliche Ressourcen provisioniert und wenn nicht mehr nötig freigegeben werden.
- Measured service. Der Ressourcenverbrauch wird aus Optimierungsgründen überwacht und gemessen.

2.1.1 Dienstmodelle

Im Cloud-Computing-Kontext kann man drei Dienstmodelle unterscheiden [NIS11]:

- **Infrastructure as a Service (IaaS):** Bei Infrastructure as a Service handelt es sich um die Bereitstellung von infrastrukturellen Ressourcen wie Datenhaltung, Server, Netzwerk oder Rechenleistung als ein virtualisierter Dienst. Die Details über die Hardware, auf der der Dienst aufbaut, bleiben dem Kunden verborgen und er hat darüber keine Kontrolle.
- **Platform as a Service (PaaS):** Platform as a Service stellt höherwertige IT-Dienste zur Verfügung, wie zum Beispiel Entwicklungs-, Datenbankenumgebung oder Middleware und bietet dem Benutzer die Möglichkeit an, eigene Anwendungen zu entwickeln. Der Benutzer hat jedoch keine Kontrolle über die zugrundeliegende Infrastruktur.
- **Software as a Service (SaaS):** Software as a Service ist eine Form von Cloud-Computing, in der Anwendungen dem Kunden als Dienst über das Internet bereitgestellt werden. Der Benutzer besitzt keine Informationen und Kontrolle über die zugrundeliegende IT-Infrastruktur.

2.1.2 Auslieferungsmodelle

NIST unterscheidet vier Modelle, wie Cloud-Computing Dienste dem Kunden bereitgestellt werden können [NIS11]:

- **Private Cloud:** Ausschließlich eine einzelne organisatorische Einheit nimmt Cloud Infrastruktur in Anspruch.
- **Community Cloud:** Eine Menge von separaten organisatorischen Einheiten, die ein gemeinsames Ziel verfolgen, nimmt die Cloud Infrastruktur in Anspruch.
- **Public Cloud:** Die Cloud Infrastruktur kann von jedem allgemein in Anspruch genommen werden.
- **Hybrid Cloud:** Eine hybride Lösung, wo die Cloud Infrastruktur eine Kombination verschiedener Cloud Infrastrukturen (Private, Public oder Community) darstellt.

2.1.3 Mandantenfähigkeit

Mandantenfähigkeit ist eine der Eigenschaften, die für den Erfolg von Cloud-Computing maßgeblich sind. Durch Mandantenfähigkeit wird es einer Anwendung ermöglicht, mehrere Konsumenten gleichzeitig mit einer einzigen Instanz zu bedienen. Auf diese Art und Weise können die Anwendungsbetreiber die Ressourcenauslastung maximieren und die Betriebskosten pro Benutzer reduzieren. In [SALM12] wird die Mandantenfähigkeit folgendermaßen definiert:

...sharing of the whole technological stack (hardware, operating system, middleware and application instances) at the same time by different tenants and their corresponding users.

Im Kontext der Mandantenfähigkeit sind zwei Konsumententypen zu unterscheiden: Mandanten und Benutzer. Mandant ist in diesem Zusammenhang ist die oberste Einheit, nach der das System für die Konsumenten aufgeteilt ist. Die Mandanten können zum Beispiel Unternehmen, Organisationen oder Abteilungen sein. Einem Mandanten können einzelne Benutzer zugeordnet werden und ein Benutzer kann wiederum mehreren Mandanten angehören. Mit Benutzern können Konsumenten auf feinerem Granularitätsebene unterschieden werden [SALM12]. Im Fall von TFS ASAP Online wird genau dem gleichen Ansatz gefolgt und die Anwendungskonsumenten werden ebenfalls einer Organisation zugeordnet. Eine Organisation kann mehrere Benutzer haben, die die Identität einer reellen Person repräsentieren. Die Benutzer werden durch ihre Microsoft-Konten identifiziert.

Die Mandantenfähigkeit kann aus zwei unterschiedlichen Perspektiven analysiert werden: des Dienstbetreibers und des Dienstkonsumenten. Für den Dienstbetreiber sind die Gründe, einen Dienst den Kunden mandantenfähig zu implementieren, wirtschaftlicher Natur. Durch Bereitstellung von Ressourcen, die von Konsumenten geteilt werden, kann die Ressourcenauslastung optimiert werden. Der Dienst wird nur ein Mal sofort für alle Konsumenten installiert und gepflegt. Auf diese Art und Weise werden die Kosten mit jedem zusätzlichen Konsumenten gesenkt und die dadurch erzielten Skaleneffekte tragen zu der Gewinnsteigung des Dienstbetreibers bei [SALM12]. Vor einem Dienstbetreiber stehen jedoch noch einige Herausforderungen. Am wichtigsten ist die Erfüllung der Mandantenanforderungen durch den Dienst. Der Betreiber möchte auch, dass sich der in Entwicklung und Betrieb investierte Aufwand am Ende rentiert. Die Ressourcenverbrauchüberwachung einzelner Mandanten soll vorhanden sein, damit man den Mandanten abrechnen kann. Die Überwachungsfunktion ist auch notwendig, um zeitgerecht aufgetretene Probleme zu identifizieren und zu beheben. Sie kann auch hilfreich für die Dienstbenutzungsanalyse sein. Die Skalierung von Ressourcen muss automatisch, abhängig von der momentanen Nachfrage, erfolgen [BHJ+12]. Aus der Sicht des Konsumenten muss die Benutzung eines mandantenfähigen Dienstes transparent sein. Das heißt, er hat den Eindruck, den Dienst allein in Anspruch zu nehmen, bis auf den Anmeldeprozess. Die Qualität und die Verfügbarkeit des Dienstes dürfen nicht durch Zugriffe anderer Konsumenten beeinträchtigt werden. [SALM12] Der Konsument soll auch die Möglichkeit haben, den Dienst nach seinen individuellen Anforderungen anzupassen [BHJ+12].

Aus den oben genannten Faktoren wurden in [SALM12] Anforderungen an eine mandantenfähige Anwendung abgeleitet. Aus der Sicht von TFS ASAP Online sind folgende interessant:

- **Mandantenbewusstsein** - Der Dienst muss fähig sein, mehrere Mandanten zu identifizieren und zu verwalten. Unterstützung der mandantenbasierten Identifizierung und hierarchischen Zugriffskontrolle für Mandanten und ihre Benutzer sind gewünscht.
- **Mandantenisolation** - Die Mandanten sollten voneinander isoliert bleiben. Sie sollten keinen Zugriff auf die Daten (Datenisolation) oder Rechenressourcen (Performanzisolation) anderer Mandanten haben.

- **Sicherheit** - Autorisierungs- und Authentifizierungsmechanismen müssen implementiert sein.

Wenn man eine Anwendung in die Cloud migriert, möchte man sich auch die Vorteile der Mandantenfähigkeit zunutze machen. In solchem Fall müssen für die bestehende Codebasis entsprechende Anpassung und Adaptationen vorgenommen werden [ABLS13]. Da die Anforderung an die Migration von TFS ASAP die Umsetzung der Mandantenfähigkeit vorsieht, werden wir uns mit dieser Problematik noch in weiteren Teilen der Arbeit beschäftigen.

2.1.4 Skalierbarkeit

Skalierbarkeit bedeutet die Fähigkeit eines Systems, eine steigende Belastung zu bestehen, ohne die Dienstqualität zu reduzieren, wenn gleichzeitig die darunterliegenden Ressourcen erhöht werden. [LR00] Durch die Hinzunahme von zusätzlichen Ressourcen steigt bei einem skalierbaren System die gesamte Leistungsfähigkeit des Systems an.

Skalierbarkeit kann auf zweierlei Arten sichergestellt werden und deswegen wird zwischen zwei Typen von Skalierbarkeit unterschieden. Die Skalierungstypen werden graphisch in der Abbildung 2.1 dargestellt:

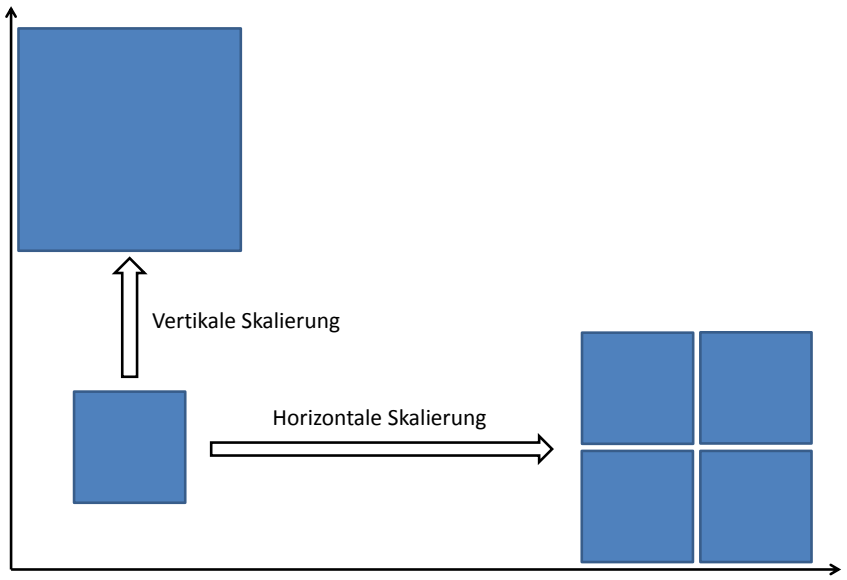

Abbildung 2.1: Die Skalierungsarten

- **Vertikale Skalierbarkeit:** Bei der vertikalen Skalierung wird die Leistungsfähigkeit eines System vergrößert, indem die Leistungsmerkmale einer Maschine/eines Knotens, wie zum Beispiel CPU oder Arbeitsspeicher gesteigert werden. In solch einem Fall

wird die Software nicht zusätzlich angepasst. Diese Vorgehensweise ist aufgrund der technologischen Grenzen eingeschränkt.

- **Horizontale Skalierbarkeit:** Bei der horizontalen Skalierung wird die Leistungsfähigkeit des Systems durch die Hinzunahme von zusätzlichen Rechnern/Knoten gesteigert. Bei dieser Vorgehensweise sind keine technologischem Grenzen gesetzt. Die Software muss aber wegen der Parallelität optimiert werden.

Skalierbarkeit ist eines der Merkmale, die den Erfolg des Cloud-Computing ausmachen. Skalierbarkeit ist möglich dank der Virtualisierung von Ressourcen [BBG11].

Je nach Szenario und Anforderungen, kann bei einer Cloud-Migration die Skalierbarkeit der zu migrierenden Anwendung gefordert werden. In solchem Fall sollt darüber nachgedacht werden, zu welchem Zweck sie dienen soll, welcher Typ von Skalierbarkeit erwünscht ist und gegebenenfalls wie die Applikation angepasst werden soll, damit ihre Skalierung möglich ist [ABLS13]. Zu diesen Überlegungen sollten noch die wirtschaftliche Seite betrachtet werden, d.h. - wie kann das Hinzuschalten von zusätzlichen Ressourcen die Ausgaben des Anbieters beeinflussen [TUS11].

2.1.5 Cloud Migration

Die Cloud Migration von einer On-Premise-Anwendung ist die Motivation und zentrales Thema dieser Diplomarbeit. In diesem Abschnitt wird der Begriff näher gebracht, mögliche Gründe einer Migration werden genannt, sowie die Arten der Cloud-Migration werden diskutiert.

Definition

Die Cloud-Migration ist ein Prozess, in dem eine Anwendung oder ihre Teile samt ihrer Umgebung, in eine Cloud-Umgebung versetzt werden. Dazu zählen digitale inhalte einer Organisation, Dienste, IT-Ressources oder die Anwendung selbst. Im Rahmen des Migrationsprozesses können Teile der zu migrierenden Applikation durch die Organisation selbst betrieben werden, während andere Teile von Cloud-Anbietern in einer Cloud-Umgebung betrieben werden.[PXW13] Eine Cloud-Migration bringt gewisse Risiken mit sich, deswegen sind Schritte wie Analyse, Planung und Durchführung notwendig, um den Anforderungen gerecht zu werden. [BLS11]

Gründe für eine Cloud-Migration

Unternehmen können unterschiedliche Gründe haben, um eine Anwendung in die Cloud zu migrieren. Eine große Rolle spielen dabei die allgemeinen Vorteile vom Cloud-Computing:

- Kostenersparnisse [JAP13]
- Flexibilität (Skalierbarkeit, Pay-per-use) [JAP13]

- Effiziente Ressourcenauslastung [JAP13]
- Ermöglichung innovativer Geschäftsszenarien [WAB+09]
- Die Firma AIT erbringt unter anderem Leistungen im Bereich Cloud-Consulting, ein Cloud-Produkt wirkt als Bestätigung des Know-Hows in diesem Bereich

Cloud-Migrationsarten

Die Cloud-Migration ist ein allgemeiner Begriff. Man kann jedoch eine Verfeinerung vornehmen und bestimmte Arten davon erkennen. In [ABLS13] werden folgende Cloud-Migrations-Arten identifiziert. Es wird dabei davon ausgegangen, dass die zu migrierende Applikation nach dem 3-Schichten-Architekturmodell aufgebaut ist.

- **Replace**: In dieser Migrationsart werden einzelne Komponenten durch Cloud-Dienste ersetzt. Sie stellt eine am wenigsten invasive Art einer Cloud Migration dar. Beispielweise kann die Daten- und/oder Anwendungsschicht migriert werden. Dies kann zusätzliche Konfigurations- und Anpassungsaktivitäten erfordern, um mit möglichen Inkompatibilitäten umzugehen, sowohl in den zu migrierenden Komponenten, als auch in den restlichen Teilen der Anwendung.
- **Partially Migrate**: In dieser Migrationsart ist eine Funktionalität der Anwendung zu migrieren. Das kann zur Folge haben, dass Komponenten aus mehreren Architekturschichten migriert werden sollen, die die zu migrierende Funktionalität umsetzen.
- **Migrate the whole stack**: In dieser Migrationsart wird die gesamte Anwendung in ihrem Ganzen migriert. Es ist der klassische Fall einer Cloud-Migration. Dies kann bewerkstelligt werden, indem die Anwendung in einer virtuellen Maschine abgekapselt wird und so in der Cloud-Umgebung betrieben wird.
- **Cloudify**: In dieser Migrationsart handelt sich um eine komplette Migration. Ein Zusammenschluss von Cloud-Diensten bildet die Funktionalität einer Anwendung ab. Ähnlich wie in der ersten Migrationsart, diverse Anpassungsaktivitäten können nötig sein.

In dieser Arbeit wird der Migrationsart *Cloudify* besonders viel Aufmerksamkeit geschenkt, denn die Migration von TFS ASAP sieht vor, dass die gesamte Anwendung in die Cloud migriert und dabei die einzelnen Komponenten auf unterschiedliche Dienste der Microsoft Azure Plattform abgebildet werden. Die daraus resultierenden Anpassungen werden detaillierter in weiteren Teilen der Diplomarbeit beschrieben.

2.2 Team Foundation Server

In diesem und folgenden Abschnitten wird die Software vorgestellt, die im Bereich der Migration von TFS ASAP relevent sind. Zuerst erfolgt die Beschreibung von TFS als Programm, für welches TFS ASAP ein Plug-In ist. Dann wird die Cloud-Version von TFS - Visual Studio

Online präsentiert. Darauffolgend werden TFS ASAP als die zu migrierende Anwendung und TFS ASAP Online als die Cloud-Version von TFS ASAP beschrieben.

Team Foundation Server[1] ist ein Produkt von Microsoft, das die Benutzer bei einem Softwareentwicklungsprozess unterstützt. Seine Funktionalität deckt folgende Bereiche ab [BWHK13]:

- Projektverwaltung
- Work Item Tracking
- Versionskontrolle
- Test Verwaltung
- Build automation
- Berichtwesen

Für diese Arbeit ist die Funktionalität **Work Item Tracking** von Relevanz. Sie leistet den Projektverantwortlichen Hilfe bei der Projektverwaltung. Work Item ist ein Sammelbegriff für Artefakte, die bei einer Projektplanung und -verwaltung verfolgt werden. Dazu zählen zum Beispiel: Requirements (Anforderungen), Bugs, oder Testfälle. Work Items sind in einer XML-Datei definiert, in der unter anderem sowohl ihre Attribute, als auch mögliche hierarchische Beziehungen zwischen anderen Work Items festgelegt sind. Gängige Attribute sind hier [BWHK13]:

- ID
- Zustand
- Nummerische Attribute (zum Beispiel: Remaining Work, Completed Work, Effort)

Die Work Items können hierarchisch in Form eines Baums in einer Eltern-Kind-Beziehung organisiert werden.

Team Foundation Server bietet die Möglichkeit an, seine Funktionalität mittels Plug-Ins zu erweitern. Eine der Möglichkeit ist die Implementierung der ISubscriber-Schnittstelle. Ein solches Plug-In kann vom TFS unter anderem über Änderungen von einzelnen Work Items informiert werden und entsprechende Aktion durchführen, wobei zu beachten ist, dass diese Aktion synchron ausgeführt wird. Eine weitere Möglichkeit stellt die Implementierung der *ITeamFoundationJobExtension*-Schnittstelle dar. Eine Klasse, die diese Schnittstelle implementiert, kann in die *JobQueue* (eine Art Warteschlange) eingereiht werden, und vom *JobAgent* ausgeführt werden. Die Jobs können einmalig oder in zeitlichen Abschnitten der aus der Warteschlange abgenommen werden, abhängig davon, wie sie eingereiht wurden. Die Ausführung eines Job erfolgt asynchron [BWHK13].

Darüber hinaus stellt Microsoft TFS API[2] den Entwicklern zur Verfügung. Mit TFS API werden Bibliotheken bereitgestellt, die unter anderem Manipulationen von Work Items ermöglichen.

[1] Team Foundation Server: http://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx
[2] Extending Team Foundation: http://msdn.microsoft.com/en-us/en-us/library/bb130146.aspx

2.3 Visual Studio Online

Während Team Foundation Server eine On-Premise-Lösung darstellt, befindet sich seit einiger Zeit im Angebot von Microsoft ein Cloud-Äquivalent. Dabei handelt es sich um VSO[3]. Dieses Produkt wird vollständig in der Cloud betrieben und deckt die meisten Funktionalitäten vom TFS ab. Microsoft möchte in der Zukunft immer mehr auf Cloud-Computing setzen und VSO soll allmählich an Bedeutung gewinnen. Der VSO erfreut sich eines häufigen Updatezyklus. Die Benutzer vom VSO können schneller die neuen Funktionalitäten genießen, als die Benutzer vom TFS.

Eine Voraussetzung, um VSO benutzen zu können, ist ein Microsoft Account. Mit dem Microsoft-Kontoi werden sich die Benutzer bei dem Dienst anmelden. Mit einem Microsoft-Konto kann man ein Visual Studio Online Konto anlegen, in dem man Projekte neu erstellen und verwalten kann. Die Benutzer können andere Benutzer in sein Visual Studio Online Konto einladen. Die Benutzer melden sich im Kontext eines VSO-Kontos an und können nur die VSO-Konto-spezifischen Daten sehen.

2.3.1 Visual Studio Online REST API

Visual Studio Online REST API[4] ist eine REST-Basierte API, die dem Benutzer diverse Interaktionen mit Visual Studio Online mit Hilfe von HTTP-Aufrufen ermöglicht. Dadurch lassen sich einige Szenarien realisieren, darunter auch dieses Szenario, das dieser Diplomarbeit zu Grunde liegt und deren motivierendes Beispiel darstellt. Die VSO REST API bietet somit einen Mechanismus an, diverse Artefakte, wie zum Beispiel Work Items, Builds oder Testpläne, abzufragen, anzulegen oder zu verändern. Dies erfolgt durch HTTP-Abfragen, die einen speziellen URL-Muster folgen müssen und in JSON-Format verschickt werden müssen. Gegebenenfalls wird mit dem Aufruf auch ein JSON-String mitgeliefert.

Für die Migration von TFS ASAP sind folgend Features von der VSO REST API von besonderer Bedeutung:

- Autorisierung - Um eine Abfrage erfolgreich durchführen zu können, muss der Benutzer autorisiert werden. Es gibt zwei Ansätze um das zu bewerkstelligen: *Basic Authentification* und *OAuth2*[5]. Mit Basic Authentification schickt der Benutzer einfach seinen Benutzernamen und sein Passwort mit dem HTTP-Aufruf. Mit OAuth2 wird wiederum ein Bearer-Token geschickt. Diese Vorgehensweise ist für Cloud-Dienste gedacht. Zuerst musst der Dienst-Betreiber seinen Dienst bei VSO registrieren. Wenn sein Dienst registriert ist, kann man den Benutzer auf eine speziellen URL weiterleiten, wo er sich gegen VSO verifiziert, und dann autorisiert die Applikation, in seinem Namen REST-Aufrufe machen zu dürfen. Wenn das gemacht ist, wird der Token für den Benutzer an die

[3] What is Visual Studio Online? - http://www.visualstudio.com/en-us/products/what-is-visual-studio-online-vs.aspx

[4] Visual Studio Online REST API Reference: http://www.visualstudio.com/en-us/integrate/reference/reference-vso-overview-vsi

[5] Authentication for the REST APIs: http://www.visualstudio.com/en-us/integrate/get-started/get-started-auth-introduction-vsi

Adresse geschickt, die während der Registrierung der Applikation angegeben wurde. Ein Token ist in der Regel 15 Minuten gültig und muss gegebenenfalls aktualisiert werden.

- Subscription (Service Hooks)[6] für ein Projekt lässt sich eine sogenannte Subscription anlegen. Dies ist ein Mechanismus, der es ermöglicht, Empfänger über unterschiedliche Ereignisse zu informieren. Um eine Subscription programmatisch anzulegen, ist die Authentifizierung mit OAuth2 notwendig. Dafür werden Informationen wie Projekt-Id, Art des Ereignissen und der Empfänger angegeben. Aus der Sicht vom TFS ASAP Online sind die interessanten Ereignisse "workitem.created“ und "workitem.updated“ und interessante Empfänger Azure Service Bus und Web Hooks, die Web Services sind.
- Work Item Tracking[7] ermöglicht es, Work Items abzufragenn und zu editieren. Die Service-Antworten sind in Form eins JSON-Strings dargestellt.

2.4 TFS ASAP

Eines der Ziele dieser Arbeit ist, TFS ASAP in die Cloud zu migrieren. Aus dem Grund wird die Software in diesem Abschnitt näher erläutert, um zum besseren Verständnis beizutragen. Zunächst wird auf die allgemeine Funktionalität eingegangen. Dann werden Mechanismen von TFS beschrieben, die das Programm benutzt und zusammen mit der Architektur, wird dargestellt, wie sich TFS ASAP diese Mechanismen einbindet. Em Ende wird schematisch erläutert, wie eine Automation durchgeführt wird.

2.4.1 Einführung

TFS ASAP[8] ist eine serverseitige Erweiterung von TFS, die dem Benutzer dabei hilft, die Work Items im Laufe eines Softwareentwicklungsprojekts zu verwalten. Diese Tätigkeit bedarf einer Menge von vielen manuellen, fehleranfälligen Arbeiten, die als Ziel haben, die Work Items in einem gepflegten Zustand zu halten. Jede Änderung von Attributen eines Work Items hat ein manuelles Editieren von zusammenhängenden Work Items zu Folge. Manche von diesen Aufgaben kommen immer wieder vor und lassen sich deswegen automatisieren.

In der Abbildung 2.2 wird eine beispielhafte Situation dargestellt, die passieren kann, wenn keine vom TFS ASAP bereitgestellten Automations durchgeführt werden. Man sieht eine Menge von Work Items. Die Work Items des Typs *Task* sind die *Requirement*s untergeordnet. Wenn der Zustand eines *Task* auf *Active* geändert wurde, heißt es, dass die Arbeit an der übergeordneten *Requirement* bereits begonnen wurde und sein Zustand auch *Active* werden soll. Auch wenn alle untergeordnete *Resolved* werden, soll die übergeordnete *Requirement*

[6]Subscribe to Visual Studio Online events from another service: http://www.visualstudio.com/en-us/integrate/get-started/get-started-service-hooks-creating-and-managing-vsi
[7]Work Item Tracking: http://www.visualstudio.com/en-us/integrate/reference/reference-vso-work-item-overview-vsi
[8]TFS ASAP : http://www.tfsasap.com/

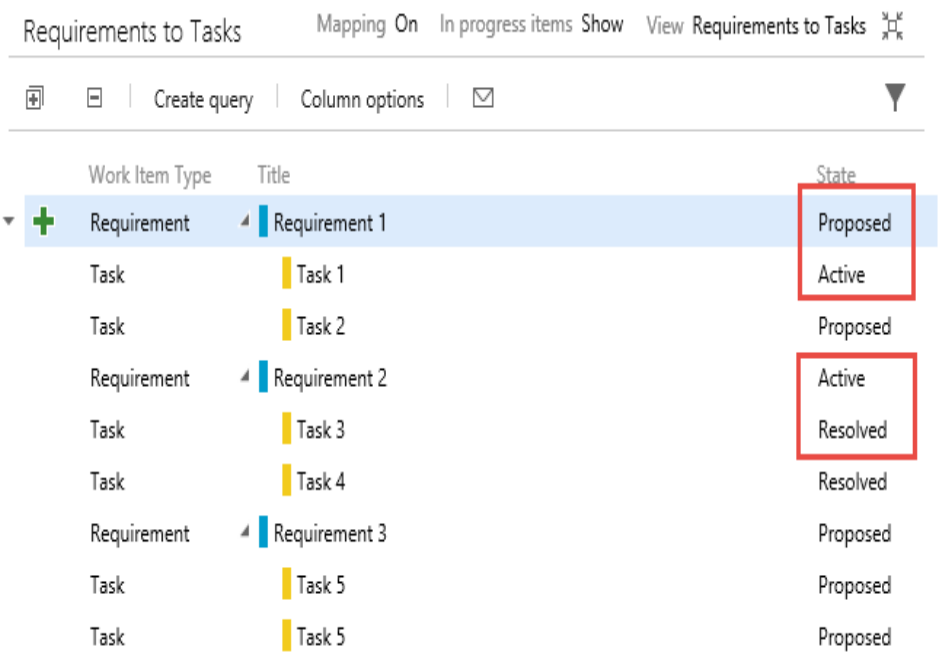

Abbildung 2.2: Zustand der Work Items ohne ausgeführten Automations [AITa]

auch *Resolved* werden. TFS verfügt nicht über solche Funktionalität, die diese Änderungen automatisch durchführen würde, und deswegen muss der Benutzer die Änderungen manuell durchführen. Die dadurch entstandenen Inkonsistenzen ist in der Abbildung 2.2 durch Rahmen markiert. In der Abbildung 2.3 sind die Work Items in einem konsistentem Zustand zu sehen, der von TFS ASAP sichergestellt wird.

Weitere Automations sind möglich, die viel komplexer sind und sich auch auf andere Attribute eines Work Items beziehen. Die Definitionen von Automations sind in einer XML-Datei abgespeichert. Es besteht die Möglichkeit, sie an seine Bedürfnisse durch Editieren der XML-Datei anzupassen. Es gibt zwei Arten von Automations: *triggered* und *scheduled*. Die *triggered* Automations sind ereignisgesteuert und werden ausgeführt, wenn ein Ereignis eintritt. Die *scheduled* Automations sind wiederum zeigesteuert und werden in Zeitintervallen ausgeführt.

2.4.2 Architektur

In der Abbildung 2.4 wird der schematische Aufbau der TFS ASAP Software dargestellt und ihr Zusammenhang mit dem TFS. Der Ablauf beim Eintritt eines Ereignisses ist auch zu sehen. Wie man der Abbildung 2.4 entnehmen kann, besteht TFS ASAP aus folgenden Komponenten:

- **ASP.NET MVC Webanwendung** - Management Portal: Diese Komponente ist verantwortlich für die Interaktion mit dem Benutzer. Er kann hier Ausführung von Automa-

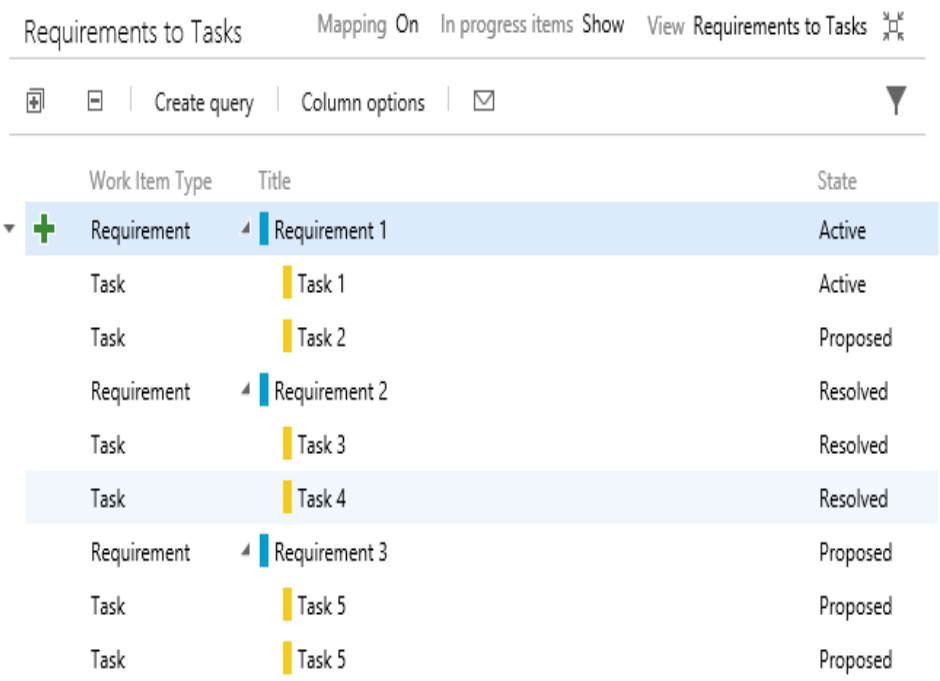

Abbildung 2.3: Zustand der Work Items mit ausgeführten Automations [AITa]

tions für die Projekt im TFS de- und aktivieren. Diese Komponente wird mit einigen Änderungen der Mandantenfähigkeit wegen in die Cloud übernommen.

- **ISubscriber** - Diese Komponente implementiert die Schnittstelle ISubscriber. ISubscriber registriert sich für die Entgegennahme von Ereignissen. Die Ereignisse werden von TFS Web Services geliefert, die im IIS betrieben werden. Da der ISubscriber vom TFS synchron ausgeführt wird, werden die Ereignisse nicht bearbeitet, sondern in einer Datenbank gespeichert. Diese Funktionalität wird in der Cloud-Version vom TFS ASAP überflüssig und somit durch andere Mechanismen ersetzt.

- **Datenbank** - In der Datenbank werden die zu bearbeitenden Ereignisse für die JobExtension gespeichert. Diese Komponente wird in der Cloud-Version durch eine Azure Service Bus Queue ersetzt.

- **JobExtension** - Diese Komponente implementiert die ITeamFoundationJobExtension Schnittstelle. Eine JobExtension wird in definierten Zeitintervalle vom TFS Job Agent aufgerufen, der dafür gedacht ist, sich wiederholende und langlaufende Aufgaben durchzuführen. Wenn das geschieht, liest sie die Ereignisse von der Datenbank und übergibt sie an die Automation-Objekte. Diese Funktionalität wird in der Cloud-Version als Worker Rolle in Rahmens eines Azure Cloud Service.

- **Automations** - Das sind die eigentlichen Implementierung der Automations, die die Ausführungslogik repräsentieren. Dazu wird TFS API eingesetzt.

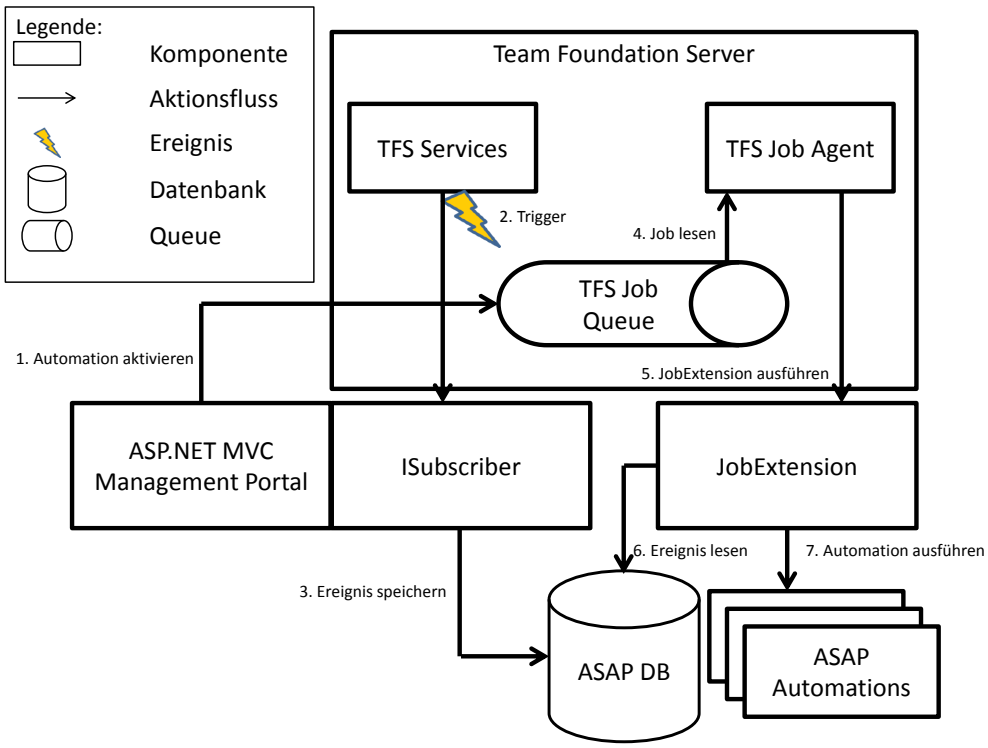

Abbildung 2.4: Architektur von TFS ASAP

2.5 TFS ASAP Online

TFS ASAP Online - so soll die in die Cloud migrierte Version vom TFS ASAP heißen. Der TFS ASAP wurde entwickelt, weil der TFS über manche Funktionalitäten nicht verfügte und der Benutzer wiederkommende, manuelle Aufgaben durchführen musste. Durch die vom TFS bereitgestellte Automations sind die Benutzer von diesen Aufgaben befreit und die Projektverwaltung ist einfacher. In dem von Microsoft eingeführten Dienst, der auf dem TFS basiert, sind solche Funktionen ebenfalls nicht berücksichtigt. Dadurch ist eine Lücke entstanden, denn im Moment gibt es auf dem Markt kein Produkt, das die Unzulänglichkeiten vom VSO beheben würde, wie der TFS ASAP das mit dem TFS macht. Die Hauptidee, die hinter der Migration vom TFS ASAP steckt, ist diese Lücke zu schließen. Die AIT als Microsoft-Partner möchte diesen Trend nicht verpassen und seine Präsenz in diesem Bereich auch erweitern. Aus dem Grund ist es für sie vorteilhaft, eine Cloud-Lösung in ihrem Portfolio aufweisen zu können. Das ist auch von Bedeutung angesichts der Tatsache, dass die Firma Leistungen im Bereich Beratung für Azure-Lösungen erbringt. Einer der Faktoren ist auch der Know-How-Gewinn im Team.

TFS ASAP Online soll anfangs unentgeltlich als Möglichkeit angeboten werden, die Funktionalität vom TFS ASAP den Kunden präsentieren zu können. Jedoch ein späterer Vertrieb gegen Bezahlung ist nicht auszuschließen. Darüber hinaus wird TFS ASAP Online intern von der Firma AIT benutzt, weil sie mit ihren Projekten auf VSO umgestiegen ist. Eine Beta-

Version vom TFS ASAP Online mit implementiertem Back-End befindet sich momentan bereits im Betrieb.

2.5.1 Architektur

In diesem Abschnitt wird die Architektur von TFS ASAP Online dargestellt. Einzelne Komponenten werden beschrieben, zusammen mit ihrer Funktionalitäten und Microsoft Azure Diensten, auf denen sie basieren. Es wird erklärt, welche Komponenten von TFS ASAP übernommen werden, welche nicht, und welche Elemente des TFS nachgebaut werden müssen, um das geforderte Szenario realisieren zu können.

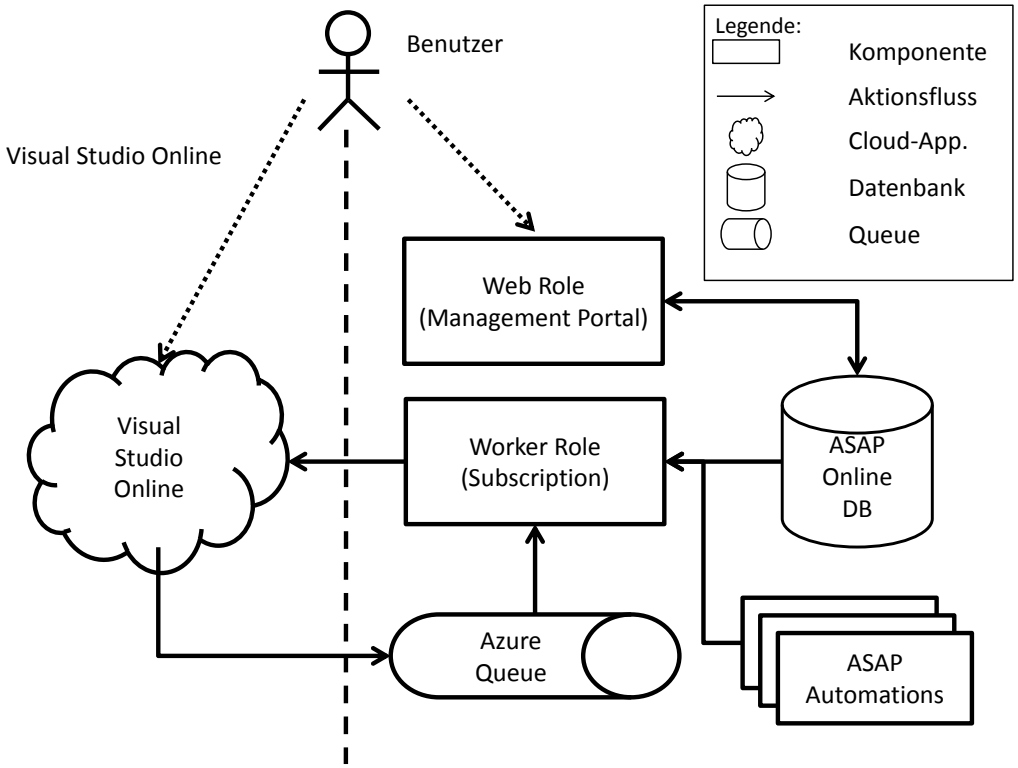

Abbildung 2.5: Architektur von TFS ASAP Online

Die Architektur vom TFS ASAP Online ist graphisch in der Abbildung 2.5 dargestellt. Wie man der Abbildung entnehmen kann, besteht TFS ASAP Online aus folgenden Komponenten:

- **Azure Service Bus Queue** - Die Rolle dieser Komponente ist, die Ereignisse vom VSO entgegenzunehmen und abzulegen, bevor sie von einer Worker Role abgenommen werden. Im Vergleich zu TFS ASAP ist das eine neue Implementierung. Damit wird eine Komponente nachgebildet, die in der On-Premise-Lösung vom TFS bereitgestellt wird.
- **Worker Role (Subscription)** - Diese Komponente nimmt die Ereignisse aus der Warteschlange, ermittelt das Projekt, welches das Ereignis betrifft, lädt für das Projekt

aktivierte Automations, und führt sie aus. An dieser Stelle kann die Applikation skaliert werden, in dem neue Instanzen der Worker Role dazu geschaltet werden, wenn die Menge der Ereignisse in der Warteschlange zu groß wird. Diese Komponente basiert auf der JobExtension vom TFS. Sie wird nach dem Anwurfsmuster *Competing Consumers Pattern* gebaut [HSB+14].

- **ASAP Online DB** - In der Datenbank werden Benutzer und ihre Daten, ihre Projekte und für sie aktivierte Automations abgelegt. Das ist eine Neuimplementierung. Die aktivierten Automations werden im Fall vom TFS ASAP in der Registry abgelegt. Außerdem werden hier auch Automation-Definitionen gespeichert. Beim TFS ASAP erfolgte das im Dateisystem.
- **Web Role Management Portal** - Das Management Portal ist eine Web-GUI, mit der der Benutzer Automations für seine Projekte aktivieren und deaktivieren kann. Die Implementierung wird so im TFS ASAP Online beibehalten. Das Portal wird jedoch um Module erweitert, die den Benutzern das Registrieren, Anmelden, Autorisieren vom TFS ASAP Online, Änderungen in ihren Projekten mit VSO REST API durchzuführen, Hinzufügen von Projekten und Einsicht in ihre Nutzungsstatistiken ermöglichen. Das ist aufgrund der Anforderung bezüglich der Mandantenfähigkeit erforderlich.
- **ASAP Automations** - Hier befindet sich der Code von Automations. Im Laufe des Projekts wurden einige Veränderungen vorgenommen. Zuerst war das die Abstrahierung der Abhängigkeiten zu TFS API. In der Implementierung für TFS ASAP Online wird an ihrer Stelle die VSO REST API verwendet. Aufgrund von Entscheidungsänderungen, die detaillierter in weiteren Teilen der Diplomarbeit beschrieben werden, wurde im Endeffekt TFS API zur Manipulation von Work Items eingesetzt.

2.5.2 Anmerkungen

Hiermit werden einige Anmerkungen aufgelistet, die zum besseren Verständnis der Problemstellung beitragen:

- TFS ASAP und TFS ASAP Online sollten nach der Migration eine möglichst große gemeinsame Code-Basis haben.
- Zur Manipulation am TFS wird beim TFS ASAP die TFS API benutzt. Um die TFS API beim TFS ASAP Online zu benutzen, muss der Benutzer aufgefordert werden, seinen Benutzernamen und sein Passwort an TFS ASAP Online zu schicken. Aus Sicherheitsgründen wurde solch eine Vorgehensweise ausgeschlossen. Deswegen wird dafür die VSO REST API verwendet. Die VSO REST API verfügt über geforderte Fähigkeiten, und stellt Mechanismen bereit, mit denen ein Benutzer die Anwendung autorisieren kann, Änderungen in seinem Namen vorzunehmen.
- VSO REST API befindet sich momentan in Version 1.2 und unterliegt stetiger Veränderung.[9]

[9] Versions of the REST APIs: http://www.visualstudio.com/en-us/integrate/support/dn800942

- TFS ASAP und TFS ASAP Online sind zwei separate Produkte und können deswegen gleichzeitig vertrieben werden, ohne das Gefahr zu laufen, dass Kunden von einem Produkt zum anderen wechseln. TFS ASAP bedient die Benutzer von Team Foundation Server, während TFS ASAP Online die Benutzer von Visual Studio Online bedient.
- TFS ASAP wird im TFS-Prozess ausgeführt und basiert auf den davon bereitgestellten Diensten, wie JobQueue oder JobAgent. Da beim Hosting in der Cloud nicht auf diese Dienste zugegriffen werden kann, müssen sie vom Entwickler in Azure nachgebildet werden.

2.6 Andere eingesetzte Technologien

In diesem Abschnitt werden andere Technologien beschrieben, die während der Migration von TFS ASAP Verwendung fanden. Das sind Visual Studio, Entity Framweork und ASP.NET MVC.

2.6.1 Visual Studio

Visual Studio[10] ist eine Programmierumgebung, die besonders für die Entwicklung von Anwendungen in .NET-Sprachen eingesetzt wird und ist in diesem Bereich ein De-Facto-Standard. Neben dem Texteditor bietet das Werkzeug auch eine Integration mit Team Foundation Server und Visual Studio Online an und somit lassen sie sich von dort bedienen.

Visual Studio kann der Entwickler auch mit seinem Azure Konto verknüpfen, was die Verwaltung und Veröffentlichung von Cloud Diensten direkt von der Umgebung ermöglicht. Mit Azure Emulator lassen sich die entwickelten Dienste lokal testen und debuggen. Die laufenden Dienste lassen sich mit Hilfe von Visual Studio debuggen.

Die oben genannten Funktionalitäten werden im Laufe der Arbeit zur Migration von TFS ASAP eingesetzt.

2.6.2 Entity Framework

Entity Framework[11] ist ein OR-Mapper. Mit Hilfe eines OR-Mappers werden die Artefakte einer relationalen Welt auf .NET-Objekte abgebildet. Entity Framework abstrahiert den Zugriff auf die Datenhaltung. Auf diese Weise können Datensätze in einer Datenbank manipuliert werden, ohne SQL-Anweisungen im Code schreiben zu müssen. Entity Framework kann auch Datenbanken von anderen Datenbankanbietern unterstützen. [SS14]

Die Abbildung zwischen den relationalen und objektorientierten Elementen wird im Entity Data Model definiert, das aus drei Komponenten besteht [SS14]:

[10] Visual Studio resources: http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx
[11] Entity Framework: https://entityframework.codeplex.com/

- **Konzeptmodell** beinhaltet das visuelle Objektmodell, in dem Klassen und Beziehungen zwischen ihnen definiert sind.
- **Speichermodel** beinhaltet das Datenbankschema.
- **Map** definiert die Abbildung zwischen den Elementen des Konzeptmodells und Speichermodells

Im Kontext von Entity Framework bestehen drei mögliche Ansätze, wie das Entity Data Model erzeugt wird [SS14]:

- **Database First:** Das Entity Data Model wird aus einer bereits existierenden Datenbank generiert.
- **Model First:** Der Entwickler erstellt zuerst das Objektmodell im visuellen Editor von Visual Studio, aus dem später ein Datenbankschema generiert wird.
- **Code first:** Hier werden zuerst Klassen entwickelt, aus welchen ein Datenmodell hergeleitet wird. Zusätzliche Konfigurationen können programmatisch definiert werden. Dieser Ansatz wird bei der Migration von TFS ASAP verwendet, um die Benutzerverwaltungslogik zu implementieren.

2.6.3 ASP.NET MVC

ASP.NET MVC[12] ist ein Framework von Microsoft für Web-Anwendungen. Es implementiert das Model-View-Controller Architekturmuster. Nach dem Muster besteht eine Anwendung aus drei Modulen, die unterschiedlichen funktionalen Aufgaben nachkommen. Dadurch wird die Trennung zwischen Logik und Präsentation gewährleistet [SS14]:

- **Model** Im Modell werden die Geschäftsobjekte repräsentiert.
- **View** Die View ist für die graphische Darstellung der Objekte und die Interaktion mit dem Benutzer verantwortlich.
- **Controller** Der Controller agiert als Bindeglied zwischen dem Modell und der View. An ihn werden Anfragen und Benutzereingaben verschickt, die er entsprechend verarbeitet.

Die ASP.NET MVC-Anwendungen lassen sich einfach in die Azure Cloud deployen. Zu diesem Zweck bieten sich zwei Möglichkeiten an: Web Role im Rahmen eines Cloud Service oder Web Site. Die Veröffentlichung in der Azure Plattform kann direkt von Visual Studio aus erfolgen, oder in Azure Management Portal kann das Deployment-Paket hochgeladen werden, dass während des Build-Prozesses entsteht[13].

Bei TFS ASAP wird die ASP.NET MVC Technologie eingesetzt, um das Managementportal zu implementieren. Die migrierte Version soll diese Implementation übernehmen und sie um Komponenten für die Mandantenfähigkeit erweitern.

[12] ASP.NET MVC: http://www.asp.net/mvc
[13] How to deploy ASP.NET MVC projects.: http://www.asp.net/mvc/overview/deployment

2.7 Microsoft Azure Plattform

Da für die Migration der Software TFS ASAP nur die Cloud-Dienste von Microsoft in Betracht gezogen werden, wird in diesem Abschnitt die Microsoft Azure Plattform eingeführt und beschrieben. Zuerst erfolgt eine kurze Beschreibung und dann werden die für diese Diplomarbeit relevanten Dienste vorgestellt.

2.7.1 Einführung

Microsoft Azure[14] ist eine Cloud-Plattform von Microsoft, die ihr Debüt 2010 feierte. Mit Microsoft Azure bietet Microsoft Infrastructure-as-a-Service und Platform-as-a-Service Dienste in Form einer Public Cloud an. Es kann beobachtet werden, dass im Laufe der Zeit neue Dienste dem Angebot hinzugefügt werden, oder die bereits angebotenen Dienste Veränderungen unterliegen, deswegen ist es wichtig, sich immer mit aktueller Dokumentation auseinanderzusetzen, damit man auf dem Laufenden bleibt.

2.7.2 Komponenten

In Microsoft Azure werden zahlreiche Dienste angeboten. Um einen Überblick zu schaffen, werden die für die Diplomarbeit relevanten Dienste in drei Gruppen aufgeteilt. Den Gruppen werden einzelne Komponenten aus dem Microsoft Azure Angebot zugeordnet und schließlich wird ihre Funktionalität, Unterschiede zu anderen Komponenten und ihre Relevanz für die Migration von TFS ASAP beschrieben. Die drei erwähnten Gruppen sind:

- **Rechendienste** - Komponenten in dieser Gruppe dienen dazu, Applikationen auszuführen.
- **Datendienste** - Komponenten in dieser Gruppe dienen dazu, Daten zu speichern und verwalten
- **Azure Service Bus** - Das ist eine Komponente zur Anwendungsintegration durch Nachrichtenaustausch.

Rechendienste

In der Gruppe Rechendienste werden Dienste für Rechenleistungsaufgaben zusammengefasst. Es gibt drei Ausführungsmodelle: Virtuelle Maschinen, Cloud Services und Web Sites [Mica]. Wie in der Tabelle 2.1 dargestellt, unterscheiden sie sich voneinander bezüglich des Kontrollgrades über die zugrundeliegende Infrastruktur und der Benutzungskomplexität. Es ist möglich, dass mehrere von diesen Komponenten in der Architektur einer Cloud-Anwendung auftreten.

Die Dienste in der Gruppe Rechendienste sind:

[14]Microsoft Azure: http://azure.microsoft.com/

Dienst	Kontrollgrad	Benutzungskomplexitätsgrad
Azure Virtual Machine	hoch	hoch
Cloud Service	mittel	mittel
Web Site	gering	gering

Tabelle 2.1: Ausführungsmodelle von Microsoft Azure Diensten - Vergleich

- **Azure Virtual Machine [Mica], [Micl], [Micc], [Mici]**

  Azure Virtual Machine stellt ein IaaS-Angebot dar. Wenn man eine virtuelle Maschine anmietet, muss man ein Image in Form einer VHD (Virtual Hard Disc) liefern. Virtuelle Maschinen unterstützen Microsoft Windows und Linux als Betriebssysteme. Zusätzlich stehen dem Benutzer erweiterte Images mit vorinstallierter Software, wie zum Beispiel MS SQL Server, zur Verfügung. Die Abrechnung erfolgt zeitabhängig.

  Für die virtuellen Maschinen hat der Benutzer die größte Kontrolle über das Betriebssystem, installierte Software und Dienste. Die administrativen Tätigkeiten lassen sich durch die Remote-Desktop-Verbindung erledigen. Die Verwaltung der virtuellen Maschine auf der Infrastrukturebene, wie die Installation von Updates, muss jedoch in Kauf genommen und vom Benutzer übernommen werden.

  Die Installation und Wartung einer Cloud-Applikation in der virtuellen Maschine muss ebenfalls vom Benutzer durchgeführt werden. Im Zusammenhang mit Migration können virtuelle Maschinen eine gute Lösung darstellen, wenn es möglich ist, dass die ganze Anwendung (Migrate the whole stack) migriert wird. In eine virtuelle Maschine können auch einzelne Komponenten einer Applikation ausgelagert werden und als Dienste der restlichen Anwendung bereitgestellt werden. Als Beispiel bietet sich hier die Auslagerung einer Datenbank in eine auf einer virtuellen Maschine installierte Datenbank.

- **Cloud Service [Mica], [Mici], [Micm], [Micc]**

  Cloud Service ist ein PaaS-Angebot für skalierbare Anwendungen. Ein Cloud Service besteht aus einer Kombination von einer oder mehreren Web Roles und Worker Roles. Die Roles sind virtuelle Maschinen, die auf Windows Server basieren. Web Role hat die Aufgabe, HTTP-Anfragen entgegenzunehmen, und sie zur Verarbeitung an eine Worker Role weiterzuleiten. Dies kann über eine Warteschlange (Azure Service Bus) geschehen. Worker Roles sind dazu geeignet, asynchrone, lange andauernde Operationen durchzuführen. Die Roles können unabhängig voneinander skaliert werden, um dem ankommenden Workload gerecht zu werden. Die virtuellen Maschinen für die Roles werden von der Azure Plattform gestartet. Während der Auslieferung muss der Anwendungscode und eine Konfigurationsdatei hochgeladen werden, die Einstellungen beinhaltet, wie zum Beispiel die Anzahl oder die Größe von den Roles. Der Benutzer hat größere Kontrolle über die zugrundeliegende Infrastruktur als bei Web Sites und kann zum Beispiel Remote-Desktop-Verbindungen mit einer Role herstellen. In dieser Hinsicht liegt deswegen ein Cloud Service zwischen Azure Virtual Machine und Web Site.

Die Aufteilung des Dienstes in Web und Worker Roles ermöglicht Umsetzung von komplexen, mehrschichtigen Architekturen und Entwicklung von hochskalierbaren Anwendungen. Diese Tatsache, sowie die Flexibilität und der Kompromiss zwischen Kontrolle und Übernahme von administrativen Tätigkeiten durch die Plattform sind der Grund dafür, dass dieses Ausführungsmodell für die Cloud-Migration der Software TFS ASAP ausgewählt wurde.

- **Web Site [Mica], [Micc], [Mici]**

  Web Site ist eine Komponente der Gruppe Rechendienste besonders für diejenigen geeignet, die vor allem schnell eine skalierbare Lösung ausliefern möchten und sich nicht mit der Verwaltung der zugrundeliegenden Infrastruktur beschäftigen wollen. Web Site ist ein PaaS-Dienst, der auf Azure Cloud Services baut. Die Applikation läuft auf einer Mengen von mit anderen Kunden geteilten virtuellen Maschinen, obwohl eine Option besteht, seine Anwendung in einer dedizierten virtuellen Maschine zu betreiben.

  Es ist möglich, eine bestehende Web-Applikation ohne Änderungen in die Cloud zu versetzen. Es besteht die Möglichkeit, die Skalierung der Anwendung zu steuern. Es handelt sich hiermit sowohl um Hinzuschalten von neuen Instanzen, als auch im Fall einer dedizierten virtuellen Maschine, ihre Leistungsfähigkeit zu vergrößern (scale up). Der Lastausgleich zwischen den Anwendungsinstanzen wird von der Microsoft Azure Platform übernommen.

  Web Sites ist eine gute Alternative, wenn man mit minimalen administrativer Arbeit eine laufende Web-Applikation erstellen möchte. Die Infrstrukturverwaltung wird den Entwicklern gespart und sie können sich der Implementierung der Anwendung widmen.

Datendienste

Im Rahmen von Microsoft Azure werden auch Datendienste angeboten, die dem Benutzer ermöglichen, Daten in der Cloud zu speichern, zu editieren und zu verwalten. Zur Datensicherheit werden die Daten in drei (mit Möglichkeit, auf sechs zu erweitern) Kopien physikalisch abgelegt.

Folgende Datendienste werden angeboten:

- **Virtuelle Maschine [Mici], [CTG+12]** - Eine der Möglichkeiten ist, eine Virtuelle Maschine mit installiertem SQL Server oder anderem Relational Database Management System (RDBMS) in Anspruch zu nehmen. Der Benutzer ist hier nicht eingeschränkt und kann zu nicht-relationalen Datenbanken greifen. Als Beispiel bieten sich hier NoSQL-Datenbanken an. Mit dieser Lösung treten übliche Vor- und Nachteile auf, die mit dem Einsatz von virtuellen Maschinen verbunden sind, wie die Notwendigkeit, administrative Arbeit selbst durchführen zu müssen.

- **Azure SQL Database, [Mici], [CTG+12]** - Das ist ein Platform-as-a-Service-Dienst für relationale Daten, der die Funktionalitäten eines gewöhnlichen RDBMS in der Cloud bereitstellt. Der Dienst basiert auf dem Microsoft SQL Server, jedoch nicht jede Funktionalität wird durch den Dienst abgebildet. Beispielweise fehlt die Unterstützung für einige Merkmale der Sprache T-SQL. Wie üblich bei den PaaS-Angeboten, ist der Benutzer von jeglichen administrativen Aufgaben befreit, wobei er immer noch Kontrolle über die Daten und Zugriff auf sie hat.

- **Tables [Mici], [CTG+12], [Mich]** - Das ist eine NoSQL-Angebot, das zum Speichern von größeren Mengen strukturierter, nicht-relationaler Daten verwendet wird. Eine Dateneinheit ist eine Entity, die aus einer Menge von *Schlüßel-Wert*-Verknüpfungen besteht. Die *Entities* werden in *Tables* gespeichert, die von den Tabellen aus der relationalen Welt das unterscheidet, dass sie keine definierte Schema haben. So kann eine *Table* unterschiedliche Typen von *Entitites* beinhalten. Mit geschickter Setzung von Eigenschaften kann man die Ausführung von Abfragen beschleunigen. Fortgeschrittenere Abfragemethoden, wie zum Beispiel *JOINT*'s, werden jedoch nicht unterstützt. *Tables* bieten nichtsdestotrotz einen schnellen Zugriff auf die Daten und eine gute Skalierbarkeit. Ein günstigerer Preis im Vergleich zu Azure SQL Database und die oben genannten Merkmale bewirken, dass *Tables* in manchen Fällen eine bessere Alternative darstellen.

- **Azure Blobs [Mici], [CTG+12], [Mice]** Mit Azure Blobs wird den Kunden ein Dienst zur Datenspeicherung in der Cloud angeboten, die sich besonders für große Mengen Dateien, Bilder und anderen unstrukturierten Daten eignen. Azure Blobs zeichnet sich durch günstige Preise, Einfachheit und Skalierbarkeit aus. Als Blobs werden Dateien beliebigen Typs und Größe bezeichnet und sie werden in einem Container zusammengefasst und verwaltet. Die Blobs liegen oft anderen Microsoft Azure-Diensten zugrunde. Es sind zwei Arten von Blobs zu unterscheiden: Block Blobs und Page Blobs. Während Block Blobs für Speichern und Senden von Cloud-Objekten, sind Page Blobs für IaaS-Festplatten optimiert. Bei Azure Virtual Machines werden intern Blobs eingesetzt.

Azure Service Bus

Azure Service Bus ist ein Infrastrukturdienst der Azure Plattform, der einen nachrichtenbasierten Kommunikationsmechanismus zwischen Anwendungen oder Anwendungskomponenten anbietet und damit verschiedene Integrationsszenarien ermöglicht. [Mici]

Um Microsoft Azure Service Bus zu benutzen, braucht man zuerst einen Namespace. Dieser dient dazu, Instanzen der verfügbaren Kommunikationsmechanismen zu gruppieren. Azure Service Bus bietet dem Benutzer drei Kommunikationsmechanismen: Queue, Topic und Relay. Eine Ausprägung solchen Mechanismus wird durch seinen Namen und den zugehörigen Namespace identifiziert. [Micb]

Hier wird nun kurz auf die Kommunikationsmechanismen eingegangen, die mit Azure Service Bus möglich sind:

- **Queue [Mici], [Micb], [Micf], [MS14]** Mit Queues wird eine asynchrone, in eine Richtung ablaufende Kommunikation zwischen Sender und Empfänger angeboten. Zwischen den beiden Kommunikationsparteien befindet sich der Vermittler - in diesem Fall die Queue. Der Sender schickt eine Nachricht an den Broker, die dieser speichert. Die gespeicherte Nachricht kann vom Empfänger abgenommen werden. Der Vorteil eines solchen Mechanismus ist, dass die Kommunikationspartner nicht gleichzeitig online sein müssen, um einen erfolgreichen Nachrichtenaustausch durchzuführen. Es findet auch keine direkte Verbindung zwischen ihnen statt. Eine Nachricht kann nur von einem Empfänger bedient werden. Es gibt zwei Modi, wie eine Nachricht gelesen werden kann. Mit ReceiveAndDelete wird sie aus der Queue gelöscht. Sie ist nicht mehr verfügbar, wenn der Empfänger abstürzt. Mit PeekLock wird die Nachricht gelesen und von der Queue gesperrt und für die anderen Empfänger unsichtbar gemacht. Wenn der Empfänger die Nachricht erfolgreich verarbeitet hat, wird er das der Queue mitteilen. Dann wird die Nachricht aus der Queue gelöscht. Der Empfänger kann auch mitteilen, dass die Verarbeitung der Nachricht fehlgeschlagen ist und dann gibt die Queue die Nachricht wieder frei. Das gleiche gilt für den Fall, dass ein definierter Zeitintervall ohne Rückmeldung des Empfängers abgelaufen ist. Falls es mehrere Empfänger gibt, erfolgt mit einer Queue ein automatischer Lastausgleich zwischen diesen. Eine Skalierung ist auch möglich und besteht darin, dass neue Empfänger dazu geschaltet werden. Diese Eigenschaften soll auch die Cloud-Version der Software TFS ASAP nutzen.
- **Topic [Mici], [Micb], [Micg], [MS14]** Topics, ähnlich wie Queues, bieten eine asynchrone, in eine Richtung ablaufende auf Nachrichten basierende Kommunikation an, wo die Kommunikationspartner nicht direkt verbunden sind. Der Unterschied ist, dass die Kommunikation dem Publish-Subscriber-Modell folgt. Ein Empfänger kann mittels eines Filters definieren, welche Nachrichten er bekommen möchte. Auf diese Art und Weise kann eine Nachricht von mehreren Applikationen empfangen werden, wenn sie die gleichen Filter definiert haben.
- **Relay [Mici], [Micb], [Mick], [MS14]** Im Gegensatz zu den bereits erwähnten Mechanismen, erfolgt hier die Kommunikation in beide Richtungen und die Nachrichten werden nicht vom Azure Service Bus gespeichert. Dieser Dienst ermöglicht es, eine hybride Cloud-Anwendung zu entwickeln, derer Komponenten sowohl in der Cloud, als auch on-premise laufen. Mit dem Service Bus Relay lassen sich sicher die im Unternehmensnetzwerk betriebenen Dienste in der Public Cloud bereitstellen, ohne die Notwendigkeit, die Sicherheitseinstellungen des Netzwerks zu verändern. Somit können Anwendungen, die sich außerhalb des Unternehmensnetzes befinden, oder in der Cloud laufende Komponenten der hybriden Cloud Applikation, auf die Funktionalität der lokalen, on-premise-Anwendung zugreifen. Bei Service Bus Relay können entsprechende Sicherheitseinstellungen vorgenommen werden, dass der Zugriff auf die Dienste kontrolliert werden kann. Somit kann den Nutzern auf eine sichere Weise die Funktionalität des Dienstes über die Cloud zur Verfügung gestellt werden.

3 Kenntnisstand

In diesem Kapitel wird der aktuelle Kenntnisstand zur Cloud-Migration untersucht sowie bezüglich des Titels und Inhalts dieser Diplomarbeit positioniert. Es wird sowohl auf Publikationen aus der Wissenschaft als auch aus der Industrie eingegangen. Da im Rahmen dieser Diplomarbeit davon ausgegangen wird, dass für die Migration der Software TFS ASAP die Cloud-Dienste von Microsoft Azure in Anspruch genommen werden, werden vor allem die Arbeiten von Microsoft berücksichtigt. Im Kapitel werden ebenfalls Evaluierungsmethoden und Metriken für einen Prozess vorgestellt. Dieses Kapitel bildet somit die Grundlage für weitere Teile dieser Diplomarbeit. Basierend auf den Inhalten aus diesem Kapitel wird eine Methodologie gewählt und angepasst, nach der die Migration durchgeführt wird.

3.1 Cloud-Migrationsmethodologien aus der Wissenschaft

Bei der Untersuchung des Kenntnisstands zum Thema Cloud-Migration leistet die Arbeit [JAP13] eine besondere Hilfe. In [JAP13] führten die Autoren eine Untersuchung durch und konsolidierten das vorhandene Wissen über den Wissenstand zur Cloud-Migration und ihren Methoden, Techniken und Werkzeugen. In der Arbeit ging man von der Migration einer On-Premise-Applikation in eine Cloud-Umgebung aus. Migrationen zwischen unterschiedlichen Cloud-Anbietern, Auslieferungsmodellen oder Cloud-Diensten waren nicht Gegenstand der Forschung. Bei der Auswahl der Publikationen wurde der wissenschaftliche Wert berücksichtigt. Aus diesem Grund wurden manche Arbeiten nicht selektiert, die in dieser Diplomarbeit erwähnt werden.

Basierend auf dem aus der Literatur gewonnenem Wissen wurde ein Referenzmodell für Cloud-Migration entwickelt. Das Referenzmodell besteht aus vier Prozessen die Aufgaben beinhalten. Die Prozesse sind:

- Planung - Das Ergebnis dieses Prozesses ist der Migrationsplan. Beinhaltet Aufgaben wie: Machbarkeitsstudie, Migrationsanforderungsanalyse, Anbieter-Auswahl, Auswahl zu migrierender Subsysteme, Auswahl Cloud-Dienste, Entwicklung der Migrationsstrategie
- Durchführung - Die eigentliche Migrationsdurchführung. Beinhaltet Aufgaben wie: Datenextraktion, Architekturanpassung, Code-Adaptation.
- Evaluierung - Bewertung der Migration. Beinhaltet Aufgaben wie: Testen, Validierung und Deployment.

- Cross-cutting Concerns - In diesem Prozess werden Aufgaben zusammengefasst, die während Durchführung jedes Hauptprozesses angewandt werden und deswegen nicht einem einzelnen Prozess zugeordnet werden können. Beinhaltet Aufgaben wie: Governance, Sicherheitsanalyse, Ausbildung, Aufwandsschätzung, Organisationsänderungen, Analyse der Mandantenfähigkeit und Skalierbarkeit

Den einzelnen Aufgaben aus dem Referenzmodell werden Publikationen zugeordnet. Die Autoren konnten feststellen, dass das Testen, Architektur-Recovery und die Auswahl der Cloud-Anbieter die populärsten Aufgaben in der Literatur sind, während Cross-cutting Concerns und Architekturadaptationen stiefmütterlich behandelt werden. Für die Prozesse *Planung* und *Evaluierung* besteht auch ein automatisierter Werkzeugensupport, im Gegensatz zu anderen Prozessen.

Die Autoren entwickelten auch ein Rahmenwerk zur Klassifizierung der Publikationen. Das Rahmenwerk definiert zwölf Kriterien, die in folgenden Gruppen zusammengefasst sind: Reifegrad (Art der Studie und die Evaluierungsmethode), Migrationscharakterisierung (u.a. Migrationsmittel, Migrationstyp, Migrationsaufgaben, Migrationszweck), Migrationssupport (Migrationsprozess vollständig/partiell beschrieben, automatisierbar, Werkzeugunterstützung) und Randbedingungen (z.B. Anwendungsarchitekturstil, Zielplattform und Dienstmodell).

Basierend auf dem Referenzmodell und dem Rahmenwerk konnte festgestellt werden, dass Kostenersparnisse, Skalierbarkeit, effiziente Ressourcenausnutzung und erhöhte Flexibilität die Haupttreiber einer Cloud-Migration sind. Während am Anfang vor allem die Migration des gesamtem Anwendungs-Stack erforscht wurde, ist im Moment die Cloudifizierung am populärsten. Die Übersichtsarbeit identifizierte Methoden und Techniken. Obwohl die Forschungsreife wächst, mangelt es immer noch an Adressierung der Cross-cutting Concerns und Werkzeugunterstützung.

Diese Arbeit bietet eine Übersicht über existierende Methodologien an. Mit dem entwickelten Rahmenwerk wurden die Prozesse und Aufgaben einer Cloud-Migration identifiziert. Für diese Diplomarbeit hat sich diese Arbeit als eine Referenz mit wichtigsten Studien im Bereich Cloud-Computing erwiesen, und das entwickelte Referenzmodell diente als Übersicht über mögliche Aufgaben bei einer Cloud-Migration. Nicht alle Publikationen, die in dieser Diplomarbeit beschrieben werden, wurden wegen ihren geringen wissenschaftlichen Werts erfasst.

In dem Artikel [ABLS13] untersuchen die Autoren ebenso eine Cloud-Migration. Die Autoren entwickeln eine Klassifizierung für Cloud-Migrationen, die in dieser Diplomarbeit und auch in [JAP13] benutzt wird. Laut diesen gibt es vier Cloud-Migrationsarten: Replace, Partially Migrate, Migrate the whole Stack und Cloudify. Die Migrationsarten sind im Abschnitt 2.1.5 beschrieben. Basierend auf einer Beispielanwendung, die nach der Schichtenarchitektur aufgebaut ist, werden Ausprägungen einer Migration untersucht, in welchen einzeln Architekturkomponenten oder Schichten migriert werden. So kann man eine Migration als Folge von nacheinander folgenden Migrationen einzelner Anwendungsbausteine sehen. Ausgehend von einem 3-Schichten-Modell, beschäftigen sich die Autoren mit den Herausforderungen, den Anpassungen und ihren Auswirkungen auf andere Anwendungsteile.

Der Fokus liegt hier auf der Daten- und Geschäftslogikschicht. Im Fall von TFS ASAP wird keine Datenschicht migriert, sondern für die Benutzerverwaltung implementiert, um die Mandantenfähigkeit-Anforderung zu erfüllen. In dem der Datenschichtmigration gewidmeten Teil ist die Taxonomie für Cloud Data Hosting [SKLU11] Lösungen nennenswert, die bei einer Entscheidung über die Auswahl einer Cloud-Daten-Betreiber helfen sollte. Um Inkompatibilitäten zu vermeiden, werden Cloud Data Patterns [SABL13] diskutiert, die Unterschiede zwischen On-Premise- und Cloud-Lösungen reduzieren. Es werden ebenfalls Anpassungen an der Datenzugriffsschicht genannt, die dazu notwendig sind, den Datenzugriff in der Cloud zu ermöglichen. Da bei TFS ASAP Online auf das Entity Framework (EF) zugegriffen wird, waren die hier erwähnten Anpassungen nicht notwendig, weil das EF den Datenbankzugriff abstrahiert. Einzige Anpassung, die vorgenommen werden muss (bei der Migration von der lokalen Testmaschine) ist das Editieren der Applikationskonfigurationsdatei. Es wird betont, dass die Datensicherheit und -Vertraulichkeit von extremer Bedeutung sind.

Für die Migration der Geschäftslogikschicht wird angenommen, dass die Anwendung nach einer Service-orientierten Architektur aufgebaut ist und Technologien wie Business Processes und Workflows benutzen. Bei TFS ASAP ist das nicht der Fall, weil das lediglich ein Plugin ist, und nicht ein komplexes Anwendungssystem, wie die Beispielapplikation. Es wird eine Notwendigkeit für ein integriertes Modell einer Anwendung mit Komponenten und dieser Zuordnung zu entsprechenden Cloud-Diensten erkannt. So ein Modell könnte sich während der Migration von TFS ASAP als sehr hilfreich erweisen.

Die Autoren haben in *Cross-cutting Concerns* Herausforderungen erkannt, die nicht einer einzigen Schicht zugeordnet werden können, sondern für alle Schichten relevant sind. Der Einfluss der physischen und logischen Verteilung der Anwendung auf ihre Performanz wird betrachtet. Der rechtliche Aspekt ist besonders wichtig, wenn der Hosting-Ort gewählt wird. Da der Betrieb von VSO in der Region North East US stattfindet, mit dem TFS ASAP Online interagieren soll, sollen sich die zu migrierenden Komponenten möglichst nah befinden, damit eine bessere Leistungsfähigkeit des System erreicht werden kann. Die Autoren empfehlen, sich mit der Thematik Skalierbarkeit auseinanderzusetzen. Eine Entscheidung muss getroffen werden, was skaliert werden soll, in welchem Ausmaß, wie (vertikal oder horizontal - siehe Abschnitt 2.1.4) und wann. Das Azure Management Portal bietet eine Option an, Skalierungseinstellungen zu manipulieren. Die Skalierungslatenz soll berücksichtigt werden. Eine Worker-Role-Instanz bei TFS ASAP Online hochzufahren kann lange dauern, da jeweils das TFS Object Model installiert wird. Zu den Cross-cutting Concerns gehören auch Überlegungen bezüglich Mandantenfähigkeit, Migrations- und Betriebskosten, Quality of Service und Sicherheit. Für alle dieser Themen werden die Herausforderungen und die Auswirkungen auf die gesamte Applikation beschrieben. Damit wird eine Fülle von Problemen aufgelistet, die für eine Migration berücksichtigt und adressiert werden sollten.

Die Arbeit liefert die Zuordnung zwischen den erforderlichen Anpassungen für jeweilige Migrationsart und Anwendungsschicht. Darüber hinaus wird der Einfluss von Umsetzung eines Cross-cutting Concerns abhängig vom Migrationstyp eruiert. Es konnte festgestellt werden, das die Migrationstyp *Cloudify* den größten Anpassungsbedarf aufweisen kann, gleichzeitig jedoch die größten Vorteile mit sich bringen kann. Darüber hinaus soll der Aspekt Sicherheit berücksichtigt werden.

In [BLS11] schlagen die Autoren Cloud Motion Framework (CMotion), ein Rahmenwerk zur Cloud-Migration einer Applikation. Als Cloud-Migration wird dort eine Migration nicht nur als in dieser Diplomarbeit von On-Premise in die Cloud, als auch eine Migration zwischen unterschiedlichen Cloud-Betreibern oder -Diensten (von der Cloud in die Cloud) verstanden. Es wird von einer Composite Application ausgegangen, die sich aus einer Vielzahl zu migrierender Komponenten zusammensetzt. Somit entspricht die im Artikel angenommene Migration der in dieser Diplomarbeit verwendeten Migrationsart - *Cloudify*. CMotion braucht das Anwendungsmodell zur Ausführung einer Migration, in dem alle Komponenten samt ihrer externen Abhängigkeiten abgebildet sind. Die Komponenten, beginnend mit der höchsten Abstraktionsstufe, werden in Unterkomponenten zerlegt, bis in die Infrastrukturebene. Für sie werden Alternativen generiert, die Dienste unterschiedlicher Cloud-Anbieter repräsentieren. Als Hilfe zur Erstellung von Alternativen wird ein konzeptuelles Modell verwendet, in dem die Zusammenhänge zwischen folgenden Artefakten beschrieben werden:

- Technologien - Klassifizierung von Standards und Programmierschnittstellen in einer Baumform
- Komponenten - Anwendungsbausteine. Sie können eine Technologie bereitstellen, und somit ein Host für andere Komponenten sein, die diese Technologie erfordern. Sie sind eine Erweiterung einer Laufzeitumgebung.
- Laufzeitumgebungen - Sie stellen eine oder mehrere Technologien bereit und können auch Technologien erfordern. Im Gegensatz zu Komponenten, sind sie im Rahmenwerk vordefiniert.
- Adaptern - Transformieren Komponenten, die eine bestimmte Technologie erfordern, in eine Komponente die mit einer anderen Technologie konform ist. Adapter können automatisch oder manuell (Anpassung durch Menschen) sein.

Basierend auf den vorhandenen Technologien und Adaptern, generiert CMotion Alternativen für alle mögliche Kombinationen. Danach wird die beste Alternative basierend auf Kosten gewählt. Darauffolgend werden die Anwendungskomponenten in die gewählte Cloud-Umgebungen migriert. Zu diesem Zweck wird hier die *composite application framework (Cafe)* eingesetzt, die es ermöglicht, Anwendungsbeschreibungen zu modellieren, die bei beim Deployment benutzt werden können. Die Anwendungskomponenten werden gegen virtuelle Dienste entwickelt, die beim Ausliefern mit den wahren Diensten ersetzt werden, die gleiche Schnittstelle implementieren. Bei CMotion wurde diese Vorgehensweise erweitert, indem nun Schnittstellenanpassungen vorgenommen werden, damit Komponenten, die geforderte Schnittstelle nicht implementieren, auch angewendet werden können.

Der durch die Autoren beschriebene Ansatz ist vielversprechend, jedoch stand die Implementierung in der finalen Version für diese Arbeit nicht zur Verfügung. Aus diesen Gründen konnte im Rahmen dieser Diplomarbeit dieses Framework keine Verwendung finden. Außerdem wird in dem Framework für eine Cloud-Migration die Anforderung nicht berücksichtigt, die Anwendung um die Cloud-spezifische Merkmale wie Skalierbarkeit oder Mandantenfähigkeit zu erweitern.

In [VA12] untersuchen die Autoren die Entscheidungsfindung und die Methodologie für eine Cloud-Migration. Für die Entscheidungsfindung ist die Machbarkeit zu analysieren, wo diverse Fragen zu beantworten sind, wie zum Beispiel ob die zu migrierende Anwendung spezielle Anforderungen hat, die durch den Cloud-Zieldienst erfüllt werden können. Laut [VA12] kann eine Applikation entweder in IaaS oder PaaS migriert werden. Für beide Alternativen nennen die Autoren Anforderungen. Für IaaS sind das Aspekte wie die Daten- oder Dienstlokalisierung und spezielle Hardwareanforderungen. Für PaaS spielt zusätzlich die Kompatibilität der Anwendung mit der gewählten Plattform eine wichtige Rolle. Zu analysieren sind hier die Programmierspracheunterstützung, Datenbankunterstützung und anbieter-spezifische Anforderungen. Ein weiterer relevanter Faktor ist neben der Machbarkeitsanalyse die Kostenabschätzung. Die Kosten einer Cloud-Migration setzen sich aus den Kosten der eigentlichen Migration und der Kosten des Cloud-Verbrauchs zusammen. Es gilt zu prüfen, ob eine Neuimplementierung oder das Hosting der Applikation bei einem herkömmlichen Hosting-Anbieter nicht günstiger sind.

Die Kosten der eigentlichen Migration hängen davon ab, ob die Anwendung in IaaS oder in PaaS migriert wird. Für die erste Alternative sind die Migrationskosten sehr gering, da die Anwendung als ganzes migriert wird und es keinen zusätzlichen Anpassungen bedarf. Es können jedoch sog. versteckte Kosten auftreten, wenn die Skalierbarkeit gewährleistet werden soll. In solchem Fall muss eine zusätzliche Komponente entwickelt werden. Für PaaS sehen die Autoren die notwendigen Anwendungsanpassungen als den wichtigsten Kostentreiber. Es muss analysiert werden, was angepasst werden muss, welche Kosten damit verbunden sind und wie lange es dauert. Anpassungen werden durch Inkompatibilitäten verursacht, wie Programmiersprache, Datenbank, Kompatibilität externer Komponenten oder GUI. Diese Behauptungen konnten während der Migration von ASAP bestätigt werden, wobei weitere Kosten bei der Erweiterung entstanden. Zum Beispiel musste die GUI erweitert werden, damit die Mandantenfähigkeit erfüllt ist. Man kann sehen, dass die Migrationskosten im Fall PaaS größer sind als bei IaaS. Es existieren jedoch keine versteckten Kosten. In dem Artikel adressieren die Autoren das Problem der Datenbank-Migration und nennen mögliche Herausforderungen in diesem Umfeld. Ihre Behauptungen lassen sie mit drei Anwendungsfällen beweisen, in welchen Anwendungen migriert werden. Auf diese Weise konnte die Theorie mit der Praxis konfrontiert werden und man konnte sehen, welche Probleme bei einer Migration auftreten können.

3.2 Cloud-Migrationsmethodologien aus der Industrie

In diesem Kapitel werden Materialien von Cloud-Anbietern aus der Industrie vorgestellt und beschrieben. In dieser Arbeit wurden die Methodologien zur Cloud-Migration von Microsoft und Amazon untersucht. Basierend auf den beiden soll im weiteren Teil der Diplomarbeit eine Methodologie zur Migration in die Cloud von TFS ASAP abgeleitet werden.

3.2.1 Microsoft Azure

In [CTG+12] präsentiert Microsoft eine Methodologie zur Cloud-Migration für Microsoft Azure[1]. Das Dokument wurde 2012 veröffentlicht, daher ist der Inhalt nicht mehr in allen Details auf dem aktuellsten Stand. Auf den Internetseiten von Microsoft sind aktuelle Versionen einzelner Kapitel aus diesem Buch abrufbar. In [CTG+12] wird von einer daten-zentrischen Applikation ausgegangen, die in die Cloud migriert werden soll. Diese Annahme trifft für die im Rahmen dieser Diplomarbeit zu migrierende Software nicht zu, jedoch stellt die vorgestellte Methodologie trotzdem noch einen großen Wert dar, da daraus wertvolle Informationen für diese Diplomarbeit abgeleitet werden können.

Das Dokument besteht aus zwei Teilen. Im ersten Teil wird zuerst eine Methodologie für eine Migration in die Cloud dargestellt. Im weiteren Teil werden diverse Azure-Dienste beschrieben. Es werden Empfehlungen für die Auswahl von Azure-Diensten und für die Migration erwähnt. Darüber hinaus folgt Beschreibung von Werkzeugen, die als Hilfsmittel bei einer Cloud-Migration dienen. Der Leser wird oft auf Dokumente für die Entwicklung von Microsoft Azure Anwendungen verwiesen, die nicht unbedingt eine Cloud-Migration als Fokus haben.

Die im ersten Teil des Dokuments beschriebene Methodologie besteht aus fünf Phasen. Der Lebenszyklus soll agil und iterativ sein. Der ursprüngliche Plan ist während jeder Iteration anhand von gewonnenen Erkenntnissen zu verfeinern. Innerhalb einer Phase sind Aufgaben aufgelistet, die durchgeführt werden müssen.

Für die Durchführung der Migration wird zwischen Migration der Anwendung und der Migration der Datenbank unterschieden. Die können entweder parallel oder nacheinander durchgeführt werden.

- **1. Analysis Phase:** Hier wird die Migration geplant. Die funktionalen , nicht-funktionalen Anforderungen müssen erhoben werden. Die Anwendung muss analysiert werden und ein Plan muss erstellt werden, welche Anpassungen vorgenommen werden müssen, welche Anwendungsteile in welche Azure Dienste migriert werden und wie es sich auf die Architektur auswirkt. Ein Teil des Migrationsplanes sind auch die Kosten- und Aufwandschätzung und die Festlegung der Meilensteine mit Terminen. Es soll auch geplant werden, wie die Anwendung getestet und verwaltet wird.
- **2. Application Migration:** Hier wird die Migration der Anwendung durchgeführt. Es gibt zwei Möglichkeiten eine Anwendung in die Azure-Cloud zu migrieren: in eine Virtuelle Maschine oder in Azure-Dienste. Das entspricht der Migration des ganzen Stack und Cloudify. (siehe Abschnitt 2.1.5) In anderen Fällen wird auf eine Hybride Lösung in [DHN+12] verwiesen. Es wird empfohlen, zuerst ein Proof-of-Concept-Projekt durchzuführen, in dem die georderten Anpassungen implementiert werden und man die Anwendung in Microsoft Azure ausliefert. Die Applikation soll so entworfen werden, dass das Deployment auf mehreren Roles möglich ist, um eine künftige Skalierung zu ermöglichen.

[1] Microsoft Azure: http://azure.microsoft.com/

- **3. Data Migration:** Migration Der Datenschicht. Migrieren in eine Virtuelle Maschine mit installiertem SQL Server oder in Azure SQL Server bieten sich für relationale Daten an. Für nicht-relationale Daten werden Blobs und Tables empfohlen.
- **4. Optimization and Testing:** Testen, ob die Anwendung funktionale Anforderungen erfüllt. Testen der Anwendungsleistungsfähigkeit (Loadtest). Identifizierung und Behebung von Fehlern und Optimierung. Die Phase ist in der Methodologie nur erwähnt. Die Thematik befindet sich nicht im Fokus der Methodologie.
- **5. Operation and Management:** Überwachung der Applikation und Sammeln von diagnostischen Daten aus den laufenden Anwendungen mit Azure Diagnostics [Micd]. Die Daten helfen bei Debuggen, Fehlersuche und -behebung, Performanzmessung, Ressourcenüberwachung und Planen.

Das TFS ASAP Szenario ist nicht so typisch. Bei der Migration müssen neue Komponenten entwickelt werden, damit die durch TFS bereitgestellte Funktionalität nachgebaut werden kann. Wie in der beschriebenen Migrationsmethodologie muss man überlegen, welche Komponenten migriert werden müssen. Bei der Migration vom TFS ASAP muss man sich zusätzlich überlegen, welche Komponenten aus dem Anwendungskontext noch nachgebaut werden müssen. Das muss in der Planung berücksichtigt werden. Darüber hinaus ASAP Online wird TFS ASAP nicht ersetzen - sie sind zwei unabhängig existierende Systeme.

3.2.2 Amazon

Amazon Web Services[2], als weiterer Potentat auf dem Cloud-Computing-Markt und einer der Pioniere dieser Technologie stellt in [Var10] eine Methodologie zur Migration von bereits existierenden Anwendungen in die Cloud vor. Die darin enthaltene Methodologie berücksichtigt nur die Cloud-Dienste von Amazon Web Services (AWS) und aus diesem Grund ist sie für die Anwendung in dieser Diplomarbeit ungeeignet, denn die Verwendung von Microsoft Azure Diensten ist Voraussetzung dieser Diplomarbeit. Trotzdem wurde sie mit einbezogen, um die Sicht auf eine Cloud-Migration zu erweitern. Im Dokument wird eine phasen-basierte Strategie zum Migrieren einer Anwendung in eine Cloud-Umgebung mit allen Schritten, Techniken und Methodologien diskutiert. Der Erfolg der Migration hängt von drei Faktoren: der Anwendungsarchitekturkomplexität, des Anwendungskopplungsgrads und der Menge der einzusetzenden Ressourcen ab. Die Praxis hat gezeigt, dass die Durchführung eines Proof-Of-Concept-Projekts zum Migrationserfolg maßgebend beiträgt.

Die Cloud-Migration-Methodologie in [Var10] besteht aus sechs Phasen. Im Allgemeinen ist die Reihenfolge nicht wichtig, in welcher einzelne Phasen durchgeführt werden. Manche Phasen können auch übersprungen werden. Die Phasen sind:

- **1. Cloud Assessment Phase** In dieser Phase erfolgt die Kosten-, Anforderungen-, Sicherheits- und Architekturanalyse. In der aus der Sicht von TFS ASAP Online interessanten Architekturanalyse werden die Anwendungskomponenten und ihre Abhängigkeiten untersucht und für sie entsprechende Cloud-Dienste zugeordnet. Der Aufwand

[2]Amazon Web Services: http://aws.amazon.com

der Migration wird geschätzt, Erfolgskriterien werden definiert und ein Migrationsplan wird erstellt. In der Kostenanalyse werden nur die Betriebskosten berücksichtigt. Die in [BLS11] erwähnten Kosten der eigentlichen Migration, als Gegensatz zur einer Neuimplementierung, werden nicht thematisiert.

- **2. Proof of Concept Phase** Das Ziel dieser Phase ist es, die AWS kennenzulernen und sich in diese Technologie einzuarbeiten. Als Weg dazu bietet es sich an, eine einfache Anwendung in AWS zu deployen. Nächster Schritt ist die Durchführung eines Proof-Of-Concept-Projektes, das die im vorigen Schritt erarbeitete Architektur validieren und die kritische Funktionalität testen soll. Ein solches Projekt kann auch zu Steigung der Unterstützung und des Bewusstseins für die Cloud innerhalb der Organisation beitragen.

- **3. Date Migration Phase** In dieser Phase erfolgt die Auswahl von Cloud-Diensten zur Datenhaltung und die Datenmigration. Ähnlich wie in der Cloud-Migrationsmethodologie von [CTG+12], besteht die Migration den Anwendung aus separater Datenmigration und Anwendungsmigration. Da TFS ASAP über keine zu migrierenden Daten verfügt, wird dieser Aspekt vernachlässigt.

- **4. Application Migration Phase** Das Ziel dieser Phase ist es, die Anwendung zu migrieren. Es werden zwei Migrationsstrategien eingeführt: Gabelstapler-Migrationsstrategie, wo die ganze Anwendung auf einmal migriert wird, und Hybride Migrationsstrategie, wo einzelne Anwendungskomponenten separat über die Zeit migriert werden. Aufgrund der Charakteristik von TFS ASAP (Integration mit VSO als Cloud-Dienst) kommt nur die erste Strategie in Frage. Aus demselben Grund müssen verschieden Anpassungen an TFS ASAP vorgenommen werden, wie das Nachbauen von TFS-Bausteinen sowie die Sicherstellung der Mandantenfähigkeit, die in diesem Dokument nicht erwähnt werden. Allerdings ermöglicht diese Migrationsstrategie Cloud-Merkmale wie Skalierbarkeit nicht. Dies ist darauf zurückzuführen, dass die angewandten Dienste IaaS-Angebote sind. Mit Migration von TFS ASAP und den PaaS-Angeboten von Microsoft ist das nicht der Fall.

- **5. Leverage the Cloud Phase** In dieser Phase wird die bereits migrierte Anwendung so angepasst, dass sie die Merkmale und Vorteile der Cloud in Anspruch nehmen kann. Dabei geht es um unter anderem um die Skalierbarkeit und um Amazon-spezifische Dienste. Im Rahmen dieser Diplomarbeit kann angenommen werden, dass es sich dabei um die Angebote von Microsoft handelt. Es soll noch überlegt werden, die Sicherheit zu erhöhen, das Deployment zu automatisieren, eine Weise zu finden, Ressourcen zu kontrollieren und Möglichkeiten zu finden, die Dienstverfügbarkeit zu erhöhen.

- **6. Optimization Phase** In dieser Phase geht es um die Optimierung der Anwendung, und darum die Verbrauchskosten zu reduzieren. Dazu gehören Aufgaben wie Monitoring, Prüfung und Analyse der Ressourcenauslastung und Performanz und gegebenenfalls eine entsprechende Reaktion in Form von Re-Engineering-Anpassungen.

Die Methodologie von Amazon in [Var10] ist viel detaillierter als die von Microsoft in [CTG+12] und klarer strukturiert. Die einzelnen Schritte, Phasen und Aufgaben sind besser

geschrieben und hinterlassen keine Unklarheiten, somit kann man fast sofort mit einer Migration beginnen. Die Relevanz eines Prof-of-Concept wird betont. Darüber hinaus werden Aufgaben aufgelistet, deren Lösung mit AWS vertraut macht. Bei Amazon handelt sich jedoch um eine Migration in IaaS, deswegen wird dem Thema Anpassung der Anwendung wenig Aufmerksamkeit geschenkt, obwohl das Thema bei der Migration von TFS ASAP der ausschlaggebende Faktor ist.

Beide Methodologien bestehen jedoch fast aus den gleichen Schritten. Für diese Diplomarbeit wird eine Kombinationen der beiden benutzt, wo die relevantsten Teile übernommen werden. Die Beschreibung der Methodologie ist im Kapitel 4 zu finden.

3.3 Validierungs- und Evaluationsmethoden

In diesem Abschnitt wird auf den Kenntnisstand bezüglich der Evaluierung und ihrer Methoden eingegangen. Darauffolgend werden Metriken, welche für die Evaluation von Prozessen relevant sind, betrachtet.

3.3.1 Grundlagen der Evaluierung

Laut [Sto02] ist Evaluation die Beurteilung des Wertes eines Gegenstands, die nach systematischen Verfahren durchgeführt wird und mit datengestützten Belegen untermauert ist. In Rahmen einer Evaluierung werden *„empirische Methoden zur Informationsgewinnung und systematische Verfahren zur Informationsbewertung anhand offen gelegter Kriterien verwendet, die eine intersubjektive Nachprüfbarkeit möglich machen.*“Ein Evaluierungsgegenstand können Produkte, Prozesse, Projekte und andere sein. Im Kontext dieser Diplomarbeit sind lediglich Prozesse relevant. Ein Evaluierungsprozess soll einem bestimmten Zweck folgen und als Ziele eine Evaluation nennt der Autor die Gewinnung von Erkenntnissen, die Ausübung von Kontrolle, die Schaffung der Transparenz und die Dokumentation des Erfolgs. Die Ziele hängen meistens miteinander zusammen.

Es werden ebenfalls Aufgaben definiert, für die eine Evaluation benutzt werden kann. Das sind:

- Die Planung eines Prozesses zu verbessern
- Die Durchführungsprozesse zu beobachten (zB. Identifikation von Problemen und Prüfung, ob geplante Zeitabläufe eingehalten werden)
- Die Wirksamkeit und Nachhaltigkeit von Interventionen im Nachhinein zu bestimmen

Als weitere Aufgaben werden Überprüfung der Zielerreichung, Erfassung der Wirkungen, Bewertung der Teilaufgaben eines Prozesses bezüglich Richtigkeit und Relevanz, sowie Analyse der Kausalität von Wirkungen erwähnt. Die Qualität einer Evaluation lässt sich mit Kriterien wie Nützlichkeit, Durchführbarkeit, Korrektheit und Genauigkeit bestimmen. In [Sto02] wird noch zwischen internen und externen Evaluation unterschieden.

Aus der Sicht dieser Diplomarbeit bietet [Sto02] eine gute Einführung in die Evaluation an, mit Zielen und Aufgaben einer Evaluation. Das Thema wird generisch behandelt, und als Evaluationsgegenstände werden nicht nur Prozesse erwähnt, die für diese Diplomarbeit relevant sind, sondern auch Produkte, Programme und andere.

3.3.2 Evaluierung von Prozessen

In diesem Abschnitt werden die Methoden zur Evaluation von Prozessen vorgestellt und entsprechende Metriken betrachtet.

CMMI

Mit dem Qualitätsmanagementmodell Capability Maturity Model Integration (CMMI) lassen sich Aussagen über die Qualität eines Produkt-Entwicklungsprozesses treffen. Die Qualität eines Prozesses wird als Reife definiert. Mit CMMI kann man eine Verbesserung der Qualität eines Prozesses erzielen. Der Ausgangspunkt nach CMMI ist eine Begutachtung, in der die Vor- und Nach-Teile eines Prozesses identifiziert werden und daraufhin Maßnahmen, die die Vorteile ausbauen und Nachteile minimieren sollten. CMMI definiert Stufen (Reifegrade) für Prozesse. Den Reifegraden werden Kriterien zugeordnet. Basierend auf den Kriterien wird der Reifegrad eines Prozesses ermittelt. Die Qualität eines Prozesses kann verbessert werden, wenn er einen höheren Reifegrad erreicht, indem er die für diesen Reifegrad definierten Kriterien erfüllt. [Gec06]

Mit CMMI wird eine wohl-definierte und strukturierte Methode zur Ermittlung und Verbesserung der Prozessqualität dargestellt. Als Nachteile dieser Methoden können große Komplexität, die durch umfangreiche Natur der Methode und hohe Anzahl an Prozessen verursacht wird, sowie Mangel der Wartungs- und Änderungsverwaltungsfähigkeiten angesehen werden. [Gec06]

SPICE

Software Process Improvement Capability Determination (SPICE) ist eine generisches Modell zur Beurteilung der Qualität eines Softwareentwicklungsprozesses. Es besteht aus zwei Dimensionen: Prozess-Dimension und Reifegrad-Dimension, die jeweils unterschiedliche Perspektiven auf eine Prozess anbieten. Die Prozess-Dimension beinhaltet fünf Kategorien: Kunden-Lieferanten-Prozesse, Entwicklungsprozesse, Unterstützende, Prozesse, Managementprozesse und Organisationsprozesse. Sechs Reifegradstufen bilden die Reifegrad-Dimension: Unvollständig, Durchgeführt, Gesteuert, Definiert, Vorhersagbar und Optimierend. SPICE hat sich als eine aufwendige und unflexible Methode erwiesen. [Gec06]

ITIL CSI

ITIL[3] ist eine Sammlung von Empfehlungen zur Umsetzung und Betrieb vom IT Service Management (ITSM) und hat sich weltweit als ein De-Facto-Standard in diesem Bereich etabliert. Mit ITSM werden mehrwertbringende IT-Dienstleistungen den Kunden in Form von Diensten bereitgestellt. Die Organisation ist service- und kundenorientiert. ITIL besteht aus fünf Komponenten:

- Service Strategy (SS) - strategische und organisatorische Sicht zur Planung und Umsetzung des Service Management
- Service Design (SD) - Empfehlungen zum Entwurf und zur Entwicklung von Services
- Service Transition (ST) - Empfehlungen zum Wandel von neuen und modifizierten Services zu Operationen
- Service Operation (SO) - Empfehlungen zur effektiven und effizienten Diensterbringung
- Continual Service Improvement (CSI) - Empfehlungen zur kontinuierlichen Verbesserung der Dienste. Es kann als Zyklus SD -> ST -> SO -> SD beschrieben werden, wobei die SS in jedem Schritt berücksichtigt wird.

Für diese Diplomarbeit ist Continual Service Improvement (CSI) besonders wichtig. Der CSI-Prozess besteht aus sieben Schritten [CS11]:

- Festlegen, was gemessen werden soll. Starten des Projekts und Auswahl der Migrationsmethodologie.
- Festlegen, was gemessen werden kann. Auswahl der Metriken.
- Datensammlung
- Datenverarbeitung
- Datenanalyse
- Informationsableitung aus den gewonnen Daten
- Anwendung von korrektiven Maßnahmen

In dieser Diplomarbeit können aus den oben genannten Schritten folgende Aktionen abgeleitet werden:

- Auswahl der Methodologie, die evaluiert werden soll. Hier ist das die Kombination der Methodologien von Microsoft und Amazon.
- Auswahl von Metriken, die gemessen werden sollten.
- Implementierung der Evaluation und Datensammeln.
- Informationsgewinnung aus den Daten.

[3]ITIL: http://www.itilofficialsite.com

- Ergebnisanalyse und weitere Schritte, wie Präsentation von Ergebnissen, und Verbesserung der Methodologie.

3.3.3 Metriken für Prozesse

In diesem Abschnitt werden Metriken vorgestellt, die zu einer Prozessevaluierung eingesetzt werden können.

In [Bal98] wird die Kontrolle als *alle Management-Aktivitäten, die sicherstellen, dass die laufenden Tätigkeiten mit dem Plan übereinstimmen*, definiert. In Plänen werden die Produktanforderungen, Zeit und Kosten festgelegt. Prozessmodelle, Richtlinien, Methoden und Werkzeuge werden zur Ausführung vorgegeben. Kontrolle liefert Informationen darüber, wie gut ein Prozess ist. Die Ausführung wird in Rahmen der Kontrolle gemessen und gegebenenfalls werden korrektive Maßnahmen ergriffen. Zur Messung von Kenngrößen eines Software-Prozesses oder Software-Produktes dienen Metriken. Prozess-Metriken können in drei Gruppen differenziert werden [Bal98]:

- Fertigstellungszeit eines bestimmten Prozesses.
- Die aufzuwendenden Ressourcen (Personen, Mitarbeiter, Maschinen).
- Die Häufigkeit von Ereignissen wie Fehler oder die Anzahl von Anforderungen zu Änderungen.

Zusätzlich werden in [Bal98] Metriken erwähnt, die ein Unternehmen aus der Industrie zur Messung eines Prozesses anwendet. Die beziehen sich auf: Mitarbeiter/Zeit/Kosten, Umfang, Fehler, Kommunikation, Schwierigkeit und Wartung.

Weitere Metriken für Prozesse werden in [Kan04] und [Gal04] eingeführt.

In [Kan04] werden folgende Metriken für Prozesse erwähnt.

- Fehlerdichte während formalen Maschinentestens (Defect density during formal machine testing) : Nachdem der Quellcode ist ins System integriert.
  - Ist die Fehlerhäufigkeit größer oder kleiner als in der vorherige Periode?
  - Ist die Fehlerhäufigkeit größer oder kleiner als in der vorherige Release?
- Fehlerauftrittsmuster während des Testens (Defect arrival pattern during formal machine testing) Unterschiedliche Muster implizieren unterschiedlichen Qualitätsgrade
  - Wie viele Fehler auftreten während der Testphase pro Zeiteinheit?
  - In welchem Muster treten die Fehler auf? Wurde der Fehler gemeldet (echtes Fehlermuster)?
- Phasen-basiertes Fehlerbehebungsmuster (Phase-based defect removal pattern)
  - Die Fehlerverfolgung in allen Phasen des Softwareentwicklungsprozesses muss sichergestellt werden. Das kann mit Mitteln wie Design Reviews, Quellcode-Inspektionen und formale Verifizierung vor dem Testen erreicht werden.

Metrik	Beschreibung
NCE	Anzahl gefundener Fehler bei Code-Inspektionen und beim Testen.
KLOC	Anzahl während Entwicklung gefundener Entwicklungsfehler
WCE	Gewichtete Anzahl gefundener Fehler bei Code-Inspektionen und beim Testen.
WDE	Gewichtete Anzahl während Entwicklung gefundener Entwicklungsfehler
NFP	Anzahl von Function-Points.
MSOT	Rechtzeitig erreichte Meilensteine
MS	Anzahl Meilensteine
TCDAM	Summe der Verzögerungen für alle Meilensteine.
NYF	Anzahl von entdeckten Fehlern in einemWartungsjahr.
WYF	Gewichtete Anzahl von entdeckten Fehlern in einemWartungsjahr.
DevH	Entwicklerstunden
ReKLOC	Anzahl von tausenden wiederverwendeten Quellcode-Zeilen.
ReDoc	Anzahl wiederverwendeter Dokumentationsseiten.
NDoc	Dokumentationsseitenanzahl.

Tabelle 3.1: Basismetriken für den Prozess

- Fehlerbehebungseffizienz (Defect removal effectiveness - DRE)

$$DRE = \frac{RD}{LD} \times 100$$

  wobei:
  RD - Anzahl behobener Fehler in der Entwicklungsphase
  LD - Latente Fehler im Produkt

[Gal04] führt folgende Metriken ein, die entsprechend den Kategorien: Prozessqualität, Prozesszeitplan, Fehlerbehebungseffektivität und Prozessproduktivität zugeordnet werden. Die genannten Metriken werden von Basismetriken abgeleitet, die in der Tabelle 3.1 erklärt sind.

- Metriken für die Prozessqualität
  - Metriken für die Fehlerdichte

    Quellcodefehlerdichte

$$CED = \frac{NCE}{KLOC}$$

    Entwicklungsfehlerdichte

$$DED = \frac{NDE}{KLOC}$$

    Gewichtete Quellcodefehlerdichte

$$WCED = \frac{WCE}{KLOC}$$

Gewichtete Entwicklungsfehlerdichte

$$WDED = \frac{WDE}{KLOC}$$

Gewichtete Quellcodefehlerdichte pro Function-Point

$$WCEF = \frac{WCE}{NFP}$$

Gewichtete Entwicklungsfehlerdichte pro Function-Point

$$WDEF = \frac{WDE}{NFP}$$

– Metriken für den Fehlerschweregrad
Durchschnittlicher Quellcodefehlerschweregrad

$$ASCE = \frac{WCE}{NCE}$$

Durchschnittlicher Entwicklungsfehlerschweregrad

$$ASDE = \frac{WDE}{NDE}$$

– McCabe'sche zyklomatische Komplexität

$$V(G) = N_{\text{close-region}}$$
$$V(G) = N_{\text{edge}} - N_{\text{node}} + 2$$
$$V(G) = N_{\text{node-with-multiple-exit}} + 1$$

- Metriken für den Prozesszeitplan
Prozesszeitplan-Einhaltung

$$TTO = \frac{MSOT}{MS}$$

Durchschnittliche Meilenstein-Verzögerung

$$ADMC = \frac{TCDAM}{MS}$$

- Metriken für die Fehlerbehebungseffektivität

Effektivität der Entwicklungsfehlerbehebung

$$DERE = \frac{NDE}{NDE + NYF}$$

Gewichtete Effektivität der Entwicklungsfehlerbehebung

$$DWERE = \frac{WDE}{WDE + WYF}$$

- Metriken für die Prozessproduktivität

Entwicklungsproduktivität

$$DevP = \frac{DevH}{KLOC}$$

Funtion-Point Entwicklungsproduktivität

$$FDevP = \frac{DevH}{NFP}$$

Quellcode-Wiederverwendbarkeit

$$CRe = \frac{ReKLOC}{KLOC}$$

Dokumentationswiederverwendbarkeit

$$DocRe = \frac{ReDoc}{NDoc}$$

Für diese Diplomarbeit wurden Metriken wie Prozesszeitplan-Einhaltung (engl. Time Table Observance) (TTO), Durchschnittliche Meilenstein-Verzögerung (engl. Average Delay of Milestone Completion) (ADMC), Umsetzungszeit und Ereignishäufigkeit verwendet. Sie wurden ausgewählt, weil ihre Erhebung relativ einfach war, vor allem angesichts der Tatsache, dass mit der Migration von TFS ASAP schnell begonnen werden musste und nicht von Anfang an Daten aufgezeichnet werden konnten.

4 Methodologie für die Cloud-Migration

In diesem Kapitel wird die Methodologie vorgestellt, die bei der Migration von TFS ASAP in die Cloud eingesetzt wird. Die Methodologie basiert auf den Methodologien von Microsoft [CTG+12] und Amazon [Var10]. Die beiden Methodologien sind ähnlich, haben ähnliche Phasen und unterscheiden sich nur in Einzelheiten. Diese Vorgehensweise wurde gewählt, um die bestmögliche Lösung für die Migration von TFS ASAP zu verwenden. Die Phasen der beiden Migrationsmethodologien werden in den Abbildungen 4.1 und 4.2 dargestellt. In folgenden Abschnitten erfolgt die Beschreibung der einzelnen Phasen der Methodologie sowie der Aufgaben, die in Rahmen jeweiliger Phase zu erledigen sind.

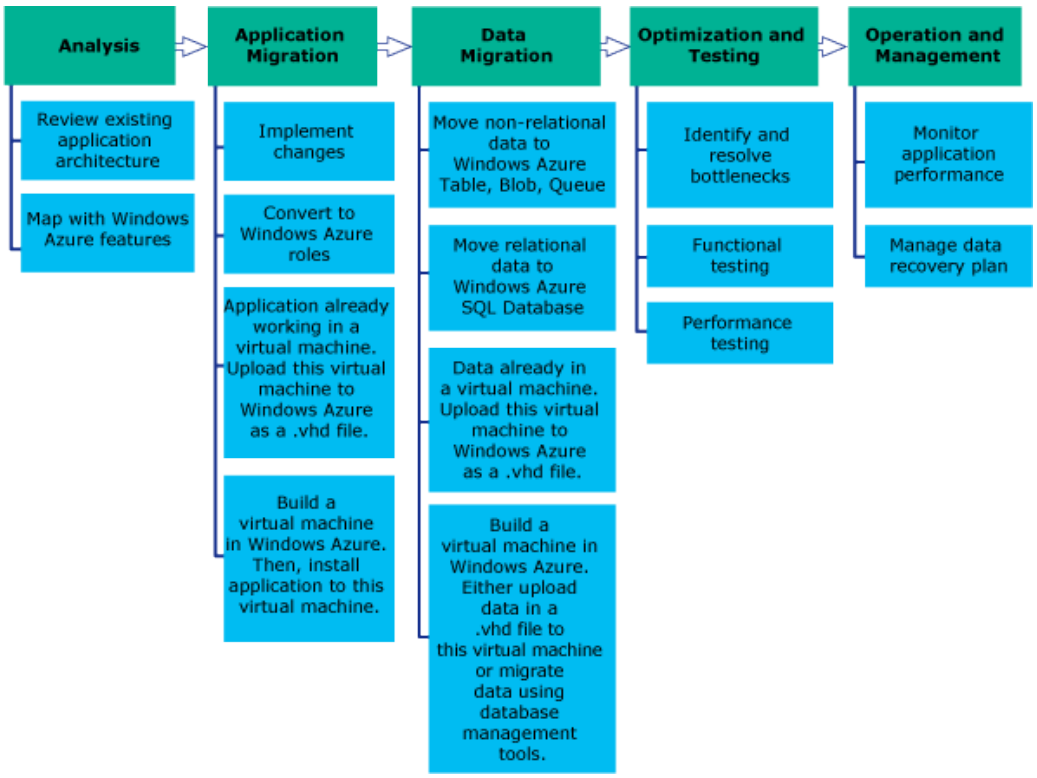

Abbildung 4.1: Phasen der Micosoft Methodologie für die Migration datenzentrischen Anwendungen [CTG+12]

Die Cloud-Migration-Methodologie in dieser Diplomarbeit besteht aus folgenden Phasen:

- **1. Analyse** (*Analysis*-Phase bei Microsoft und *Cloud Assessment Phase* bei Amazon)

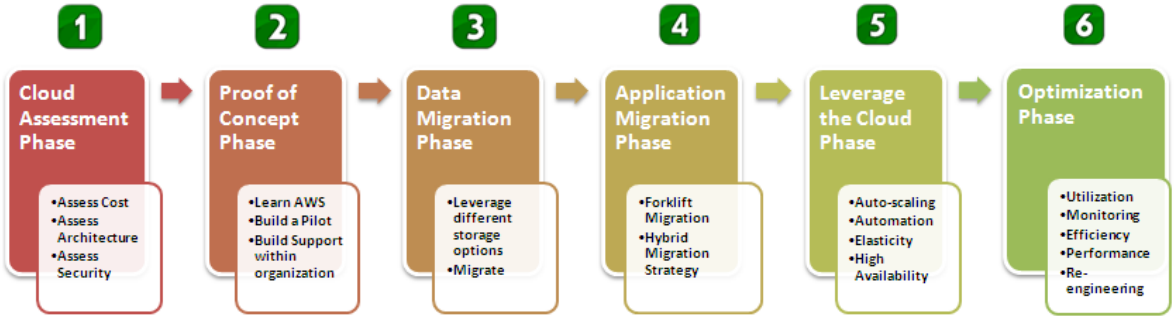

Abbildung 4.2: Phasen der Amazon Methodologie für die Migration [Var10]

- **2. Proof of Concept** (wird in *Analysis*-Phase bei Microsoft genannt, bei Amazon als *Proof of Concept Phase* explizit erwähnt)
- **3. Migration** (*Application Migration*-Phase bei Microsoft und *Application Migration Phase* bei Amazon)
- **4. Optimization and Testing** (*Optimization and Testing*-Phase bei Microsoft, *Leverage the Cloud Phase* und *Optimization Phase* bei Amazon)
- **5. Operation and Management** (*Operation and Management*-Phase bei Microsoft, *Leverage the Cloud Phase* und *Optimization Phase* bei Amazon)

Sie werden in den folgenden Abschnitten ausführlicher beschrieben.

Der größte Unterschied zu den oben genannten Methologien ist das Fehlen der Phase *Daten-Migration*. Der Grund dafür ist, dass TFS ASAP über keine zu migrierenden Daten verfügt. Das Datenmodell in TFS ASAP Online wird für Benutzerverwaltung, Mandantenfähigkeit und Abbildung der Beziehungen zwischen Geschäftsobjekten wie *VSO-Konto*, *ProjectCollection*, *Project* verwendet. Sonst können die einzelnen Phasen der in diesem Kapitel präsentierten Methodologie Phasen der Methodologien von Microsoft und Amazon zugeordnet werden. Die Phasen der Microsoft-Methodologie und Amazon-Methodologie sind oben in der Phasenliste in Klammern nach der Phase aufgeführt, mit der sie in Verbindung stehen.

4.1 Analyse und Planung

Das Ziel dieser Phase ist es, die zu migrierende Anwendung zu analysieren, die notwendigen Anpassungen festzulegen und einen Migrationsplan zu erstellen. Die genauen Aufgaben für diese Phase werden in den folgenden Abschnitten näher vorgestellt. Diese einzelnen Aufgaben müssen nicht in einer bestimmten Reihenfolge erledigt werden, sondern können parallel abgearbeitet werden. Dabei ist zu beachten das die Aufgaben in Beziehung zueinander stehen und sich somit untereinander beeinflussen können. Das Ergebnis der Analyse und Planung-Phase ist der Migrationsplan und die an die Cloud angepasste Anwendungsarchitektur.

Die Aufgaben dieser Phase sind:

4.1.1 Erhebung der Anforderungen

Diese Aufgabe kommt in der Migration von Microsoft [CTG+12] vor. Inhalt ist die Geschäftsziele einer Migration zu identifizieren. Dabei kann die Beantwortung folgender Fragen hilfreich sein:

- Werden mit der migrierten Anwendung neue Kunden angesprochen?
- Wird die Mandantenfähigkeit gefordert, um eine Mehrzahl von geographisch verteilten Benutzern zu unterstützen?
- Welche Dienste eignen sich mehr für die Applikation: Azure Virtual Machines oder Azure Cloud Services?

  Ein Prototyp der Benutzerschnittstelle sollte erstellt werden, der die durch die Software zu realisierende Geschäftsszenarien darstellt. Basierend auf dem Prototypen ist ein Anforderungsdokument zu erstellen, das die Beschreibung, die Anforderungen an das Produkt, sowie die Rahmenbedingungen beinhaltet.

Die Anforderungen sind zunächst nur grob zu erfassen und werden während dieser Phase noch verfeinert, wenn neue Erkenntnisse gewonnen werden.

4.1.2 Erkundung der Microsoft Azure Technologie

Im Rahmen dieser Aufgabe soll einen Überblick über die bei Microsoft Azure bereitgestellten Angebote verschaffte werden. Die Azure Dienste und ihre Vorteile und Nachteile sollten kennen gelernt und verstanden werden. Bei der Aufgabe liegt der Fokus sowohl auf den Grundlagen der Azure-Plattform, als auch auf die Eignung einzelner Dienste für die Migration von TFS ASAP.

4.1.3 Kostenschätzung

In dieser Aufgabe werden auch die Kosten grob geschätzt, die während des Betriebs der Anwendung und Inanspruchnahme von diversen Diensten der Microsoft Azure Plattform entstehen. Zu diesem Zweck sollte man sich mit dem Abrechnungsmodell von Microsoft Azure vertraut machen. In diese Überlegungen sollen die Kosten für die Entwicklungs- und Test-Umgebung mit einbezogen werden. Hier kann der *Azure Pricing Calculator*[1] benutzt werden.

[1] Azure Pricing Calculator: http://azure.microsoft.com/de-de/pricing/calculator/

4.1.4 Anwendungsanalyse

In Rahmen dieser Aufgabe wird der Anwendungsaufbau in Hinsicht auf die künftige Migration analysiert. Im Idealfall ist eine Dokumentation vorhanden, aus welcher hervorgeht, wie die Applikation aufgebaut ist, aus welchen Komponenten sie besteht, in welcher Beziehung die Komponenten zueinander stehen und welche externen Abhängigkeiten die Komponenten aufweisen. Wenn eine solche Dokumentation zur Beantwortung dieser Fragen nicht zur Verfügung steht, dann muss sie im Rahmen dieser Aufgabe erstellt werden. Zusätzlich muss auch darüber nachgedacht werden, welche Komponenten übernommen werden können, welche angepasst werden müssen, für welche Komponenten eine Migration ausgeschlossen ist und für welche Komponenten eine Migration keinen Sinn macht. Eigenschaftsunstimmigkeiten zwischen einer On-Premise-Lösung und einer Cloud-Lösung sind zu identifizieren. Die Funktionsweise der zu migrierenden Anwendung muss verstanden werden. Im Kontext der Migration von TFS ASAP ist wichtig, dass eine gemeinsame Code-Basis zwischen TFS ASAP und TFS ASAP Online existiert.

Die Erkenntnisse dieser Aufgabe sollen in die Anforderungen einfließen.

4.1.5 Redesign

In dieser Aufgabe werden aus den im vorigen Teil gewonnenen Erkenntnissen die für die Migration in die Cloud notwendigen Anpassungen abgeleitet. Eine Zuordnung der Komponenten zu Microsoft Azure Diensten erfolgt in diesem Schritt. Für die Komponenten, die nicht migriert werden, werden neue Komponenten mit Hilfe von Microsoft Azure Diensten nachgebildet. Dies gilt auch für die externen Abhängigkeiten der Komponenten. In diesem Fall sind das Elemente vom TFS. In diesem Teil sind auch diejenigen Komponenten zu entwerfen, die diverse Cloud-Eigenschaften sicherstellen, wie zum Beispiel Mandantenfähigkeit. Während der vorige Schritt den IST-Zustand zum Ergebnis hatte ist hier der SOLL-Zustand das Ergebnis.

In diesem Schritt werden auch Funktionalitäten identifiziert, die die Cloud-Version von TFS ASAP aufweisen soll, um die Funktionsweise von TFS ASAP zu erfüllen. Diese Funktionalitäten sollten prototypisch im Rahmen des Proof-of-Concept-Projektes implementiert werden, damit die Machbarkeit der Migration sichergestellt werden kann.

4.1.6 Migrationsplan

In dieser Aufgabe wird der notwendige Aufwand zur Durchführung der Migration geschätzt. Es werden die Zeiten für alle Tätigkeiten die im Migrationsplan vorkommen, abgeschätzt. Die Tätigkeiten werden im Laufe dieser Phase abgeleitet und setzen sich aus Aufgaben zusammen, die zur Migration in die Cloud notwendig sind.

Ziel dieser Phase ist einen Plan für die Cloud-Migration zu erstellen und Meilensteine zu definieren. Der Plan soll die in den vorhergehenden Schnitten abgeleiteten Aufgaben zur Umsetzung von Anforderungen mit Realisierungsterminen und geschätzten Aufwänden für

sie beinhalten. Die Aufgaben sind priorisiert. Schon in diesem Schritt sollen die Performanz- und Skalierbarkeits-Tests in Erwägung gezogen und im Migrationsplan mit berücksichtigt werden. Dies gilt ebenfalls für die Planung des ganzheitlichen Anwendungslebenszyklus und Anwendungsverwaltung. In der Organisation sollten ebenfalls Lernkosten einkalkuliert werden.

4.2 Proof of Concept

Diese Phase wurde in der Methodologie von Amazon in [Var10] explizit als separate Phase definiert, während ein Proof-Of-Concept-Projekt in der Methodologie von Microsoft [CTG+12] lediglich als ein Teil der Analyse-Phase nur beiläufig empfohlen wird. Für die Anforderungen dieser Diplomarbeit ist es jedoch eine wichtige Aufgabe aufgrund des geringen Know-Hows im Kontext von Microsoft Azure.

Mit dieser Phase soll die Machbarkeit der Cloud-Migration der gewählten Software nachgewiesen werden. Die in der Analyse-Phase abgeleiteten kritischen Funktionalitäten sollten prototypisch implementiert werden und auf der Microsoft Azure Plattform ausgeliefert werden, um die gewählte Lösung auf die Richtigkeit zu prüfen.

Eine nebenläufige Aufgabe dieser Phase ist erste Erfahrungen mit Microsoft Azure als Technologie zu machen und die wichtigsten Dienste kennen und einsetzen zu lernen.

4.3 Anwendungsmigration

In dieser Phase erfolgt die Durchführung der Migration in die Cloud. Die Architekturanpassungen und Codeänderungen, die in der Analyse-Phase festgelegt wurden und in in Proof-Of-Concept-Phase erfolgreich validiert wurden, werden nun in das Produkt einfließen. Es wird die zusätzliche, notwendige Infrastruktur nachgebaut, die für das gewählte Szenario nicht out of the Box zur Verfügung gestellt werden kann. Darüber hinaus werden Module implementiert, durch welche die Cloud-Computing Merkmale wie Mandantefähigkeit ermöglicht werden. Abhängig von der in der früheren Phase getroffenen Entscheidung werden die Anwendungskomponenten entweder in Azure Roles konvertiert oder in einer Azure Virtual Machine ausgeliefert. Wenn man Code-Anpassungen vornimmt, sollte man auch Themen wie Performanz, Skalierbarkeit und Sicherheit im Hinterkopf behalten.

4.4 Optimierung und Testen

In dieser Phase muss die Anwendung, nach einer erfolgreich durchgeführten Migration, getestet und optimiert werden. Es gilt zu prüfen, ob die Anwendung die Anforderungen erfüllt. Zur Prüfung der Performanz der Anwendung müssen Load-Tests durchgeführt werden. Außerdem ist zu überprüfen und sicherzustellen, dass die Anwendung gemäß der

funktionalen und nicht-funktionalen Anforderungen verhält. Alle entdeckte Fehler müssen behoben werden. Sobald dies erledigt ist, sollten Probleme, die sich auf ihre Features, Funktionalität, Performanz und Skalierbarkeit beziehen, adressiert und behoben werden. Zu beachten ist, dass Testen und Optimieren bereits in der frühen Analyse-Phase berücksichtigt werden sollten.

4.5 Betrieb und Verwaltung

In dieser Phase sollte ein Monitoring-Mechanismus implementiert werden, der das Sammeln von Daten ermöglicht. Die Auswertung dieser Daten kann helfen, die Anwendung zu überwachen und gegebenenfalls zu optimieren. Dazu bietet sich Windows Azure Diagnostics[2] an. Mit Hilfe der Technologie wird ermöglicht, von einer auf Windows Azure laufenden Applikation diverse diagnostische Daten zu beziehen. Auf Basis dieser Daten kann man die Anwendung debuggen und Fehler identifizieren, die Leistungsfähigkeit und Ressourcenverbrauch messen, den Netzwerkdatenverkehr analysieren, um den Kapazitätseinsatz zu planen oder die Daten einer externen Prüfungsstelle zur Verfügung zu stellen.

Ähnlich auch wie in den herkömmlichen Software-Prozessen, werden alle aufgetretene Probleme in dieser Phase werden auch adressiert und gelöst. An das System können auch neue Anforderungen gestellt werden, die dann auch in weiteren Entwicklungsprojekten umgesetzt werden sollten[LL07].

4.6 Erhebung von Metriken

In diesem Abschnitt wird erklärt, wie der in den früheren Abschnitten vorgestellte Cloud-Migration-Prozess gemessen und bewertet wird. Zu diesem Zweck werden die Schritte von der Vorgehensweise von ITIL CSI benutzt, die in dem Abschnitt 3.3.2 vorgestellt wurden.

Um die ausgewählte Cloud-Migration-Methodologie zu bewerten und zu verbessern, werden Daten gesammelt. Für alle Phasen der Migration werden folgenden Kenngrößen erhoben:

- Zeit - Wie lange sollte die Durchführung der Phase nach dem Plan dauern? Wie lange hat die Durchführung einer Phase gedauert? Kam es zu Verzögerungen?
- Ereignis - Ereignis kann zwei Ausprägungen haben: Problem und Rücksprung. Als Problem wird ein Ereignis definiert, das während der Durchführung einer Phase aufgetreten ist und den Fortschritt maßgeblich hindert. Dies kann eine Abweichung von der Spezifikation oder ein Fehler in der Spezifikation sein [Bal98]. Rücksprung tritt dann auf, wenn man die Ergebnisse der früheren Phasen ändern muss. Ein Beispiel wäre hier die Änderung einer Designentscheidung. Es wird sowohl die Anzahl von Ereignissen erfasst, als auch ihre Dichte in einer Zeiteinheit.

[2] Aktivieren der Diagnose in Azure Cloud Services und Virtual Machines: http://azure.microsoft.com/de-de/documentation/articles/cloud-services-dotnet-diagnostics/

- Lernbarkeit - Waren Vorkenntnisse zur Durchführung der Phase notwendig? Wie einfach war diese Phase zu lernen?
- Schwierigkeitsgrad - Wie einfach war diese Phase durchzuführen?
- Verständlichkeit - Wie einfach konnte diese Phase verstanden werden?
- Zufriedenheitsgrad - Wie zufrieden war man mit der Phase im Nachhinein?
- Meilensteine - Die Metriken TTO und Durchschnittliche Meilenstein-Verzögerung ADMC aus dem Abschnitt 3.3.3 werden eingesetzt.

Das Datenformat, welches zur Erhebung der Daten basierend auf den aufgeführten Metriken für jede Phase verwendet wird, ist in der Tabelle 4.1 dargestellt. Die Basiszeiteinheit ist die Kalenderwoche (KW). Eine KW besteht aus fünf Kalendertag (KT). Für einen KT wird von einem Dauer von 5 Stunden ausgegangen.

Metrik	**Eigenschaft**
Geplante Zeit [KW]	_____
Umsetzungszeit [KW]	_____
Verzögerung [KW]	_____
Anzahl von Ereignissen/-Problemen/Rücksprüngen	_____
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	_____
Anzahl der Meilensteine	_____
TTO	_____
ADMC [Meilenstein/KW]	_____
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: _____
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: _____
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: _____
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: _____

Tabelle 4.1: Format für die Datenerhebung in jeder Phase

Zur Erfassung der Daten für Ereignisse, wird ein Datenformat eingesetzt, das in der Tabelle 4.2 dargestellt ist. ID ist ein Feld, das das Ereignis eindeutig identifiziert. ID setzt sich aus der Phasen-Nummer, in der das Ereignis eintritt, einem Bindestrich, einem der zwei Buchstaben (P für Problem, Z für Rücksprung) und einer Nummer. Zum Beispiel ein Ereignis mit ID 3-Z3 trat in der Phase 3 ein, seine Klasse ist Rücksprung und es ist das dritte Ereignis dieser Klasse in dieser Phase. Ein zusätzliches Feld zur Identifizierung von Ereignissen ist das Feld Name, das einen benutzerfreundlichen Namen eines Ereignisses darstellt. Das Feld Klasse

gibt an, zu welcher Ereignis-Klasse das Ereignis gehört. In der Beschreibung erfolgt eine textuelle Charakterisierung des Ereignisses. Das Feld Lösung beinhaltet eine ausführliche Beschreibung der gefundenen Lösung. Falls irgendwelche Anpassungen an der Migration oder an dem Prozess selbst vorzunehmen sind, wird das im Feld Anpassungen ausgeführt. Das Feld Kommentar bietet eine Möglichkeit an, weitere Informationen zu erfassen, für die keines der oben genannten Felder geeignet scheint.

Eigenschaft	**Wert**
ID	
Name	
Klasse	
Beschreibung	
Lösung	
Anpassung	
Kommentar	

Tabelle 4.2: Format zur Ereigniserfassung

5 Anwendung der Methodologie an TFS ASAP

In diesem Kapitel wird die in den früheren Teilen dieser Diplomarbeit entwickelte Cloud-Migrations-Methodologie angewendet. In den folgenden Abschnitten wird die Durchführung der jeweiligen Phasen der Methodologie samt ihrer Aufgaben beschrieben.

5.1 Analyse und Planung

In diesem Abschnitt erfolgt die Beschreibung der Tätigkeiten, die im Rahmen der Phase *Analyse und Planung* aus dem Kapitel 4 vorkommen. Auf die Tätigkeiten wird in den folgenden Unterabschnitten eingegangen.

5.1.1 Erhebung der Anforderungen

Ziel dieser Aufgabe ist, die Anforderungen an TFS ASAP Online zu erheben. Als erste Teilaufgabe wurde ein Prototyp für die Benutzerschnittstelle von TFS ASAP Online erstellt. Dazu diente Microsoft PowerPoint[1] und ihre Funktionalität *Storyboarding*[2]. Mit *Storyboarding* lassen sich unter anderem Szenarien für die Interaktion mit dem Benutzer einer Anwendung visuell und prototypisch darstellen. Das Ziel war dabei die Erstellung einer Spezifikation für die Benutzerschnittstelle von TFS ASAP Online. Man sollte sich an der bestehenden On-Premise-Version von TFS ASAP orientieren und möglichst viele Teile der Implementierung übernehmen. Aus dem Prototypen konnten die ersten Anforderungen abgeleitet werden. Zusätzlich erforderte diese Aufgabe, sich mit TFS ASAP und seiner Funktionsweise vertraut zu machen, was zu einem besseren Verständnis beigetragen hat.

In dem Prototypen wurden folgende Szenarien aus Kundensicht berücksichtigt:

- Kundensicht
  - Registrierung bei TFS ASAP Online mit Microsoft Account
  - visualstudio.com Accounts hinzufügen
  - Regeln auf der gesamten Collection aktivieren
  - Regeln auf einzelnen Projekten aktivieren
- Verwaltungssicht

[1] Microsoft PowerPoint: http://office.microsoft.com/de-de/powerpoint/
[2] Storyboarding von Ideen mit PowerPoint: http://msdn.microsoft.com/de-de/library/hh409276.aspx

  - Liste der visualstudio.com Accounts
  - Kontaktdaten
- optional Verwaltungssicht für den Kunden
  - Verbrauch
  - Guthaben

Der Prototyp [Sab14c] bestand aus ca. 30 Folien, in welchen der Interaktionsfluss und Basisszenarien dargestellt wurden. Er erhob keinen Anspruch auf Vollständigkeit und die Benutzeroberfläche unterlag im Laufe der Migration vielen Veränderungen im Vergleich zu dem Prototypen. Der Prototyp wurde auch Brian Harry[3] aus Microsoft geschickt, um seine Meinung zum Produkt zu erfragen. Die beispielhaften Folien aus dem Prototypen sind in den Abbildungen 5.1 und 5.2 zu sehen.

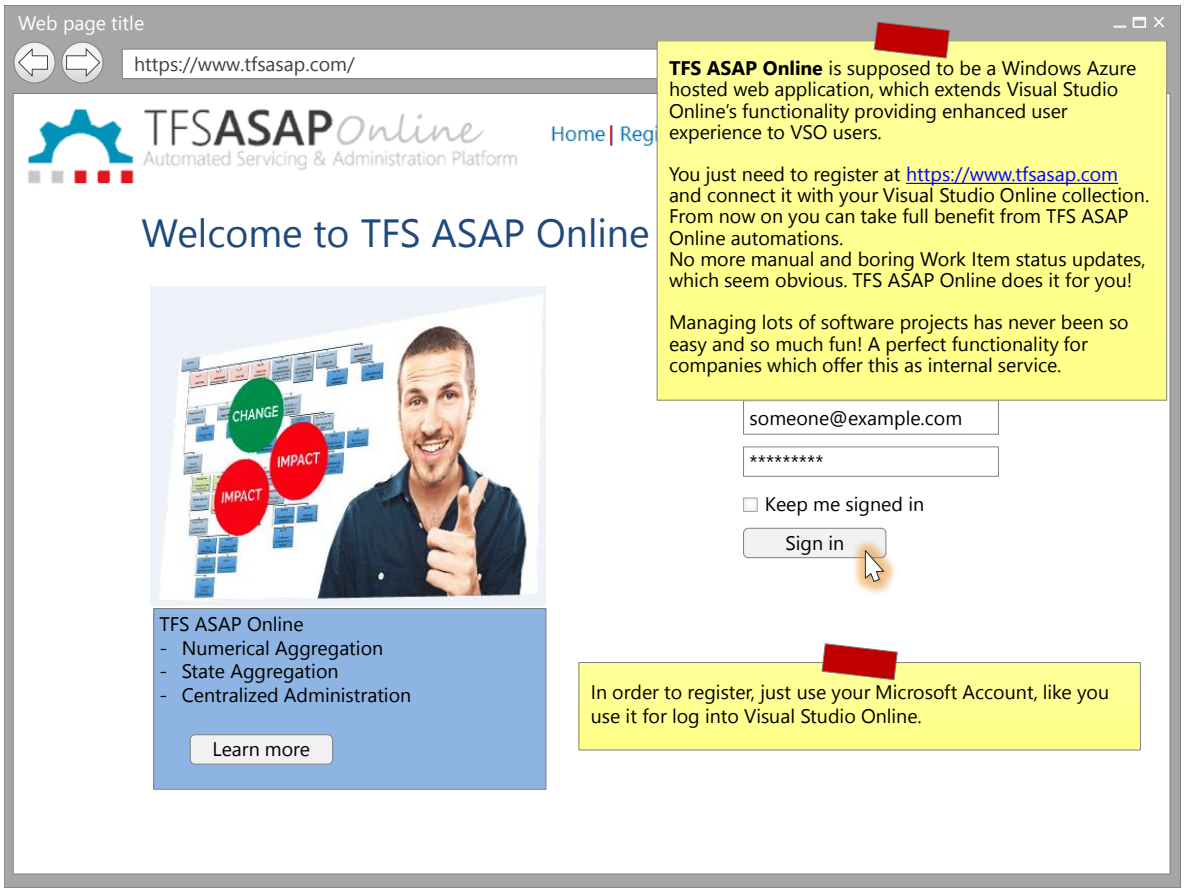

Abbildung 5.1: TFS ASAP Online Prototyp: Landing Page [Sab14c]

Die Anforderungen wurden basierend auf den Erkenntnissen bei der Erstellung vom Prototypen, Gesprächen mit dem Betreuer aus der Industrie sowie anderen Anwendern von des TFS ASAP, und der Analyse der Architektur aus dem Abschnitt 5.1.4 abgeleitet. Zur Erfassung der

[3]Brian Harry's Blog: http://blogs.msdn.com/b/bharry/

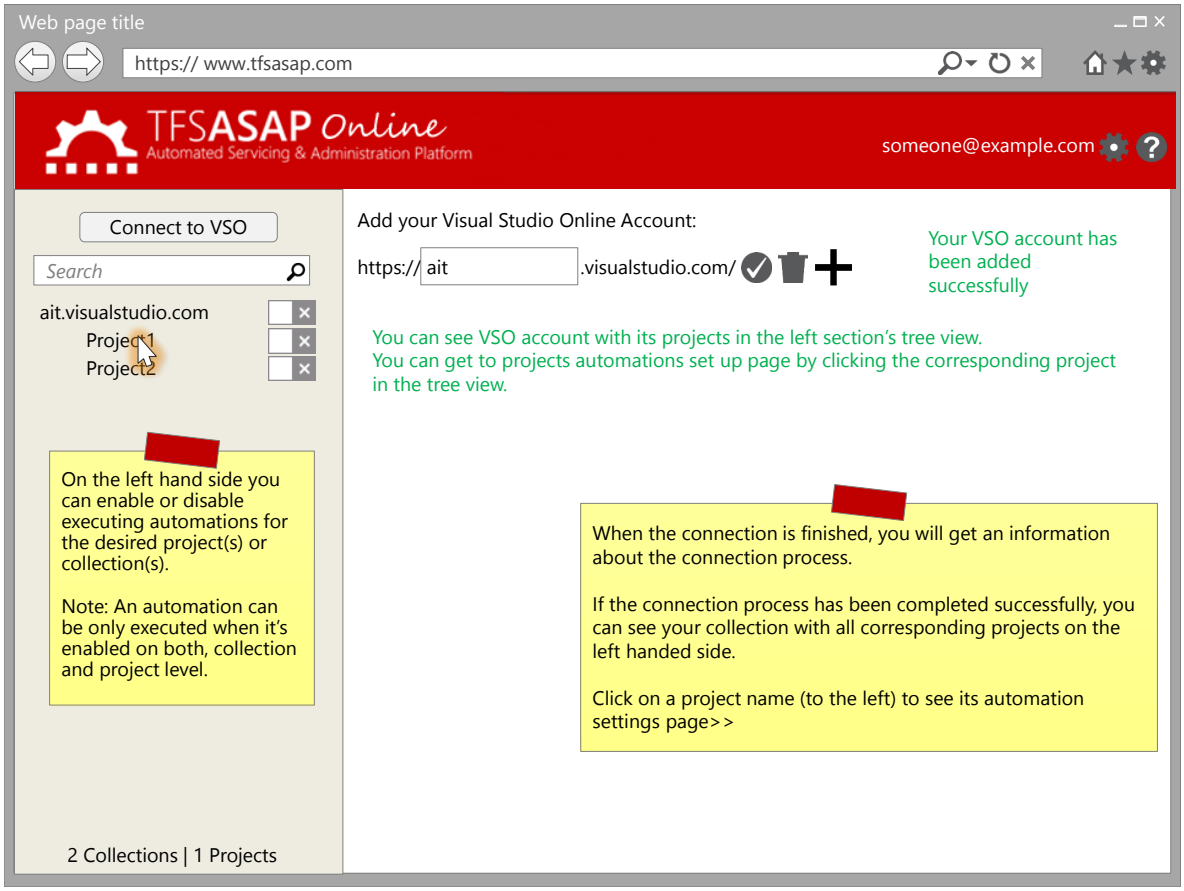

Abbildung 5.2: TFS ASAP Online Prototyp: Nach dem Hinzufügen von einer VSO Project Collection [Sab14c]

Anforderungen wurde das kostenlose Werkzeug der Firma AIT eingesetzt - AIT WordToTFS[4]. Dieses Werkzeug ist ein Plug-In für Microsoft Word[5]. Mit Microsoft Word mit installiertem AIT WordToTFS lassen sich Anforderungen in einem Word-Dokument erfassen. Sie können dann auf eine einfache Art und Weise in TFS oder VSO als *Work Items* importiert und synchronisiert werden. Die aus dem Anforderungsdokument [Sab14d] generierte *Work Items* dienten als Ausgangspunkt zur Planung der Aufgaben und zur Fortschrittsüberwachung während der Migration von TFS ASAP.

Das Anforderungsdokument beginnt mit der Beschreibung der Ausgangssituation und Ziels der Migration. Diese Beschreibung ähnelt in ihrem Inhalt den Abschnitten 1.1 und 1.2 und aus diesem Grund wird diese hier nicht näher betrachtet. Weiterhin werden die Zielgruppe und Rahmenbedingungen für TFS ASAP Online identifiziert. Als Nutzer der Anwendung werden TFS ASAP Online Nutzer und TFS ASAP Online Administratoren erwähnt. Aus den im Dokument beschriebenen Anwendungsfällen, werden funktionale und nicht-funktionale Anforderungen abgeleitet. Unter den nicht-funktionalen Anforderungen werden Aspekte wie Mandantenfähigkeit, Skalierbarkeit und Performanz berücksichtigt. Die Anforderungen

[4]AIT WordToTFS: http://www.aitgmbh.de/?id=222
[5]Microsoft Word: http://office.microsoft.com/de-de/word/

werden als Features zusammengefasst und die Features machen die Arbeitspakete aus, die während der Migration durchgeführt werden müssen. Das Dokument beinhaltet auch den ersten Entwurf für die Zielarchitektur der Anwendung. Im Glossar werden Begriffe und ihre Beschreibungen aufgelistet.

5.1.2 Erkundung der Microsoft Azure Technologie

Diese Aufgabe bestand darin, sich mit der Technologie von Microsoft Azure vertraut zu machen und sie besser kennen zu lernen. Um dies zu bewerkstelligen, ist man der Recherchearbeit nachgekommen. Man hat die Grundlagen kennen gelernt, die dazu notwendig sind, Microsoft Azure einzusetzen. Die Funktionsweise, Vorteile und Nachteile von den Basis-Diensten und komplexeren Diensten wurden erkundet. Die Mechanismen zur Auslieferung und die Verwaltung von Microsoft Azure Diensten wurden auch erforscht. Die Teile der Ergebnisse dieser Aufgabe befinden sich in dem Abschnitt 2.7.

5.1.3 Kostenschätzung

Bei der Kostenschätzung ging man davon aus, dass für die Entwicklung und Betrieb von TFS ASAP Online die im Rahmen der Microsoft Azure Subscriptions für AIT bereitgestellten Volumen ausreichen und nicht überschritten werden können. In der Tabelle 5.1 ist die Kostenabschätzung für TFS ASAP Online zu sehen. Alle von TFS ASAP Online eingesetzten Dienste werden samt ihrer Skalierungsdetails und geschätzten maximalen monatlichen Kosten aufgelistet. Aus der Tabelle wurden maximale monatliche Kosten für die Anwendung abgeleitet. Es ergaben sich ca. 102 Euro.

Dienst	Skalierung	Maximale Kosten
Azure Web Role	2-3 sehr kleine Instanzen, Abhängig von der CPU-Auslastung	3x12 EUR/Monat = 36 EUR/Monat
Azure Worker Role	2-5 sehr kleine Instanzen, Abhängig von der Queue	5x12 EUR/Monat = 60 EUR/Monat
Azure SQL Database	Max. 500 MB	4 EUR/Monat
Azure Service Bus Quene	1 Mio. Nachrichten	1EUR/Monat
Azure Storage	<25 GB	1 EUR/Monat

Tabelle 5.1: Kostenabschätzung für die Nutzung TFS ASAP Online

Es wurde auch angenommen, dass die Migration von TFS ASAP bis hin zur Basis-Version von TFS ASAP Online im Rahmen dieser Diplomarbeit durchgeführt wird. Deswegen sollten keine weitere Personalkosten entstehen.

5.1.4 Anwendungsanalyse

In diesem Abschnitt wurde die Anwendung unter Berücksichtigung der Migration in die Cloud analysiert. Im Gegensatz zu der Analyse von TFS ASAP, die während der Erstellung

des Prototypen von TFS ASAP Online stattgefunden hat, wurde die Funktionsweise aus der technischen Sicht betrachtet. Zur Unterstützung bei dieser Aufgabe standen mir Mitarbeiter von AIT zur Verfügung, die an der Entwicklung von TFS ASAP beteiligt waren. Darüber hinaus waren die Dokumentationsdokumente sehr hilfreich. Die Dokumente [AITd] und [AITc] sind auf der offiziellen Produkt-Webseite von TFS ASAP abrufbar und sie haben den Charakter einer Einleitung für den Endkunden, wo der Umgang mit der Software erklärt wird, wie Installation und Verwaltung. [AITb] ist ein internes Dokument von AIT, in dem die Funktionsweise aus der technischen Sicht erklärt wird. Die Funktionsweise und Architektur wurde bereits im Abschnitt 2.4 beschrieben und aus diesem Grund wird es auf diese Aspekte nicht mehr eingegangen. Um die TFS-Plug-Ins besser zu verstehen, war auch [Sto11] sehr hilfreich.

Der Quellcode wird wie üblich bei Visual Studio und .NET-Sprachen mittels einer Solution und Projekten organisiert. Eine Solution besteht aus einer Menge von Projekten. Projekte können als Bausteine oder Komponenten angesehen werden. Die TFS ASAP-Solution besteht aus den unten gelisteten Projekten. Bei jedem Projekt wird kurz seine Funktion erklärt und im Kommentar werden zusätzliche Informationen erwähnt, welche für die Cloud-Migration relevant sind.

- WorkItemAutomations.Common
  - Funktion: Grundfunktionalität für die *triggerred* Automations.
  - Kommentar: Die Logik kann so übernommen werden. Es bestehen jedoch Abhängigkeiten zum TFS Object Model. Die Abhängigkeiten müssen eventuell abstrahiert werden.
- WorkItemAutomations.Common.Tests
  - Funktion: Tests für WorkItemAutomations.Common.
  - Kommentar: Kann so übernommen werden.
- WorkItemAutomations.Extended
  - Funktion: Erweiterte Funktionalität für die *triggerred* Automations.
  - Kommentar: Die Logik kann so übernommen werden. Es bestehen jedoch Abhängigkeiten zum TFS Object Model. Die Abhängigkeiten müssen eventuell abstrahiert werden.
- WorkItemAutomations.Extended.Tests
  - Funktion: Tests für WorkItemAutomations.Extended.
  - Kommentar: Kann so übernommen werden.
- ScheduledAutomations.Common
  - Funktion: Funktionalität für die *scheduled* Automations.

  - Kommentar: Die Logik kann so übernommen werden. Es bestehen jedoch Abhängigkeiten zum TFS Object Model. Die Abhängigkeiten müssen eventuell abstrahiert werden.

- ScheduledAutomations.Common.Tests
  - Funktion: Tests für ScheduledAutomations.Common.
  - Kommentar: Kann so übernommen werden.
- ASAP.Common
  - Funktion: Diese Projekt beinhaltet die infrastrukturelle Basisfunktionalität. Dazu gehören *Factory*-Klasse (Dependency Injection), Lizenz- und Logging-Funktionalitäten. Darüber hinaus sind da Schnittstellen für Objekte (*ITeamFoundationServer*, *ITeamProject* und *ITeamProjectCollection*[6]) und Dienste definiert.
  - Kommentar: Das Projekt muss übernommen werden, da es die notwendigen Schnittstelen beinhaltet, auf die die Grundfunktionalität von anderen Projekten aufbaut. Manche Services werden trotzdem nicht benötigt. Der Service, der die Schnittstelle *ISecurityService* muss neu implementiert werden, da die alte Implementation auf den Windows-Benutzer zugreift, während bei TFS ASAP Online das Microsoft Konto benutzt wird.
- ASAP.Common.Tests
  - Funktion: Tests für ASAP.Common.
  - Kommentar: Kann so übernommen werden.
- ASAP.SelfService.Core
  - Funktion: Dieses Projekt erweitert die Web-Benutzerschnittstelle von TFS.
  - Kommentar: Die Web-Benutzerschnittstelle von VSO kann nicht erweitert werden, deswegen wird dieses Projekt nicht übernommen..
- ASAP.SelfService.Core.Tests
  - Funktion: Test für ASAP.SelfService.Core.
  - Kommentar: Wie oben.
- ASAP.Setup
  - Funktion: Installationsassistent für TFS ASAP.
  - Kommentar: Die Installation wird vom Betreiber übernommen, der Benutzer braucht diese Funktionalität nicht.
- ASAP.Setup.CA
  - Funktion: Installationsassistent für TFS ASAP.

[6] In den .NET-Sprachen hat sich eine Konvention etabliert, dass die Namen von Schnittstellen mit dem großen Buchstaben I beginnen.

  - Kommentar: Die Installation wird vom Betreiber übernommen, der Benutzer braucht diese Funktionalität nicht.

- Common
  - Funktion: Gemeinsame Grundfunktionalitäten, die TFS ASAP benutzt. Das sind unter anderem: Komponente zur Verwaltung der Konfiguration von Automations, Aktivieren von Automations auf dem Projektebene und Laden von Bibliotheken.
  - Kommentar: Das Projekt wird übernommen, obwohl manche Komponenten nicht benutzt werden. Die Klasse *IProjectConfigurationService*, die zur Aktivierung von Automations dient, muss neu implementiert werden. Die Informationen wurden bisher in der Registry gespeichert. Für einen Cloud-Service, der möglicherweise aus mehreren Instanzen bestehen kann, unter Berücksichtigung der Mandantenfähigkeit ist eine solche Lösung nicht adäquat.
- Common.Client
  - Funktion: Hier befinden sich die Implementierungen von TFS ASAP Objekten aus dem Projekt ASAP.Common und Services für das Projekt Management.Web.
  - Kommentar: Alle Klassen müssen hier neu implementiert werden.
- Common.Server
  - Funktion: Hier befinden sich die Implementierung von TFS ASAP Objekten aus dem Projekt ASAP.Common und Services für das Projekt JobExtension.
  - Kommentar: Alle Klassen müssen hier neu implementiert werden.
- Common.Tests
  - Funktion: Tests für Common, Common.Client und Common.Server.
  - Kommentar: Kann so übernommen werden.
- Contracts
  - Funktion: Hier werden Schnittstellen für Objekte und Services zur Ausführung und Konfiguration von Automations zusammengefasst.
  - Kommentar: Das es sich in diesem Projekt nur um Schnittstellen handelt, kann es übernommen werden.
- JobExtension
  - Funktion: Projekt mit der Implementierung der JobExtension. Liest in zeitlichen Abständen Ereignisse aus der Datenbank, die dort von Subscriber gespeichert wurden, und führt Automations aus.
  - Kommentar: Der Code wird nicht übernommen, weil da eine Abhängigkeit zum Objekt der Klasse *TeamFoundationRequestContext* besteht. Diese Abhängigkeit kann nicht in einem Cloud-Szenario gewährleistet werden. In der Worker Role soll jedoch auf dem Abarbeitungsablauf dieser Komponente basiert werden.

- LicenseGenerator.Console
  - Funktion: Ein Projekt, dessen Aufgabe ist, Lizenzen zu generieren.
  - Kommentar: Ist für ein Cloud-Dienst nicht relevant und wird nicht übernommen.
- Management.Web
  - Funktion: Web-Anwendung zur Verwaltung von TFS ASAP: Aktivieren von Automations.
  - Kommentar: Wird übernommen, aber Anpassungen müssen vorgenommen werden. Zusätzlich muss auch um Komponenten für Mandantenfähigkeit und Benutzerverwaltung erweitert werden.
- Subscriber
  - Funktion: Implementiert die Schnittstelle *ISubscriber*. Empfängt Ereignisse von TFS und speichert sie in einer Datenbank ab.
  - Kommentar: Wird nicht übernommen. Diese Funktionalität übernimmt die Azure Service Bus.
- WorkItemExtensions.Core
  - Funktion: Im Rahmen dieses Projektes werden die *Work Item*-Definitionen erweitert.
  - Kommentar: In VSO steht die Möglichkeit nicht zur Verfügung, *Work Item*-Definitionen zu erweitern. Deswegen wird das Projekt nicht übernommen.
- WorkItemExtensions.Web
  - Funktion: Visualisierung der erweiterten *Work Items* im Rahmen des Projekts WorkItemExtensions.Core.
  - Kommentar: Wird nicht übernommen, weil WorkItemExtensions.Core auch nicht übernommen wird.

Aus den obigen Erkenntnissen kann man die Komponenten ermitteln, die ohne Anpassungen übernommen werden können, die mit Anpassungen übernommen werden können und die nicht übernommen werden. Zu de Komponenten, die ohne Anpassungen übernommen werden können, gehört das Projekt *Contracts*. Projekte, die nicht übernommen werden, sind: *ASAP.SelfService.Core*, *ASAP.SelfService.Core.Tests*, *ASAP.Setup*, *ASAP.Setup.CA*, *LicenseGenerator.Console*, *WorkItemExtensions.Core* und *WorkItemExtensions.Web*. Die restlichen Projekte müssen angepasst werden, oder basierend auf deren Logik werden neue Komponenten entwickelt.

5.1.5 Redesign

In dem vorigen Abschnitt wurde der Aufbau und die Architektur von TFS ASAP analysiert. Nun muss basierend auf dem gewonnenen Wissen ein Entwurf der zu migrierenden Anwendung erstellt werden. Für TFS ASAP Online wurde eine Zielarchitektur erarbeitet, die im Abschnitt 2.5 zu finden ist. Für die zu übernehmenden Komponenten, die angepasst werden müssen, werden die notwendigen Anpassungen abgeleitet. Diese sind:

- Für die Komponenten zur Ausführung von Automations bieten sich zwei Möglichkeiten an. Entweder müssen die Abhängigkeiten zu TFS-Bibliotheken durch Abstraktion und Neuimplementierung der Logik mit der VSO REST API behoben werden, oder die Implementierung kann unverändert bleiben, vorausgesetzt der Code mit den Abhängigkeiten zu TFS-Bibliotheken kann in einer Cloud-Umgebung ausgeführt werden.
- Für die infrastrukturellen Basisfunktionalitäten müssen gegebenenfalls Abhängigkeiten behoben werden, die für eine Cloud-Umgebung nicht geeignet sind. Das sind die Implementierungen der Schnittstellen *IProjectConfigurationService* und *ISecurityService*. Die erste Implementierung benutzt die Registry zur Speicherung von Einstellungen. Die andere benutzt den angemeldeten Windows-Benutzer zur Authentifizierung. Beide Implementierungen sollten angepasst werden und auf der neu angelegten Datenbank basieren.
- Neben den Projekten *Common.Server* und *Common.Client* soll ein Projekt *Common.Cloud* angelegt werden, in dem die Schnittstellen für Objekte und Services von TFS ASAP Online implementiert werden.
- Das Projekt *Management.Web* wird übernommen, jedoch muss es angepasst werden. Der Authentifizierungsmechanismus basiert auf Windows-Authentifizierung und muss durch eine Authentifizierung mit Hilfe von Microsoft Account[7] ersetzt werden. Darüber hinaus muss das Portal mit Mandantenfähigkeit erweitert werden. Dazu gehört die Implementierung des Anmeldevorgangs und des Flusses für das Hinzufügen von VSO-Accounts. Die Benutzerverwaltungslogik soll Benutzer und Mandanten berücksichtigen.
- Die Projekte *Subscriber* und *JobExtension* werden nicht übernommen, aber ihre Logik wird in der Worker Role verwendet. Die Worker Role muss zusätzlich, basieren auf den Ereignissen, den Benutzer und seinen Mandanten erkennen, seine Einstellungen aus der Datenbank lesen, und dann entsprechend eine Automation ausführen. Die Worker Role wird auch zu einer Komponente, die skaliert wird. Zur Implementierung der Worker Role wurde das Entwurfsmuster *Competing Consumers Pattern* aus [HSB+14] angewendet.

Neben den notwendigen Anpassungen wurden auch zusätzliche Komponenten entworfen, die für TFS ASAP Online notwendig sind. Diese sind:

[7] Microsoft Account - A single login to services and devices: http://www.microsoft.com/en-us/account/default.aspx

- Azure SQL Datenbank - Hier werden die Daten abgelegt, deren Datenmodell in der Abbildung 5.3 dargestellt ist. Zum einen befinden sich in der Datenbank die Informationen über die angemeldeten Benutzer und deren Mandanten. Zum anderen sind das auch Daten, die die VSO-Accounts der Benutzer und ihre interne Struktur mit *Collections* und *Projects*. Die Datenbank dient als Integrationskomponente zwischen der Web-Benutzerschnittstelle und der Worker Role.
- Azure Service Bus - Diese Komponente empfängt Ereignisse von VSO, die von der Worker Role gelesen und verarbeitet werden.

In der Abbildung 5.3 ist das für TFS ASAP Online in dieser Aufgabe erarbeitete Datenmodell dargestellt. Das Model wurde im Laufe der Migration mehrfach korrigiert und verbessert. Es besteht aus folgenden Entitäten:

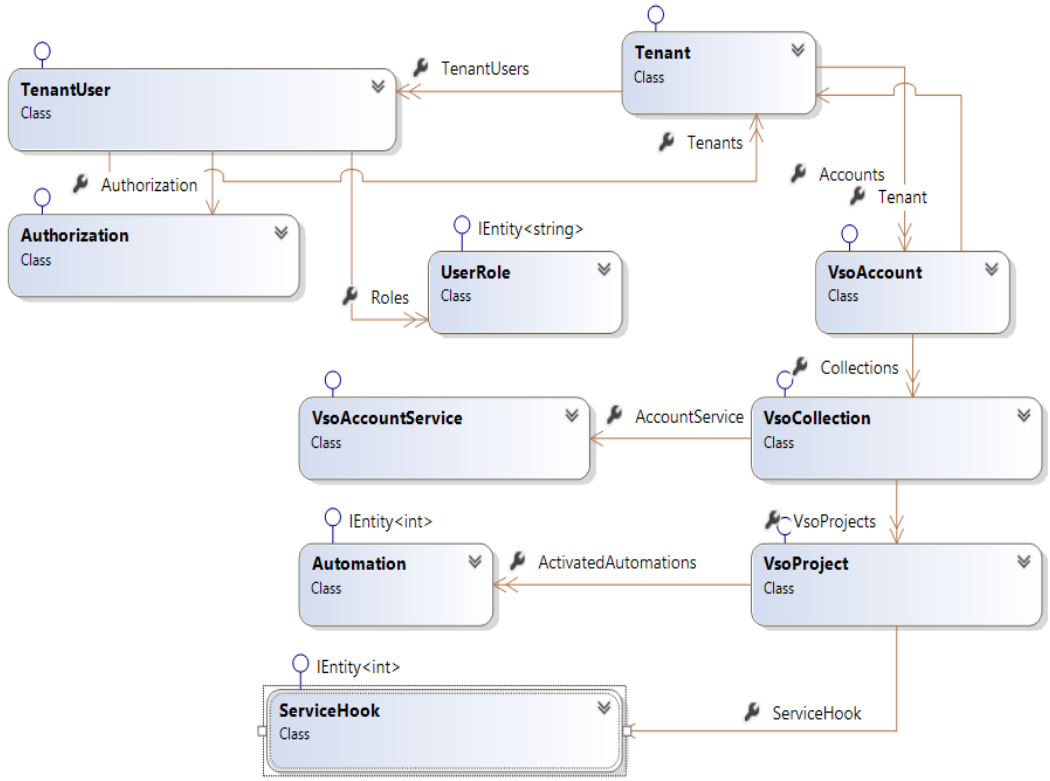

Abbildung 5.3: TFS ASAP Online: Datenmodell

- Tenant - Entspricht einer Organisation. Kann mehrere TenantUser (Benutzer) haben, die ihre VSO-Accounts (das heißt, sie sind Besitzer eines VSO-Accounts - *Owner*) für diesen Tenant registrieren können.
- TenantUser - Entspricht einem Benutzer. Wird mit dem Microsoft Account seines Benutzers verknüpft. Microsoft Account dient bei TFS ASAP Online zur Authentifizierung

von Benutzern. Der TenantUser kann mehreren Tenants angehören. Der TenantUser muss auch seine VSO-Identität für TFS ASAP Online autorisieren, damit die Benutzung von der VSO REST API in seinem Namen erfolgen kann (siehe auch Abschnitt 2.3.1).

- Autorization - Beinhaltet Autorisierungsinformationen der VSO-Identität eines TenantUsers. Das sind unter anderem der *Token*, und der *RefreshToken*, der dazu dient, einen neuen Token zu beziehen, wenn der alte abgelaufen ist (siehe auch Abschnitt 2.3.1).
- UserRole - Definiert die Rollen, welche eine TenantUser hat. Dient zuerst dazu, die Benutzer zu unterscheiden, die nur Konsumenten von TFS ASAP Online sind, und die, die auch Administrator-Rechte haben.
- VSOAccount - Repräsentiert einen VSO-Account, die einer Form wie *https://<account-name>.visualstudio.com/* ist. Ein VSO-Account besteht aus Collections, die wiederum aus Projekten bestehen. Im Moment hat ein VSO-Account nur eine Collection, und es besteht eine 1-zu-1-Beziehung zwischen dem VSOAccount und VSOCollection, so dass es sinnvoll wäre, die zwei Entitäten mit einer zu ersetzen. Da es möglich ist, dass dies bald nicht mehr der Fall ist[8], entschied man sich, das Datenmodell generisch zu gestalten, damit zusätzlicher Aufwand in der Zukunft gespart werden kann. Ein VSO-Account darf nur einmal im System vorkommen. Das hat zur Folge, dass der TenantUser seinen VSO-Account nicht zweimal hinzufügen darf oder dass er den gleichen VSO-Account bei seinem anderen Tenant hinzufügen darf.
- VSOCollection - Repräsentiert die Collections bei einem VSO-Account. Im Moment hat eine VSO-Account nur eine Collection: *DefaulfCollection*.
- VSOAccountService - Repräsentiert einen AccountService einer Collection. Mit dem AccountService lassen sich administrative Tätigkeiten für eine Collection erledigen. Diese Entität war notwendig, als die Anforderung erschien, dass in der *History* eines Work Items nicht der Benutzer als der Verursacher der Änderung erscheinen sollte, sondern der AccountService. Diese Entscheidung hatte auch zur Folge, dass das TFS Object Model zur Ausführung von Automations benutzt wurde.
- VSOProject - Repräsentiert die Projekte einer Collection.
- Automations - Repräsentiert die Automations, deren Ausführung für das gegebene Projekt aktiviert ist.
- ServiceHook - Repräsentiert die Service Hooks von Visual Studio Online REST API, die für das gegebene Projekt angelegt sind. Ein angelegter Service Hook bedeutet, dass das System sich für die Ereignisse in diesem Projekt registriert hat.

Darüber hinaus wurden in diesem Schritt kritische Funktionalitäten abgeleitet, die während der Proof-of-Concept-Phase zu validieren sind. Zu prüfen war, ob folgende Funktionalitäten implementierbar sind und somit die damit verbundenen Anforderungen erfüllt werden können.

[8] Let us create multiple collections on Visual Studio Online - Customer Feedback for Microsoft: https://visualstudio.uservoice.com/forums/121579-visual-studio/suggestions/3473653-let-us-create-multiple-collections-on-visual-studi

- **Automations ausführen** - Es muss geprüft werden, ob eine Automation von einer Cloud-Umgebung aus ausgeführt werden kann. Es geht hierbei um eine beliebige Automation, derer Konfiguration nicht dynamisch geladen ist, sondern hart kodiert. Informationen wie Benutzer oder Mandant werden nicht berücksichtigt.
- **Anmelden mit dem Microsoft Account** - Es muss geprüft werden, wie eine Anmeldung mit Microsoft Account in eine Microsoft Azure Umgebung zu implementieren ist.
- **Visual Studio Online REST API** - Es musste geprüft werden, wie weit einsetzbar die Visual Studio Online REST API ist. Die Features, die geprüft werden sollen, sind:
  - Autorisierung der Anwendung mit VSO REST API.
  - OAuth2-Autorisierung und Beziehen vom Token.
  - Work Item Tracking - Abfragen und Veränderung von Work Items mit Visual Studio Online REST API.

5.1.6 Migrationsplan

In dieser Aufgabe wird der Plan für die Migration von TFS ASAP in die Microsoft Azure Cloud Umgebung erstellt. Zu diesem Zweck wurden für alle Phasen der Cloud-Migrations-Methodologie, die im Kapitel 4 entwickelt wurde, Umsetzungszeiten abgeschätzt und Meilensteine definiert. Die Basiszeiteinheit, die den Abschätzungen zugrunde liegt, ist die KW. Eine KW besteht aus fünf KT. Für einen KT wird von einem Dauer von 5 Stunden ausgegangen.

Der Projektstart wurde für den 14. Juli 2014 angesetzt. An diesem Tag sollten die Arbeiten an der Cloud-Migration von TFS ASAP beginnen. Darüber hinaus ein wichtiges für das Projekt Datum war 3. November 2014. An diesem Tag sollte in Redmond, USA der MVP Global Summit[9] stattfinden. Most Valuable Professional (MVP) ist eine Auszeichnung, die von Microsoft an Personen mit besonders herausragender technischer Kompetenz und einem starken Community-Engagement verliehen wird. Jährlich treffen sich die MVP's an einem von Microsoft organisierten Gipfel, um zu diskutieren und Wissenstransfer zu betreiben. Da einige AIT-Mitarbeiter auch MVP's sind, erhoffte man sich, dass man dort TFS ASAP Online den Gästen vorstellen könnten, um Feedback und Unterstützung zu bekommen. Aus diesem Grund wurden die Zeitrahmen zur Durchführung der Migration von KW 29 (14.7.14 - 18.07.14) bis einschließlich KW 44 (27.10.14 - 31.10.14) festgelegt. Am 31.10.14 sollte die Veröffentlichung des Endproduktes für die Kunden erfolgen. Zusätzlich wurde für Ende der KW 40 (3.10.14) als Endtermin für die Bereitstellung der Preview-Version von TFS ASAP Online angesetzt. Auf diese Weise sollte das Migrationsprojekt eine Dauer von 16 KW haben. Bei der Planung wurde berücksichtigt, dass zur Erledigung der Aufgaben gewisse Lern- und Einarbeitungszeit notwendig ist. In der Tabelle 5.2 wird der Zeitplan für die Migration von TFS ASAP dargestellt. Er beinhaltet die einzelnen Phasen. Für alle Phasen sind Start- und End-KW's definiert, sowie die Dauer. In der letzten Zeile sind die aggregierten Werte für die gesamte Migration zu sehen.

[9] MVP Global Summit: http://mvp.microsoft.com/summit

Nr.	Name	Start	Ende (einschließlich)	Dauer (KW)
1	Analyse und Planung	KW 29	KW 31	3 KW
2	Proof of Concept	KW 32	KW 33	2 KW
3	Anwendungsmigration	KW 34	KW 38	5 KW
4	Optimierung und Testen	KW 39	KW 41	3 KW
5	Betrieb und Verwaltung	KW 42	KW 44	3 KW
∑	**Migration**	**KW 29**	**KW 44**	**16 KW**

Tabelle 5.2: Zeitplan für die Migration von TFS ASAP

Für die einzelnen Phasen der Migration wurden folgende Meilensteine definiert:

- **1. Analyse und Planung:** folgende Dokumente werden gefordert: Prototyp, Anforderungsdokument, Migrationsplan, Anwendungsanalyse, Design, Kostenplan und Microsoft Azure Überblick
- **2. Proof of Concept:** Machbarkeit folgender Funktionalitäten muss bewiesen werden: Der gesamte Fluss der Automationsausführung: von Ereignis bis zur Veränderung des Work Items, Anmelden mit Microsoft Account und Manipulation von VSO mit VSO REST API
- **3. Anwendungsmigration:** folgende Funktionalitäten sind zu implementieren. Die Funktionalitäten 1-5 stellen die erforderte Basisfunktionsumfang und sie müssen unbedingt implementiert werden. Die Funktionalitäten sind nach ihrer Priorität sortiert.
  - 1. Anmeldung bei TFS ASAP Online
  - 2. Hinzufügen eines VSO-Accounts
  - 3. Triggerred Automations aktivieren
  - 4. Triggerred Automations ausführen
  - 5. Anwendungsportal anpassen
  - 6. Logging und Monitoring
  - 7. Web GUI für Administratoren
  - 8. Scheduled Automations aktivieren
  - 9. Scheduled Automations ausführen
- **4. Optimierung und Testen:** Eine getestete Version mit behobenen Fehlern. Die Tests beziehen sich auch auf Performanz und Skalierbarkeit.
- **5. Betrieb und Verwaltung:** Eine Version mit implementiertem Monitoring-Mechanismus.

5.2 Proof of Concept

Diese Phase hat zwei Ziele. Zu einem ist das ein Wissensaufbau im Bereich Microsoft Azure und Entwicklung von Cloud-Anwendungen. Zum anderen ist eine Machbarkeitsstudie durchzuführen, die bei der Beantwortung der Frage helfen soll, ob die gewählten Lösungen realisierbar sind. Die Lösungen wurden in der vorherigen Phase gewählt und sie stellen die kritische Funktionalität von TFS ASAP Online dar, die die Software aufweisen muss, um alle geforderten Szenarien zu realisieren. In den folgenden Unterabschnitten werden Aufgaben vorgestellt, die im Rahmen dieser Phase erledigt wurden. Für jede zu testende kritische Funktionalität wurde eine separates Projekt angelegt, so dass die Funktionalitäten unabhängig voneinander geprüft werden konnten. Auf diese Weise entspricht ein Unterabschnitt einer Funktionalität.

5.2.1 Automations ausführen

In dieser Aufgabe musste geprüft werden, ob es möglich ist ein Szenario zu realisieren, in dem ein Ereignis von VSO erzeugt wird und an einen Cloud-Dienst in Microsoft Azure verschickt wird. Der Cloud-Dienst soll das Ereignis empfangen und eine Automation für einen VSO-Account ausführen, basierend auf den im Ereignis beinhalteten Informationen.

Zuerst wurde ein Azure Service Bus manuell mit Azure Management Portal angelegt. In einem VSO-Account wurde eine Subscription erstellt, in dem man die vorher angelegte Azure Service Bus als Empfänger einstellt. Darauffolgend hat man einen Work Item in VSO geändert. Um zu prüfen, ob das Ereignis tatsächlich erzeugt und an die Service Bus verschickt wird, benutzte man das Werkzeug Service Bus Explorer. Mit Service Bus Explorer[10] lassen sich unter anderem Nachrichten aus einer Queue lesen. Mit diesem Werkzeug konnte festgestellt werden, dass der Subskription-Mechanismus funktioniert.

Der nächste Schritt bestand darin, eine Worker Role zu entwickeln und zu prüfen, ob sie Nachrichten aus der Queue empfangen und sie weiter verarbeiten kann. Das war das erste Mal während der Diplomarbeit, dass eine Programmierungsaufgabe zu erledigen war. Als Erstes sollte die Azure SDK for .NET[11] heruntergeladen und installiert werden. Azure SDK for .NET ist eine Fülle von Werkzeugen für Visual Studio, Kommando-Zeile, Bibliotheken, die eine Unterstützung bei der Entwicklung von Microsoft Azure Lösungen anbieten. Die Azure SDK liefert auch Vorlagen für Visual Studio Projekte. Eine dieser Vorlage ist *Worker Role with Service Bus Queue* und diese wurde für das Testprojekt ausgewählt.

Für die Implementierung der Worker Role wurde das Entwurfsmuster *Competing Consumers Pattern* aus [HSB+14] eingesetzt. Das Empfangen von Nachrichten wurde Ereignis-gesteuert implementiert, das heißt, dass die Worker Role von der Queue angesprochen wird, sobald eine neue Nachricht in der Queue erscheint. Es wurde auf der Implementation von *JobExtension* aus TFS ASAP basiert. In der Konfiguration der Worker Role wurden Verbindungsinformationen des Azure Service Bus Namespace und der Queue abgelegt. Die Ereignisse kommen

[10] Service Bus Explorer: https://code.msdn.microsoft.com/windowsapps/Service-Bus-Explorer-f2abca5a
[11] What is the Azure SDK for .NET?: http://azure.microsoft.com/en-us/documentation/articles/dotnet-sdk/

in Form von JSON-String vor. Wrapper-Klassen zur (De-)Serialisierung von JSON-String, die als Ant wurden erstellt. Dazu wurde das Online-Werkzeug json2csharp[12] benutzt. Um zuerst keinen zusätzlichen Aufwand zu verursachen, wurde Objekte dieser Klasse in Objekte der Klasse *WorkItemChangedEvent* konvertiert, die normalerweise von TFS erzeugt werden und die von TFS ASAP Automations benutzt werden. Damit musste keine Änderungen am Code von Automations vorgenommen werden. Bei TFS, um eine Automation ausführen zu können, muss zuerst ermittelt werden, ob die gegebene Automation für das Projekt aktiviert ist und ihre Konfiguration muss aus einer externen XML-Datei eingelesen werden. Der Einfachheitshalber wurden diese Informationen in dem Testprojekt in der Abarbeitungsmethode hart-kodiert.

```
1  <ServiceDefinition name="ASAP.Cloud" xmlns="http://schemas.microsoft.com/
       ServiceHosting/2008/10/ServiceDefinition" schemaVersion="2014-06.2.4">
2    <WorkerRole name="WorkerRoleWithSBQueue" vmsize="ExtraSmall">
3      <Startup>
4        <Task commandLine="SetupFiles\InstallTFSObjectModel.cmd"
             executionContext="elevated" taskType="simple">
5          <Environment>
6            <Variable name="EMULATED">
7              <RoleInstanceValue xpath="/RoleEnvironment/Deployment/@emulated"
                   />
8            </Variable>
9          </Environment>
10       </Task>
11       ...
12     </Startup>
13     ...
14   </WorkerRole>
15   ...
16 </ServiceDefinition>
```

Listing 5.1: Konfigurationsdatei Worker Role mit Installation des TFS Object Models

Zur Veränderung der Work Items wurde der Code für Automations nicht geändert. Da wird die TFS API eingesetzt. Diese ist eine Bibliothek, die unter anderem zur Manipulation von TFS benutzt werden kann. Somit konnte sichergestellt werden, dass die TFS API auch von der Cloud aus einsatzfähig ist. Zu diesem Zweck wurden die TFS ASAP Schnittstellen aus dem Projekt *ASAP.Common* implementiert. Zur Authentifizierung wurden Login und Passwort von meinem VSO-Account im Klartext abgelegt. Die Bibliotheken mussten in der Worker Role während des Deployments mit TFS Object Model Installer[13] installiert werden. Dafür muss man in der Konfigurationsdatei ein Task definieren. Der Ausschnitt aus der

[12] json2csharp: http://json2csharp.com/

[13] Team Foundation Server 2013 Object Model Installer: https://visualstudiogallery.msdn.microsoft.com/3278bfa7-64a7-4a75-b0da-ec4ccb8d21b6

Konfigurationsdatei befindet sich in dem Codeausschnitt 5.1. Diese Vorgehensweise wurde aus dem Open-Source-Projekt Team Foundation Server OData API[14] übernommen.

Der erste Test wurde auf der lokalen Maschine mit Microsoft Azure Emulator durchgeführt, welcher mit Azure SDK geliefert wurde. Solch eine Vorgehensweise ermöglicht auch das Debuggen des Quellcodes. Für dieses Szenario konnte der Emulator nur Azure Cloud Service emulieren. Deswegen wurde ein richtiger, kein emulierter Azure Service Bus benutzt. Auf diese Weise konnten die Fehler identifiziert und behoben werden. Nach der Fehlerbehebung konnte man feststellen, dass der Test in einer emulierten Umgebung erfolgreich war. Danach galt es den Test in einer reellen Cloud-Umgebung durchzuführen. Das Deployment der Worker Role war einfach dank der Unterstützung von Visual Studio. Mit Hilfe des Assistenten konnte der Deployment Profile erstellt werden und der Dienst wurde in Azure ausgeliefert. Der Test hat sich ebenfalls als erfolgreich ergeben - bei Manipulation eines Work Items wurde das Ereignis generiert, an die Queue geschickt, da von der Worker Role gelesen. Die Worker Role konnte eine Automation basierend auf den Informationen aus dem Ereignis ausführen. Zusätzlich wurde eine Möglichkeit entdeckt, einen in der Cloud laufenden Dienst zu debuggen[15].

5.2.2 Anmelden mit dem Microsoft Account

Der Benutzer von TFS ASAP Online sollte sich mit seinem Microsoft Account anmelden können. Der Microsoft Account dient der Authentifizierung von Benutzern. Im Rahmen dieser Aufgabe sollte ein Pilotprojekt erstellt werden, wessen Ziel war, den Mechanismus, der sich hinter der Authentifizierung mit Microsoft Account verbirgt, zu verstehen. Zum Testen dieser Funktionalität wurde in Visual Studio ein ASP.NET MVC-Projekt angelegt, das sowohl lokal und als auch in der Cloud-Umgebung als Azure Website getestet wurde. Zu diesem Zweck mussten zwei Versionen der Anwendung zur Verwendung der Microsoft Account Authentifizierung registriert werden, da es für eine Anwendung nur eine RedirectUrl-Adresse eingegeben werden kann. Die Vorgehensweise wurde auf der Webseite von Microsoft Azure[16] beschrieben. Damit das Testen in der lokalen Umgebung möglich ist, mussten einige Einstellungen vorgenommen werden, über die in [Day14] und [Pel14] nachzulesen ist. Zusammen mit dem Testen der Microsoft Account Authentifizierung wurde auch eine SQL Express Datenbank angelegt, die nur eine Entität beinhaltete. In dieser Entität erfolgte die Verknüpfung von den Informationen aus den *Claims* [BBB+11] des Microsoft Account und den TFS ASAP Online-spezifischen Daten. Zur Erstellung der Datenbank wurde der Code First Ansatz und eine lokale SQL Express Datenbank benutzt. Um die Veränderung des Datenmodells einzuspielen, fand die Funktionalität Code First Migrations[17] Verwendung. Mit Code First

[14] Team Foundation Server OData API: https://tfsodata.visualstudio.com/

[15] Debugging a Cloud Service or Virtual Machine in Visual Studio: http://msdn.microsoft.com/en-us/library/azure/ff683670.aspx

[16] Register your apps to use a Microsoft Account login: http://azure.microsoft.com/en-us/documentation/articles/mobile-services-how-to-register-microsoft-authentication/

[17] Code First Migrations and Deployment with the Entity Framework in an ASP.NET MVC Application: http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/migrations-and-deployment-with-the-entity-framework-in-an-asp-net-mvc-application

lässt sich das Datenmodell einer Datenbank mithilfe von NuGet[18] aktualisieren.

Diese Aufgabe wurde erfolgreich durchgeführt und es konnte wie in der vorherigen Aufgabe festgestellt werden, dass die geforderte kritische Funktionalität machbar ist.

5.2.3 Visual Studio Online REST API

In dieser Aufgabe ging es darum, sich mit der neuen Visual Studio REST API vertraut zu machen. Einige für TFS ASAP Online relevante Abfragen sollten durchgeführt werden, sowie die Funktionsweise der OAuth2-Authentifizierung von VSO REST API war zu prüfen. Für die erste Teilaufgabe wurde ein Konsole-Projekt angelegt, in dem einige Abfragen gegen einen privaten VSO-Account durchgeführt wurden. Es handelte sich um das Lesen und Editieren von Work Items. Es wurde die Notwendigkeit erkannt, Wrapper-Klassen für die JSON-Strings zu entwickeln, die mit VSO REST API via HTTP-Protokoll verschickt werden. Die JSON-String sollten mit den Wrapper-Klassen sollten (de-)serialisiert werden. Zu ihrer Implementierung wurde die Bibliothek Json.NET[19] eingesetzt.

Eine weitere Herausforderung bestand darin, die OAuth2-Authentifizierung zu prüfen. Dazu wurde ein ASP.NET MVC-Projekt mit einer SQL Express Datenbank angelegt. Die Einstellung der Test- und Cloud-Umgebung sah ähnlich aus, wie bei der Aufgabe mit dem Anmelden mit dem Microsoft Account - die Anwendung musste registriert werden und eine entsprechende Methoden zur Entgegennahme des Autorisierung-Codes (CallbackUrl) musste implementiert werden. Danach wurde der Autorisierung-Token angefordert und bezogen. Mit dem Token wurde geprüft, ob die Durchführung von REST-Abfragen funktioniert. Das Gleiche galt für die Aktualisierung vom Token, denn seine Gültigkeit dauert nur ca. 15 Minuten. Die Informationen zu OAuth-Authentifizierung können in [Mic14] gefunden werden. Die Informationen, die mit dem Token geliefert wurden, wurden in der Datenbank abgelegt. Die Funktionalitäten wurden ähnlich wie in der vorherigen Aufgabe in der Cloud mit Azure Websites getestet. Für die Datenbank wurden der Code First Ansatz und für das Aktualisieren des Datenmodells Code First Migrations verwendet. Mit dieser Aufgabe konnte die Frage nach Machbarkeit und Tauglichkeit der geforderten Funktionalität positiv beantwortet werden.

5.3 Anwendungsmigration

In dieser Phase findet die Durchführung der eigentlichen Migration von TFS ASAP Online statt. In der vorherigen Proof-Of-Concept-Phase wurden Erkenntnisse und Informationen gewonnen, mit welchen die erhobenen Anforderungen und die Zielarchitektur aus den früheren Phasen validiert wurden. Ihre Umsetzung wird in diesem Abschnitt beschrieben.

[18] Entity Framework Code First Migrations: Alpha - NuGet Package of the Week 10: http://www.hanselman.com/blog/EntityFrameworkCodeFirstMigrationsAlphaNuGetPackageOfTheWeek10.aspx
[19] Json.NET: http://james.newtonking.com/json

5.3.1 Anpassungen

Im ersten Schritt sollten Anpassungen vorgenommen werden und in den Quellcode der Applikation einfließen. Diese Aufgabe hat sich als sehr langwierig ergeben, da es an vielen Stellen gearbeitet werden musste und im Laufe der Arbeiten manche Entscheidungen aus der Phase *Analyse und Planung* geändert wurden. Alle Schritte werden hier in chronologischer Reihenfolge dargestellt und es wird immer darauf hingewiesen, wenn ein Schritt Folgen im weiteren Durchlauf verursacht hat.

Zuerst wurde die Worker Role implementiert. Dazu wurde ein Projekt Namens *WorkerRoleWithSBQueue* angelegt. Der Code wurde von dem Pilotprojekt der Proof-of-Concept-Phase übernommen und weiter verfeinert. Zuerst wurde die TFS API für die Kommunikation mit dem VSO benutzt. Im neu hinzugefügten Projekt *Common.Cloud* wurden die Schnittstellen aus dem Projekt *ASAP.Common* implementiert. Diese Schnittstellen waren *ITeamFoundationServer*, *ITeamProject* und *ITeamProjectCollection*. Dazu wurde die TFS API eingesetzt, die auch im Fall von TFS ASAP ihre Verwendung fand. Diese erste vorläufige Implementierung kam ebenfalls aus der Proof-of-Concept-Phase. Diese verfügte jedoch lediglich über rudimentäre Basisfunktionalität, die notwendig war, um die Lösung zu validieren. Sie wurde sie um weitere erforderliche Implementierungen ergänzt. Die Worker Role beinhaltete hart-kodierte Definitionen von Automations aus TFS ASAP. Es wurde nicht dynamisch ermittelt, welche Automations für das eingetroffene Ereignis durchzuführen sind, sondern alle im Code abgelegten Automations wurden ausgeführt.

Zu diesem Zeitpunkt kam es zu einer Entscheidungsänderung. Man entschied, dass zur Manipulation von Work Items die VSO REST API benutzt wird. Auf diese Weise mussten der Login und das Passwort vom Benutzer nicht aufgefordert werden. Statt dessen benutzte man zur Autorisierung den OAuth-Mechanismus, der für solche Szenarien zur Verfügung stand. Diese Entscheidung hatte zusätzliche Anpassungsaufgaben zur Folge. Im Code der Automations mussten jegliche Abhängigkeiten zur TFS API durch Schnittstellen abstrahiert werden. Ein REST-Client zur Ausführung von REST-Aufrufe mit Autorisierung mit OAuth wurde entwickelt. Der REST-Client musste in die Implementierungen der extrahierten Schnittstellen integriert werden. Das Gleiche galt für die TFS ASAP Objekte, wie *ITeamFoundationServer*, *ITeamProject* und *ITeamProjectCollection*. Bei dem Testen wurde festgestellt, dass die VSO REST API immer noch in Kinderschuhen steckt. Ein Problem bereitete die Änderung des Felds *State* eines Work Items. Aufgrund dieses Problems, das auch viele Leute aus der Community meldeten, verfasste ich einen Blog-Beitrag [Sab14b], der eine Lösung für das Problem beschrieb. Zur Implementierung mussten die XML-Konfigurationsdateien dem Automations-Projekt beigefügt werden. Vor der Ausführung einer Automation, die den Wert des Feldes *State* eines Work Items ändern sollte, sollten die der Ausgangs- und Zielzustand für die entsprechende Transition gefunden werden, damit die Werte für die weiteren erforderlichen Felder ermittelt werden konnten.

Die Änderungen von Work Items, die mittels VSO REST API gemacht werden, kommen in der *History* des Work Items vor, als wären sie vom Benutzer durchgeführt worden, der eine OAuth-Autorisierung für TFS ASAP Online gemacht hat. Da in der Historie des Work Items unterschieden werden sollte, wenn eine Änderung nicht direkt vom Benutzer verursacht

wurde, kehrte man zum dem Ansatz mit der TFS API zurück und man nahm in Kauf, dass der Benutzer seine Login-Daten eingeben soll. Hinter den Kulissen wurde von VSO der sogenannte *AccountService* bezogen [Hin14]. Dies bedeutete sehr viel Fleißarbeit, die in die Wiederherstellung des Codes investiert werden musste. Dank der Ablage des Quellcodes in der Versionskontrolle musste nicht alles von Anfang an implementiert werden. Am Ende waren in der Solution von TFS ASAP Online zwei Projekte vorhanden: *Common.Cloud* mit der Implementierung mit der VSO REST API und *Commmon.CloudTfsApi* mit der Implementierung mit der TFS API. Obwohl die Implementierung mit der VSO REST API nicht in das Endprodukt hinein geflossen ist, war die Arbeit daran nicht sinnlos und das Ergebnis kann bald seinen Einsatz finden, denn die nächste Version von TFS ebenfalls die REST API zur Verfügung stellen sollte, so wie das Brian Harry von Microsoft in den Kommentaren unter seinem Blog-Eintrag angekündigt hat.[20].

Danach wurden Test durchgeführt, mit denen man feststellen konnte, dass alle Automations von TFS ASAP funktionieren. Dann war die Anpassung der Web-Benutzerschnittstelle aus dem Projekt *Management.Web* angesagt. Als erste Funktionalität musste die Authentifizierung mit Microsoft Account implementiert werden. Zu diesem Zweck wurde das Datenmodell, der im Rahmen der Phase *Analyse und Planung* entstanden ist, eingesetzt. Das Datenmodell wurde während der Implementierung mehrfach angepasst. Es wurden Klassen für den Datenzugriffsschicht unter Einsatz der Entwurfsmuster *Repository*[21] und *Unit of Work*[22] implementiert. Die Datenbank wurde mittels Code First Ansatzes aus den entwickelten Klassen generiert und jegliche Änderungen des Datenmodells wurden mittels Code First Migrations eingespielt. Es wurde der Ablauf der Registrierung eines Benutzers bei TFS ASAP entwickelt. Der Benutzer meldet sich mit seinem Microsoft Account, bestätigt, dass sein TFS ASAP Online Konto angelegt werde sollte und dann hat er die Möglichkeit, einen Tenant auszuwählen oder einen neuen anzulegen. Danach wird er dazu aufgefordert, seine VSO-Identität für TFS ASAP Online zu autorisieren. Der REST-Client aus dem vorherigen Schritt wurde um die Funktionalitäten erweitern, die es ermöglichen Informationen über die VSO-Accounts und ihre Collections und Projekte abzufragen. Die Tokenverwaltungsmechanismus wurde optimiert. Danach wurde das Hinzufügen von VSO-Accounts implementiert. Die hinzugefügten VSO-Account mit ihren Collections und Projekten werden in der Datenbank hinterlegt. Auf diese Weise konnte das System über alle Daten verfügen, die notwendig sind, um die einzelnen Automations für Collections und Projekte anzuzeigen und ihre Ausführung zu (de-)aktivieren. Für die Anzeige der Automations musste die Automationskonfigurationsdatei dem Projekt *Management.Web* hinzugefügt werden. Nur die in der Konfigurationsdatei aufgelisteten Automations werden in der Web-Benutzerschnittstelle dargestellt. Dazu mussten die Schnittstellen *IProjectConfigurationService* implementiert werden. Nun wurden die Informationen, ob die Ausführung von Automations für eine Collection oder ein Projekt aktiviert ist, in dem Datenmodell abgebildet. Für das Aktivieren der Automations wurde zusätzlich das Anlegen von Subscriptions mittels VSO REST API implementiert. Dazu musste der REST-Client erweitert werden und ein zusätzlicher Entitätstyp *ServiceHook* dem Datenmodell hinzugefügt werden.

[20] A new API for Visual Studio Online: http://blogs.msdn.com/b/bharry/archive/2014/05/12/a-new-api-for-visual-studio-online.aspx
[21] Repository Design Pattern: http://martinfowler.com/eaaCatalog/repository.html
[22] Unit of Work Design Pattern: http://martinfowler.com/eaaCatalog/unitOfWork.html

Ein zusätzlicher Entitätstyp wurde auch für die aktivierten Automations eines Projektes hinzugefügt (*Automations*). Mit der Implementierung der Schnittstelle *ISecurityService* wurde gewährleistet, dass nur diejenigen Nutzer, die die Rolle *Project Collection Administrator* haben, die Ausführung der Automations für die jeweilige Collection (de-)aktivieren können.

Dieses Projekt wurde zuerst lokal mit Azure Emulator und SQL Server Express getestet. Dann wurde es als Azure Web Site deployed. Visual Studio stellt einen Assistenten zur Verfügung, mit dem eine Web Site mit der Datenbank schnell und einfach bei Microsoft Azure zu deployen ist.

Eine vorläufige Version von TFS ASAP Online, die sich aus einer Version der entwickelten Worker Role mit einigen hart-kodierten Automations, das Portal und die Datenbank zusammensetzte, wurde zu diesem Zeitpunkt in Microsoft Azure deployed und intern mit dem VSO-Account von AIT getestet. Die Benutzer sollten alle beobachteten Unstimmigkeiten unverzüglich melden. Somit wurde die erste Testphase angegangen.

Da das Portal und die Datenbank, die als die Integrationskomponente diente, fertig implementiert waren, konnte man die Funktionalität der Worker Role weiter ausbauen. Der Abarbeitungsfluss eines Ereignisses wurde so erweitert, dass nun basierend auf den Informationen, die mit dem Ereignis mitgeliefert werden, VSO-Account, Collection und Projekt ermittelt werden, auf die sich das Ereignis bezieht. Anhand dieser Daten kann festgestellt werden, ob und welche Automations aktiviert sind und somit ausgeführt werden sollten.

Somit wurden alle Komponenten implementiert und konnten ausgeliefert werden. Inzwischen wurde das Portal, der bisher als Web Site deployed wurde, in einem Azure Cloud Service als Web Role mit der Worker Role zusammengefasst. Alle Dienste wurden in Microsoft Azure deployed. Die engen Partner von AIT wurden eingeladen, den Dienst zu testen. Der nächste Schritt war es, den Dienst in einer *Public-Preview*-Version öffentlich zu machen [Weh14a]. Darauffolgend waren Optimierungen und ein umfangreiches Testen angesagt, das weiter in dem Abschnitt 5.4 beschrieben wurden. Es resultierte in der Freigabe von der Basis-Version für die Kunden [Weh14b]. TFS ASAP Online hat es in die offizielle Liste[23] der VSO-Erweiterungen geschafft [Rü14].

Aufgrund des Zeitdrucks und der oben erwähnten Probleme wurden manche Anforderungen nicht implementiert. Es handelt sich hierbei um die Ausführung von *scheduled* Automations. Die Verwaltungsseite für die Dienst-Benutzer wurde nur rudimentär implementiert. Die Verwaltungsseite für die Dienst-Administratoren wurde nicht implementiert, aber einige von den geforderten Funktionalitäten wurden mit Application Insights sichergestellt.

5.3.2 Deployment

In diesem Abschnitt werden einige Worte der Deployment-Umgebung gewidmet. Während der Migration von TFS ASAP wurden unterschiedliche Konfigurationen von Diensten und Subskriptionen verwendet. Mit Subskriptionen lassen sich die Dienste von Microsoft Azure verwalten [Micj]. In den früheren Entwicklungsphasen wurde meine Subskription für das

[23] Visual Studio Online Integrations: http://www.visualstudio.com/en-us/explore/vso-integrations-directory-vs

Hosting von den Diensten verwendet. Dann wurde eine Subskription von einem anderen Mitarbeiter von AIT wegen eines größeren verfügbaren Kontingents benutzt. Im Moment werden zwei Subskriptionen verwendet, eine für das produktive System und die andere für das Staging-System, in dem eine Version von TFS ASAP Online ausgeliefert wird, die noch vor dem Hosting in der produktiven Umgebung entwickelt oder getestet wird.

Für jede dieser Subskriptionen musste auch die Anwendung für die Benutzung von der Microsoft Account Authentifizierung und VSO REST API OAuth2 Authentifizierungsmechanismen angemeldet sein. Zusätzlich musste das gleiche für die lokale Umgebung geschehen. Die Einstellungen fließen in verschiedene Baukonfigurationen der Anwendung ein. Das gleiche galt für die Ablage der Verbindungszeichenkette[24] der Datenbank und der Azure Service Bus Queue.

Für alle Subskriptionen waren folgende Dienste zu deployen:

- Azure Service Bus Queue: wird manuell im Azure Management Portal angelegt.
- Azure SQL Database: wird manuell im Azure Management Portal angelegt und das Datenmodell wird mittels Code First Migrations von Visual Studio aus aktualisiert..
- Azure Cloud Service: wird von Visual Studio aus deployed.
  - Worker Role: In den Skripten, die bei der Einrichtung des Dienstes ausgeführt werden, werden die Einstellungen für den Internet Information Services (IIS) und ApplicationInsigths vorgenommen. Für den IIS wird die Ausführung von 32-Bit Applikationen aktiviert, und das Ausschalten des Dienstes wenn er lange nicht aktiv ist deaktiviert.
  - Web Role: In den Skripten, die bei der Einrichtung des Dienstes ausgeführt werden, werden die Einstellungen für ApplicationInsights vorgenommen und das TFS Object Model wird installiert.
- Azure Storage: für Application Insigts. Wird von Visual Studio aus automatisch erstellt.

5.4 Optimierung und Testen

Während dieser Phase führte man Aktivitäten durch, die als Ziel hatten, Fehler und Optimierungspotenziale zu identifizieren. An erster Stelle wurde die Worker Role mit dem VSO-Account von AIT getestet. Das heißt, dass nur die Worker Role in Betrieb war, die einige Automations für laufende Projekte von AIT ausführte. Die Benutzer wurden gebeten, alle beobachteten Unstimmigkeiten unverzüglich zu melden. Zu dieser Zeit wurden keine festgestellt.

Später wurde die Anwendung, die bereits über das Frontend und das Backend verfügte, einer kleinen Menge von Benutzern bereitgestellt, die sich aus den Mitarbeitern und den Partnern

[24] Connection String

von AIT zusammensetzte. Dabei sollten ebenfalls die aufgefallenen Probleme gemeldet werden. So wurden einige Fehler bezüglich der Benutzeroberfläche und Funktionalität des Portals gemeldet. Dazu wurden noch neue Wünsche identifiziert, die in das Anforderungsdokument einfließen. Einige dieser Fehler und Optimierungen sind unten aufgelistet:

- Der Portal-Aufruf dauerte unter Umständen sehr lange. Es wurde herausgefunden, dass die Web Role von der Azure-Plattform ausgeschaltet wird, wenn sie über längere Zeit inaktiv bleibt. Das Problem wurde behoben, indem beim Deployment der Web Role entsprechende Skripte ausführen ließ, die diese Funktionalität deaktivieren.
- Synchronisieren von Collections und Projekten eines VSO-Account über VSO REST API, falls inzwischen neue hinzugefügt wurden.
- Optimierung des Kontrollflusses bei der Anmeldung.
- Hilfe-Anzeige mit Erklärung der Automations.
- Erweiterung der Benutzerverwaltung (Funktionalitäten wie Löschen der Benutzer, der Tenants).
- Beheben der Probleme mit den Abfragen über VSO REST API. Der Token wurde zu ehe bezogen und der REST-Abruf erfolgte mit einem nicht mehr gültigen Token. Das Beziehen des Tokens wird auf einen späteren Zeitpunkt verschoben.

Neben der Tests und Optimierungen wurden auch Loadtests definiert und beschrieben. Die Ausführung der Tests konnte jedoch nicht von mir durchgeführt werden, denn dies hätte den Zeitrahmen dieser Diplomarbeit gesprengt. Die Tests wurden teilweise implementiert und ausgeführt von einem anderen AIT-Mitarbeiter, jedoch mit meiner Unterstützung. Zur Erinnerung, jede Änderung eines Work Items führt zu einer Nachricht in der Queue. Diese Nachrichten werden durch Worker Role Instanzen abgearbeitet und dabei die aktivierten Regeln ausgeführt. Abhängig von der Länge der Queue werden zusätzliche Worker Role Instanzen aktiviert. Für die Loadtests wurde folgende Fragen zur Beantwortung definiert:

- Wo liegt der Grenzwert den das System bewältigen kann. Es sollte die Abhängigkeit zwischen der Anzahl von Work Item Änderungen, der Anzahl von VSO-Accounts, der Anzahl von Worker Role Instanzen und der Warteschlangenlänge.
- Wie lang dauert die Abarbeitung einer Regel in Abhängigkeit von der Anzahl der Work Item Änderungen.
- Wie viele Automations werden ausgeführt in einer Zeiteinheit?

Zuerst sollte nur der aktuelle Stand ermittelt werden, ohne Optimierungen vornehmen zu müssen. Der Test soll vollautomatisch ablaufen. Es war ausreichend, wenn der Test manuell gestartet werden konnte. Die Daten von einzelnen Testläufen sollten vergleichbar sein um später eine Optimierung zu ermöglichen.

Zur Implementierung der Tests wurden Lösungsideen vorgeschlagen. An erster Stelle sollten zehn VSO-Accounts mit zehn Projekten manuell angelegt werden. Dann waren neue Work

Items mit der TFS API anzulegen und zu manipulieren. Die Anzahl der pro Sekunde manipulierten Work Items sowie die Queue Länge sollten in einer Datei weggeschrieben und eine Chart mit Microsoft Excel erzeugt werden. Zu diesem Zweck soll Service Bus REST API[25] und Azure Runtime and Diagnostic Library[26]. In der Testmethode sollten drei Threads gestartet werden, die parallel Work Items verändern, die Informationen aus der Azure Service Bus abfragen und den Zustand von Worker Roles ermitteln. Die Tests wurden teilweise implementiert und ausgeführt. Aufgrund von Zeitengpässen wurden weitere Arbeiten momentan eingestellt.

5.5 Betrieb und Verwaltung

In der letzten Phase der Migration soll ein Monitoring-Mechanismus zur Verwaltung der Anwendung implementiert werden. Anders als in [CTG+12] empfohlen, hat man sich in diesem Fall für Application Insights[27] auf Kosten von Azure Diagnostics entschieden. Grund dafür war, dass der Einsatz dieses Werkzeugs recht simpel ist und es eine gut erklärte Einleitung dafür gibt[28]. Mit Application Insights wird eine Reihe von Werkzeugen, die Funktionalitäten zur Performanz-, Verfügbarkeit- und Benutzerüberwachung bereitstellen, angeboten. Die bei TFS ASAP Online eingesetzte Version wird aus Visual Studio Online aus benutzt. Die aktuelle Version wird jedoch von Microsoft Azure Management Portal aus benutzt.[29].

Application Insight wurde lediglich für die Web Role benutzt, denn die Funktionalität für Worker Role steht noch nicht zur Verfügung[30]. Für einen Azure Cloud Service sieht die Installation von Application Insights folgendermaßen aus. Man muss Application Insights Tools for Visual Studio 2013 herunterladen und installieren[31]. Danach sind *Application Insights Telemetry*-Einstellungen bei dem Projekt in Visual Studio anzupassen und das *Azure PaaS*-Werkzeug für Application Insights ist zu installieren. Darauffolgend müssen einige Dateien dem Projekt in den Folder *AppInsightAgent* hinzugefügt werden. Das sind ein PowerShell-Skript und eine Batch-Datei. Ihre Einstellungen sind so anzupassen, dass sie ins Build-Verzeichnis beim Bauen kopiert werden. Dann sollte die .csdef-Konfigurationsdatei des Azure Cloud Service angepasst werden, indem ein neuer Eintrag hinzugefügt wird. Der Eintrag ist in dem Code-Ausschnitt 5.2 dargestellt.

[25] Service Bus REST API Reference: http://msdn.microsoft.com/en-us/library/azure/hh780717.aspx
[26] Azure Runtime and Diagnostic Library Reference for .NET http://msdn.microsoft.com/en-us/library/dn511024.aspx
[27] Application Insights: http://www.visualstudio.com/en-us/explore/application-insights-vs.aspx
[28] Application Insights for Visual Studio Online: http://msdn.microsoft.com/en-us/library/dn481095.aspx
[29] Application Insights for Visual Studio Online: http://msdn.microsoft.com/en-us/us-en/us-en/library/dn481095.aspx
[30] text
[31] Application Insights Tools for Visual Studio: https://visualstudiogallery.msdn.microsoft.com/82367b81-3f97-4de1-bbf1-eaf52ddc635a

```
<WebRole>
  <Startup>
    <Task commandLine="AppInsightsAgent\UnifiedBootstrap.bat"
       executionContext="elevated" taskType="simple">
    </Task>
    ...
  </Startup>
  ...
<\WebRole>
```

Listing 5.2: Start-up Taks für Application Insights

Nach dem wiederholten Bauen und Deployment in Microsoft Azure sind die Monitoring-Daten bei VSO zu sehen. Es werden Daten wie durchschnittliche Verfügbarkeit, Antwortzeit, CPU-Auslastung, Anzahl von laufenden Instanzen, geographische Benutzerverteilung oder die durch die Anwendung geworfenen Ausnahmen in Form von Dashboards bereitgestellt, die den Anwendungsbetreibern bei der Applikationsüberwachung, und Identifizierung und Behebung von Fehlern helfen. So konnte zum Beispiel die Ursache einer fehlerhaften Funktionsweise des Portals herausgefunden werden, denn dank Applications Insights konnte die geworfenen Ausnahmen eingesehen werden. Der Fehler bestand darin, dass in dem Fall, wenn man zwei VSO-Accounts hinzugefügt und dann den ersteren gelöscht hat, wurden keine VSO-Accounts angezeigt. Die Ursache lag im alten Quellcode von TFS ASAP, wo davon ausgegangen wurde, dass es nur einen TfsServer (VSO-Account bei TFS ASAP Online) gibt. Aus diesem Grund wurde in diesem Fall eine *NullReferenceException* geworfen. Eine entsprechende Anpassung der Logik hat den Fehler behoben.

Application Insights unterstützt im Moment Worker Roles nicht[32], deswegen konnte diese Technologie zur Überwachung der Worker Roles nicht eingesetzt werden. Um die Applikation besser kontrollieren zu können, konnte jedoch der für die Automations bereits implementierte Logging-Mechanismus verwendet werden, der bei On-Premise TFS ASAP Informationen, Warnungen und Ausnahmen in eine Text-Datei schreibt. Man musste hier nur einen Storage Account in Microsoft Azure anlegen und die Konfigurationsdatei so anpassen, dass der Logging-Mechanismus die Informationen in den neu angelegten Storage Account schreibt. Die Konfigurationsdatei der Worker Role sollte ebenso um die Information über den Storage Account erweitert werden. Der Logging-Mechanismus für die Automations berücksichtigt jedoch keine Informationen über den Tenant, wenn diese im Kontext eines VSO-Accounts ausgeführt werden. Aus diesem Grund wurde das Loggen dieser Informationen in die Ereignis-Bearbeitungsmethode der Worker Role eingebaut. Auf diese Weise werden auch Informationen protokolliert wie:

- Auf welche VSO-Account, Collection, Project und Work item bezieht sich das eingetroffene Ereignis

[32]New agent for Application Insights available: http://blogs.msdn.com/b/visualstudioalm/archive/2014/04/16/new-agent-for-application-insights-available.aspx

- Zu welchen Tenant gehört der gegebene VSO-Account
- Wenn irgendwelche Ausnahmen während der Abarbeitung einer Automation geworfen werden, werden sie auch protokolliert.

Die Anwendung befindet sich in Betrieb und auf jegliche Probleme wird reagiert. Zusätzlich werden auch sowohl die nicht umgesetzten, als auch die neu dazugekommenen Anforderungen implementiert. Dazu zählen zum Beispiel Funktionalitäten wie die scheduled Automations oder auch die Unterstützung für Microsoft Azure Active Directory zur Benutzerverwaltung. Die Funktionalitäten werden jedoch von anderen Mitarbeitern der Firma AIT entwickelt und deswegen werden sie in dieser Diplomarbeit nicht weiter betrachtet.

6 Diskussion der Ergebnisse

In diesem Kapitel erfolgt die Diskussion der im Rahmen dieser Diplomarbeit erhaltenen Ergebnisse. Dieses Kapitel setzt sich aus drei Teilen zusammen. Im ersten Teil werden alle während der Migration gesammelten Daten dargestellt. Im weiteren Teil dieses Kapitels werden diese Daten analysiert. Zum Ende des Kapitels werden aus den Erfahrungen und Erkenntnissen, die im Laufe des Projektes gemacht wurden, Empfehlungen für künftige Cloud-Migrationen, sowie für die Entwicklung von Anwendungen unter Berücksichtigung von möglichen Cloud-Migrationen in weiteren Verlauf des Anwendungslebenszyklus, abgeleitet.

6.1 Präsentation der erhobenen Daten

In diesem Abschnitt werden die Daten dargestellt, die für alle Phasen der Migration von TFS ASAP gesammelt wurden. Die Daten wurden erst im Nachhinein erhoben, deswegen können sie unvollständig und ungenau sein. Der Grund für diese Vorgehensweise ist der Zeitdruck, und die Anforderung seitens des Managements von AIT, die Migration schnell durchzuführen. Für jede Phase ist eine Tabelle zu sehen, in der die SOLL- und IST-Zeiten für einzelne Meilensteine erfasst wurden. Aus den zwei Werten wurde auch die Differenz errechnet. Im weiteren wird für jede Phase eine Tabelle mit erfassten Daten für die Metriken aufgeführt, wie im Abschnitt 4.6 beschrieben. Die Umsetzungszeit wird aus den IST-Zeiten der Meilensteine abgeleitet und bedeutet die Zeit, die von dem Arbeitsbeginn an einem Meilenstein bis zum Arbeitsende an diesem verlief. Das heißt, dass während dieser Zeit nicht unbedingt nur an dem einzelnen Meilenstein gearbeitet wurde. Für die Metriken Lernbarkeit, Verständlichkeit, Schwierigkeitsgrad und Zufriedenheit erfolgte eine kurze, textuelle Beschreibung. Darüber hinaus werden auch die Tabellen mit erhobenen Daten für Ereignisse aufgeführt. Da bereits genannte Sachverhalte in Beschreibungen von Ereignissen vorkommen können, werden entsprechende Verweise auf frühere Teile der Diplomarbeit gemacht, in welchen sie erklärt sind. Berücksichtigt wurden nur ausgewählte Ereignisse. Das Auswahlkriterium war der subjektive Einfluss der Ereignisse auf den Fortschritt der Migration.

6.1.1 Analyse und Planung

Zuerst wurden die Daten für die Phase Analyse und Planung erhoben. In der Tabelle 6.1 wird der Zeitplan für die Erreichung der Meilensteine gezeigt.

Die erhobenen Daten für diese Phase sind in der Tabelle 6.2 zu sehen.

Meilenstein	SOLL-Wert [KT]	IST-Wert [KT]	Differenz [KT]
Migrationsplan	0,5	1	0,5
Microsoft Azure Überblick	1	2	1
Prototyp	1	2,5	1,5
Anforderungsdokument	2	2	0
Anwendungsanalyse	2	3	1
Redesign	3	4	1
Kostenplan	0,5	0,5	0
Summe:	**10**	**15**	**5**

Tabelle 6.1: Zeitplan für die Meilensteine der Phase 1: Analyse und Planung

Die Metriken wurden aus den Daten errechnet, die während der Umsetzung einzelner Meilensteine erhoben wurden. Diese Daten wurden aggregiert und werden im weiteren unten aufgeführt.

Im Grunde genommen waren für diese Phase nicht viele Vorkenntnisse erforderlich. Diese Phase wurde so entwickelt, dass die Wissensdefizite im Rahmen dieser Phase eliminiert werden sollten. Aufgrund von begrenzten Erfahrung in Projektverwaltung und dem nicht immer ausreichendem Wissen bezüglich der Funktionsweise von TFS ASAP und fehlenden Überblick über Microsoft Azure fällt die Benotung jedoch nicht perfekt aus. Dazu musste noch die Bedienung von den eingesetzten Werkzeugen gelernt werden. Die Ziele für einzelne Aufgaben waren überwiegend klar und verständlich formuliert. Eine Ausnahme ist hier die Aufgabe *Anforderungsanalyse- und -spezifikation*. Hier wären Richtlinien wünschenswert, in welchen festgelegt wird, was berücksichtigt werden sollte. Die Durchführung der Phase war ziemlich schwierig. Gründe dafür sind: kaum Erfahrung in Projekt Planung, nur geringes, theoretisches Wissen im Bereich Cloud Computing und Microsoft Azure, und teilweise mangelhafte, verteilte Dokumentation von TFS ASAP und die Komplexität und nicht immer saubere Architektur der Anwendung. Es ergab sich, dass man mit der Durchführung aller Aufgaben zufrieden war. Leider wurden bei den meisten Aufgaben die Zeitschätzungen nicht eingehalten, deswegen dauerte die Umsetzung dieser Phase länger als angenommen. Aus diesem Grund wird ein Punkt abgezogen.

Die aufgezeichneten Ereignisse für diese Phase sind in der Tabelle 6.3 zu finden.

Metrik	Eigenschaft
Geplante Zeit [KW]	2
Umsetzungszeit [KW]	3
Verzögerung [KW]	1
Anzahl von Ereignissen/-Problemen/Rücksprüngen	1 / 0 / 1
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	0,333 / 0,000 / 0,333
Anzahl der Meilensteine	7
TTO	0,286
ADMC [Meilenstein/KW]	0,1433
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: 2,5
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: 2
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: 3
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: 4

Tabelle 6.2: Die erhobenen Daten aus der Phase 1: Analyse und Planung

Eigenschaft	Wert
ID	1-Z1
Klasse	Rücksprung
Name	Verlängerung der Dauer der Phase 1
Beschreibung	Die angenommene Dauer für die Phase konnte nicht mehr eingehalten werden.
Lösung	Der geplante Umsetzungsdauer für diese Phase wurde verlängert.
Anpassung	siehe oben
Kommentar	Keine Kommentar

Tabelle 6.3: Ereignis: 1-Z1

6.1.2 Proof of Concept

In der Tabelle 6.4 wird der Zeitplan für die Erreichung der Meilensteine der Phase Proof of Concept angezeigt.

Meilenstein	SOLL-Wert [KT]	IST-Wert [KT]	Differenz [KT]
Automationsausführung	5	5	0
Anmeldung mit Microsoft Account	2	4	2
VSO REST API	3	6	3
	10	**15**	**5**

Tabelle 6.4: Zeitplan für die Meilensteine der Phase 2: Proof of Concept

Die erhobenen Daten für diese Phase sind in der Tabelle 6.5 zu sehen.

Metrik	Eigenschaft
Geplante Zeit [KW]	2
Umsetzungszeit [KW]	3
Verzögerung [KW]	1
Anzahl von Ereignissen/-Problemen/Rücksprüngen	3 / 2 / 1
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	1,000 / 0,667 / 0,333
Anzahl der Meilensteine	3
TTO	0,667
ADMC [Meilenstein/KW]	0,33
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: 4
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: 1
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: 3,5
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: 4

Tabelle 6.5: Die erhobenen Daten aus der Phase 2: Proof of Concept

Hier waren viele Kenntnisse erforderlich. Da diese fehlten, mussten diese während dieser Phase gewonnen werden. Dazu zählen unter anderem: Deployment von Azure Diensten, ASP.NET MVC, REST API mit HTTP-Aufrufen, Identity Management (Microsoft Account), TFS API und technische Einzelheiten von TFS ASAP. Die Aufgaben wurden klar und verständlich formuliert. Diese Phase war relativ schwer durchzuführen, vor allem aufgrund von mangelnder Erfahrung in den oben genannten Bereichen. Man war mit der Durchführung der Aufgaben dieser Phase ziemlich zufrieden, jedoch wie früher konnte der Zeitplan nicht eingehalten werden.

Die aufgezeichneten Ereignisse für diese Phase sind in den Tabellen 6.6, 6.7. und 6.8 zu finden.

Eigenschaft	**Wert**
ID	2-P1
Klasse	Problem
Name	Bereitstellung der TFS API für die Worker Role
Beschreibung	Die Worker Role braucht die TFS API Bibliothek, um die Automations ausführen zu können. Die Versuche, die Bibliotheken in dem Worker-Role-Projekt zu referenzieren und in der Cloud auszuliefern, blieben erfolglos. Immer wieder kamen Meldungen, dass Bibliotheken noch fehlen.
Lösung	TFS API musste zusätzlich installiert werden.
Anpassung	Eine Installationsdatei für TFS API musste dem Worker-Role-Projekt als Ressource beigefügt werden. In der Einstellungsdatei der Worker Role musste ein neuer Task definiert werden, der das TFS Object Model bei der Aktivierung der Worker Role installiert.
Kommentar	Diese Vorgehensweise wurde von dem Projekt TFS OData API übernommen. Siehe dazu Abschnitt 5.2.

Tabelle 6.6: Ereignis: 2-P1

Eigenschaft	Wert
ID	2-P2
Klasse	Problem
Name	Authentifizierung bei VSO REST API
Beschreibung	Die Authentifizierung bei VSO REST API funktionierte nicht, obwohl man der Anleitung auf der Seite von Microsoft folgte.
Lösung	Man musste den Authentication-Header des Http-Requests ersetzen. Statt *Authorization: Authorization: bearer access_token* sollte *Authorization: Bearer access_token* verschickt werden.
Anpassung	Die fehlerhafte Dokumentation war ein Anlass dafür, einen Blog-Beitrag mit der Problemlösung und Anpassungsmöglichkeiten zu verfassen. Die Vorgehensweise wurde dort beschrieben [Sab14a].
Kommentar	Der Fehler auf der Seite von Microsoft wurde nach der Meldung behoben.

Tabelle 6.7: Ereignis: 2-P2

Eigenschaft	Wert
ID	2-Z1
Klasse	Rücksprung
Name	Verlängerung der Dauer der Phase 2
Beschreibung	Die angenommene Dauer für die Phase konnte nicht mehr eingehalten werden.
Lösung	Die geplante Umsetzungsdauer für diese Phase wurde verlängert.
Anpassung	siehe oben
Kommentar	Keine Kommentar

Tabelle 6.8: Ereignis: 2-Z1

6.1.3 Anwendungsmigration

In der Tabelle 6.9 wird der Zeitplan für die Erreichung der Meilensteine angezeigt. Die erhobenen Daten für diese Phase sind in der Tabelle 6.10 zu sehen.

Hier waren viele Kenntnissen erforderlich. Vor allem waren das die Kenntnisse, die auch für die Phase Proof of Concept von Relevanz waren. Das gewonnene Know-How musste jedoch bedeutend erweitert werden. Die Aufgaben waren nicht immer klar formuliert. Die Anforderungen für die zu implementierenden Funktionalitäten mussten noch verfeinert werden. Diese Phase war schwer durchzuführen, vor allem aufgrund von mangelnder Erfahrung in den oben genannten Bereichen und sich ständig verändernden Anforderungen. Grund dafür waren auch die Vielfältigkeit der zu implementierenden Funktionalitäten. Man war mit der Durchführung der Aufgaben dieser Phase nicht ganz zufrieden. Zum einem konnte der Zeitplan nicht eingehalten werden. Zum anderen war diese Phase schwierig. Die geforderte

Basisfunktionalität wurde jedoch umgesetzt.

Die aufgezeichneten Ereignisse für diese Phase sind in den Tabellen 6.11, 6.12, 6.13, 6.14, 6.15, 6.16 und 6.17 zu finden.

Meilenstein	SOLL-Wert [KT]	IST-Wert [KT]	Differenz [KT]
Anmeldung bei TFS ASAP Online	4	8	4
Hinzufügen eines VSO Accounts	3	5	2
Triggered Automations aktivieren	2	3	1
Triggerred Automations ausführen	4	10	6
Anwendungsportal anpassen	3	4	1
Web GUI für Administratoren	4	5	1
	20	**35**	**15**

Tabelle 6.9: Zeitplan für die Meilensteine der Phase 3: Anwendungsmigration

Metrik	Eigenschaft
Geplante Zeit [KW]	4
Umsetzungszeit [KW]	7
Verzögerung [KW]	3
Anzahl von Ereignissen/-Problemen/Rücksprüngen	7 / 1 / 6
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	1,000 / 0,143 / 0,857
Anzahl der Meilensteine	6
TTO	0
ADMC [Meilenstein/KW]	0,5
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: 3
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: 2
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: 4
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: 3,5

Tabelle 6.10: Die erhobenen Daten aus der Phase 3: Anwendungsmigration

Eigenschaft	**Wert**
ID	3-P1
Klasse	Problem
Name	Die Änderung des Felds *State* eines Work Items mit VSO REST API schlägt unter Umständen fehl.
Beschreibung	Man konnte den *State* eines Work Items mit VSO REST API nicht immer ändern. Eine Http-Antwort wurde zurückgesendet, die auf einen fehlerhaften REST-Aufruf hinwies.
Lösung	Die Fehlfunktion war ein Anlass dafür, einen Blog-Beitrag mit der Problemursachem und -lösung, und Anpassungsmöglichkeiten zu verfassen. Die Vorgehensweise wurde dort beschrieben [Sab14b].
Anpassung	Zusätzlich mussten in den Projektdateien von TFS ASAP Online die Definitionen für Process Templates abgelegt werden, wo sich Informationen über erlaubte Transitionen für Work Items befinden. Aus diesen Definitionen wurden die zur *State*-Änderung notwendige Felder ermittelt. Diesen wurden entsprechende Werten zugewiesen und mit einem REST-Aufruf verschickt.
Kommentar	Inzwischen wurde die VSO REST API aktualisiert, und die oben genannte Vorgehensweise findet keine Verwendung mehr. Darüber hinaus wird im Moment die TFS API zur Manipulation von Work Items bei TFS ASAP Online eingesetzt, wo so ein Problem nicht existiert . Mehr Einzelheiten dazu findet man im Abschnitt 5.3.

Tabelle 6.11: Ereignis: 3-P1

Eigenschaft	Wert
ID	3-Z1
Klasse	Rücksprung
Name	Änderung der Technologie zur Manipulation von Work Items auf VSO REST API
Beschreibung	Am Anfang wurde TFS API für die Änderungen von Work Items eingesetzt. Die Authentifizierung stellte bei dieser Lösung ein Problem dar, wenn Login und Passwort des Benutzers an TFS ASAP übergeben werden mussten.
Lösung	Aus dem obigen Grund wurde zu VSO REST API gegriffen, die die OAuth-Authentifizierung bereitstellte. Auf diese Weise wurde das Problem eliminiert.
Anpassung	Der gesamte Quellcode musste so umgebaut werden, dass die Abhängigkeiten von TFS API mittels Schnittstellen abstrahiert wurden. Man implementierte die Schnittstellen mit VSO REST API.
Kommentar	Die VSO REST API wird im Moment nicht mehr zu Manipulationen von Work Items eingesetzt. Der Code ist jedoch erhalten geblieben. Mehr Einzelheiten dazu findet man im Abschnitt 5.3.

Tabelle 6.12: Ereignis: 3-Z1

Eigenschaft	Wert
ID	3-Z2
Klasse	Rücksprung
Name	Hosting des Portals als Web Site
Beschreibung	Für das Hosting des Portals war zuerst eine Web Role vorgesehen. Laut der Aussage eines AIT-Mitarbeiters, würden die Web Roles durch die Microsoft Azure Plattform bei langer Untätigkeit ausgeschaltet, um Ressourcen zu sparen. Dies sei nicht der Fall bei Web Sites, bei welcher eine Option *Always on* vorhanden ist, die dieses Verhalten eliminiert.
Lösung	Noch vor der Auslieferung der Anwendung wurde die Entscheidung geändert, dass das Portal als Web Site gehostet wird.
Anpassung	Bis auf die Änderung einiger Stellen in der Dokumentation mussten keine Änderungen vorgenommen werden. Die Auslieferung erfolgte von Visual Studio aus. Dank des eingebauten Assistenten verlief es einwandfrei.
Kommentar	Neben der Auslieferung einer Web Site ermöglicht der Assistent eine gleichzeitige Erstellung einer Datenbank, was in Anspruch genommen wurde.

Tabelle 6.13: Ereignis: 3-Z2

Eigenschaft	Wert
ID	3-Z3
Klasse	Rücksprung
Name	Änderung der Technologie zur Manipulation von Work Items auf TFS API.
Beschreibung	In der *History*-Feld von Work Items erscheint der Benutzer als Verursacher der Änderung, wenn sie mit VSO REST API gemacht wird. Es sollte unterscheidbar sein, ob eine Änderung durch den Benutzer selbst oder durch einen Dienst (in unserem Fall TFS ASAP Online) verursacht wird.
Lösung	Man stieg erneut auf TFS API um und man bezog den Service Account, um sicht mit diesem gegenüber VSO zu authentifizieren.
Anpassung	Der alte Zustand des Quell-Codes musste wiederhergestellt werden. Das Datenmodell musste um Information für *Service Account* erweitert werden. Die Web-GUI war ebenso anzupassen.
Kommentar	Es wird immer noch gefordert, dass der Nutzer einmalig Alternate Credentials bei VSO aktiviert und seine Credentials übergibt. Einzelheiten befinden sich im Abschnitt 5.3.

Tabelle 6.14: Ereignis: 3-Z3

Eigenschaft	Wert
ID	3-Z4
Klasse	Rücksprung
Name	Verschiebung der Implementierung der Funktionalität "Monitoring und Logging"
Beschreibung	Die Implementierung des Monitoring- und Logging-Mechanismus für TFS ASAP Online wurde zunächst für die Phase 3 geplant. Es wurde jedoch dann auf die Phase 4 verschoben.
Lösung	Siehe Beschreibung
Anpassung	Siehe Beschreibung
Kommentar	Die Entwicklung der Methodologie wurde noch während der Migration fortgesetzt und Änderungen an der Methodik wurden vorgenommen. Aus diesem Grund kam die Planänderung.

Tabelle 6.15: Ereignis: 3-Z4

Eigenschaft	Wert
ID	3-Z5
Klasse	Rücksprung
Name	Verlängerung der Dauer der Phase 3
Beschreibung	Die angenommene Dauer für die Phase konnte nicht mehr eingehalten werden.
Lösung	Der geplante Umsetzungsdauer für diese Phase wurde verlängert.
Anpassung	siehe oben
Kommentar	Keine Kommentar

Tabelle 6.16: Ereignis: 3-Z5

Eigenschaft	Wert
ID	3-Z6
Klasse	Rücksprung
Name	Verzicht auf die Implementierung von Scheduled Automations
Beschreibung	Während der Durchführung der Phase *Anwendungsmigration* stellte man fest, dass der angenommene Zeitplan nicht eingehalten werden konnte.
Lösung	Die Lösung war Verzicht der Implementierung von Scheduled Automations und ihre Verschiebung für die künftige Entwicklungsarbeiten an TFS ASAP Online.
Anpassung	Aufgrund des Verzichts konnten andere Anforderungen realisiert werden. Der Zeit- und Meilensteinplan mussten korrigiert werden.
Kommentar	Die Scheduled Automations befinden sich momentan in Entwicklung. Sie werden von einem anderen AIT-Mitarbeiter entwickelt.

Tabelle 6.17: Ereignis: 3-Z6

6.1.4 Optimierung und Testen

In der Tabelle 6.18 wird der Zeitplan für die Erreichung der Meilensteine angezeigt.

Meilenstein	SOLL-Wert [KT]	IST-Wert [KT]	Differenz [KT]
Getestete und optimierte Preview-Version	10	10	0
	10	**10**	**0**

Tabelle 6.18: Zeitplan für die Meilensteine der Phase 4: Optimierung und Testen

Die erhobenen Daten für diese Phase sind in der Tabelle 6.19 zu sehen.

Metrik	Eigenschaft
Geplante Zeit [KW]	2
Umsetzungszeit [KW]	2
Verzögerung [KW]	0
Anzahl von Ereignissen/-Problemen/Rücksprüngen	2 / 0 / 2
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	1 / 0 / 1
Anzahl der Meilensteine	1
TTO	1
ADMC [Meilenstein/KW]	0
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: 3
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: 1
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: 4
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: 3

Tabelle 6.19: Die erhobenen Daten aus der Phase 4: Optimierung und Testen

In Bezug auf den Test selbst waren nur wenige Kenntnisse vorhanden. Zur Konzipierung der Load-Tests fehlten Kenntnisse, vor allem bezüglich der Benutzung von Azure Management API. Die Aufgabe wurden klar und verständlich formuliert. Die Tests durchzuführen und Fehler zu beheben war nicht schwierig. Konzipierung von Load-Tests war ebenfalls einfach. Jedoch die Implementierung und Durchführung von Load-Tests waren schwierig, was dazu führte, dass sie nicht umgesetzt wurden. Deswegen so eine hohe Bewertung. Man war mit den Tests zufrieden. Leider konnten die Load-Tests nicht so durchgeführt werden, dass sinnvolle Ergebnisse geliefert wurden. Der Zeitplan wurde eingehalten, jedoch wurde nicht der volle Umfang umgesetzt.

Die aufgezeichneten Ereignisse für diese Phase sind in den Tabellen 6.20 und 6.21 zu finden.

Eigenschaft	Wert
ID	4-Z1
Klasse	Rücksprung
Name	Hosting des Portals als Web Role
Beschreibung	Es stellte sich heraus, dass die Option Always on für Web Sites nur für kostenpflichtige Variante von Web Sites verfügbar ist.
Lösung	Man wechselte wieder zu Web Role.
Anpassung	Man musste die Tasks anpassen, die während des Hochfahrens einer Web Role ausgeführt werden. Mehr Einzelheiten dazu im Abschnitt 5.4
Kommentar	keine

Tabelle 6.20: Ereignis: 4-Z1

Eigenschaft	Wert
ID	4-Z2
Klasse	Rücksprung
Name	Verzicht auf die Implementierung und die Durchführung von Load-Tests
Beschreibung	Wegen der Komplexität, des großen Einarbeitungsaufwands und des Zeitdrucks fiel die Entscheidung, dass die Load-Test zuerst nicht implementiert und durchgeführt werden.
Lösung	Die Load-Tests wurden nicht implementiert und durchgeführt.
Anpassung	Mehr Zeit konnte in die anderen Aufgaben der Migrationsphase investiert werden.
Kommentar	Die Load-Tests wurden teilweise implementiert. Ein Durchlauf lieferte suspekte Ergebnisse. Die Ursache wurde nicht analysiert. Die Ergebnisse werden aus diesem Grund in dieser Diplomarbeit nicht aufgeführt.

Tabelle 6.21: Ereignis: 4-Z2

6.1.5 Betrieb und Verwaltung

In der Tabelle 6.22 wird der Zeitplan für die Erreichung der Meilensteine angezeigt.

Meilenstein	SOLL-Wert [KT]	IST-Wert [KT]	Differenz [KT]
Basic-Version mit Monitoring	10	5	-5
	10	**5**	**-5**

Tabelle 6.22: Zeitplan für die Meilensteine der Phase 5: Betrieb und Verwaltung

Die erhobenen Daten für diese Phase sind in der Tabelle 6.23 zu sehen.

Metrik	Eigenschaft
Geplante Zeit [KW]	2
Umsetzungszeit [KW]	1
Verzögerung [KW]	-1
Anzahl von Ereignissen/-Problemen/Rücksprüngen	1 / 0 / 1
Er.-/Pr.-/Rückspr.-dichte [Ereignis/KW]	1 / 0 / 1
Anzahl der Meilensteine	1
TTO	1
ADMC [Meilenstein/KW]	-1
Lernbarkeit	Waren viele Vorkenntnisse erforderlich? 1: keine, 5: sehr viele: 2
Verständlichkeit	Wie einfach konnte diese Phase verstanden werden? 1: sehr einfach, 5: sehr schwer: 1
Schwierigkeitsgrad	Wie einfach war diese Phase durchzuführen? 1: sehr einfach, 5: sehr schwer: 1,5
Zufriedenheitsgrad	Wie zufrieden war man mit der Durchführung der Phase im Nachhinein? 1: sehr zufrieden, 5: sehr unzufrieden: 5

Tabelle 6.23: Die erhobenen Daten aus der Phase 5: Betrieb und Verwaltung

Die Kenntnisse in Methoden zum Monitoring von Azure-Diensten waren erforderlich. Die Aufgabe wurde klar und verständlich formuliert. Die Implementierung von Monitoring- und Logging-Mechanismen war einfach zu implementieren. Die Behebung von aufgefallenen Fehlern ebenso. Man war mit der Phase sehr zufrieden. Das ist die einzige Phase, bei der der Zeitplan nicht nur eingehalten wurde, sondern auch die Umsetzungsdauer kürzer ausfiel als geplant.

Die aufgezeichneten Ereignisse für diese Phase sind in der Tabelle 6.24 zu finden.

Eigenschaft	Wert
ID	5-Z1
Klasse	Rücksprung
Name	Einsatz von Azure Active Directory für Identity Management
Beschreibung	Während der Test-Phase wurde die Anforderung identifiziert, dass TFS ASAP Online auch Azure Active Directory zur Authentifizierung benutzen sollte.
Lösung	Die Anforderung wurde gespeichert.
Anpassung	keine Angaben
Kommentar	Die Funktionalität wurde bereits implementiert, jedoch nicht im Rahmen dieser Diplomarbeit, sondern von einem anderen AIT-Mitarbeiter.

Tabelle 6.24: Ereignis: 5-Z1

6.2 Analyse und Diskussion der erhobenen Daten

In diesem Abschnitt werden die während der Migration von TFS ASAP erhobenen Daten diskutiert und analysiert. Aufgrund von Zeitmangel konnte die Migration nur mit Hilfe der im Kapitel 4 entwickelten Cloud-Migration-Methodologie durchgeführt werden. Aus diesem Grund konnte die Migrationsmethodologie nicht so evaluiert werden, dass ein Vergleich zwischen den in beiden Migrationen gesammelten Daten erfolgt. Die Durchführungsdauer der Migration wurde am Anfang für 12 Kalenderwochen angesetzt, mit einer Pufferzeit von 16 Kalenderwochen. Wie die Praxis zeigte, musste die gesamte Pufferzeit aufgebracht werden, um eine Basislösung liefern zu können. Der Grund dafür können die Komplexität der zu migrierenden Software und ihr Anpassungsbedarf, der Projektschwierigkeitsgrad, wenig Know-How und Erfahrung in den notwendigen Methoden und Technologien, und auch bestimmt das Parkinsonsche Gesetz [Par57] sein, das lautet:

„Arbeit dehnt sich in genau dem Maß aus, wie Zeit für ihre Erledigung zur Verfügung steht."

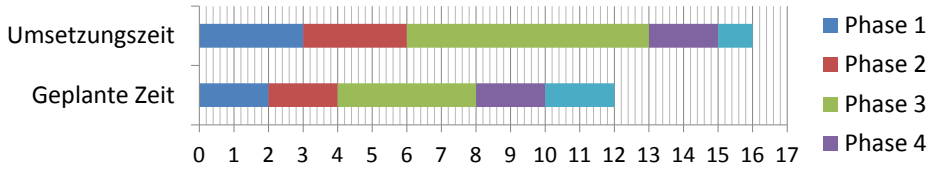

Abbildung 6.1: Der Zeitplan der Migration von TFS ASAP: Oben - die tatsächliche Umsetzungsdauer; Unten - die geplante Umsetzungsdauer

Das Diagramm mit dem geplanten Zeitplan der Migration von TFS ASAP und mit der tatsächlichen Umsetzungsdauer sieht man in der Abbildung 6.1. In dem Diagramm in der Abbildung 6.2 werden die Verzögerungen für jede Phase der Migration von TFS ASAP dargestellt. Wie man dem Diagramm entnehmen kann, traten Verzögerungen für alle drei ersten Phasen der Migration auf, wobei die Verzögerung der Phase 3 am größten war und

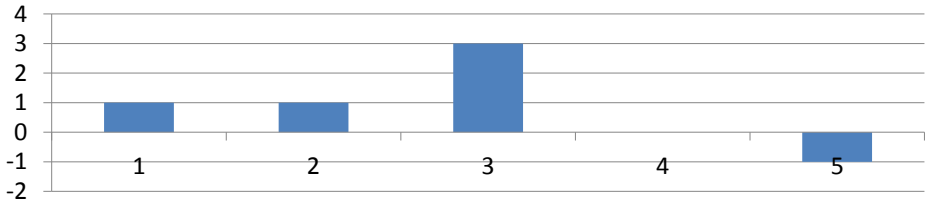

Abbildung 6.2: Die Verzögerungen für jede Phase bei der Migration von TFS ASAP

betrug 3 Wochen, bei jeweils einer Woche für die Phasen *Analyse und Planung* und *Proof of Concept*. Es kann auf die Daten aus dem Diagramm in der Abbildung 4 zurückgeführt werden, in welcher die Anzahl der aufgetretenen Ereignisse in jeder Phase zu sehen ist. Wie man sehen kann, traten in der Phase 3 sieben Ereignisse auf, was genau die Hälfte aller Ereignisse während der Migration ausmacht. Außerdem ist zu betonen, dass sechs von diesen Ereignisse von der Klasse Rücksprung sind. Das kann auf eine mangelhaft durchgeführte Analyse und Planung in der Phase 1 hindeuten. Besonders ausschlaggebend war in der Phase 3 die zweifache Änderung der Technologie, die zu der Manipulation von Work Item einzusetzen war. Die Anpassungsarbeiten, die durch diese Entscheidungsänderungen verursacht wurden, erwiesen sich als besonders zeitraubend.

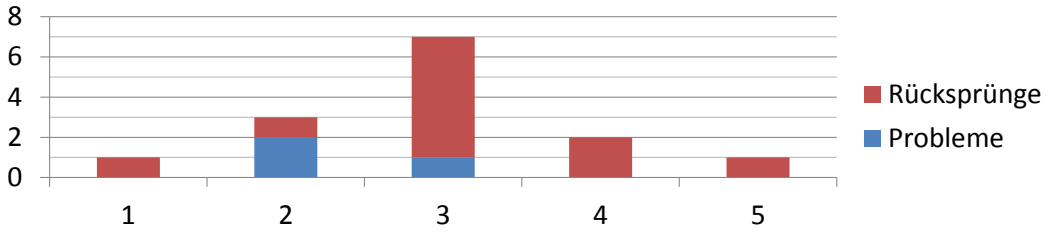

Abbildung 6.3: Anzahl der aufgetreten Ereignisse während jeder Phase der Migration von TFS ASAP

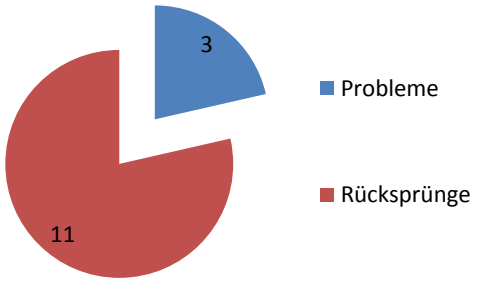

Abbildung 6.4: Anzahl von Problemen und Rücksprüngen während der Migration von TFS ASAP

Zwei von drei Ereignissen der Phase *Proof of Concept* waren Ereignisse der Klasse Problem,

was damit zu rechtfertigen ist, dass diese Phase zum Testen unterschiedlicher Technologien vorgesehen war, und Probleme in dieser Phase waren vorauszusehen. Darüber hinaus trat ein Problem nur in Phase 3 auf. Für zwei während der Migration aufgetretene Probleme wurden Blog-Beiträge mit möglichen Lösungen verfasst und auf dem Blog von AIT veröffentlicht. Das Verhältnis zwischen Anzahl von Problem und Rücksprüngen ist in der Abbildung 6.4 zu sehen.

Abbildung 6.5 zeigt, dass die Ereignisdichte in fast allen Phasen gleich war und eins betrug. Die Ausnahme ist hier nur die erste Phase, wo nur eine Ereignis der Klasse Rücksprung auftrat, was eine Planänderung wegen der Zeitüberschreitung verursachte. Da nur während der Phasen 2 und 3 ein Problem festgestellt wurde, war die Problemdichte nur bei diesen Phasen größer als null. Bei der Durchführung der Phase 4 konnte der Zeitplan eingehalten werden,

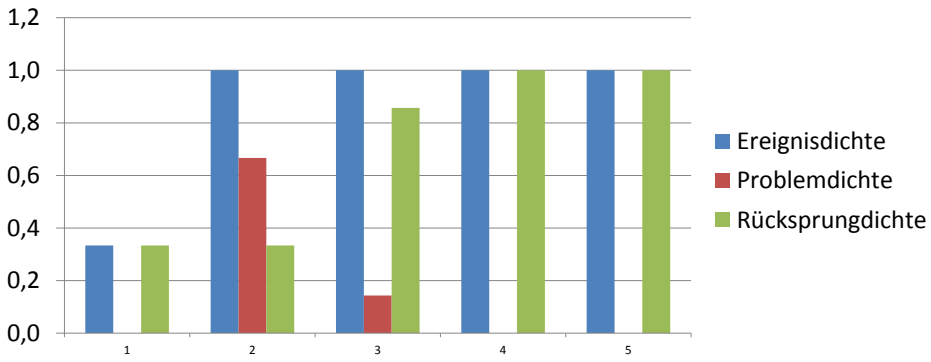

Abbildung 6.5: Ereignisdichte für jede Phase der Migration von TFS ASAP

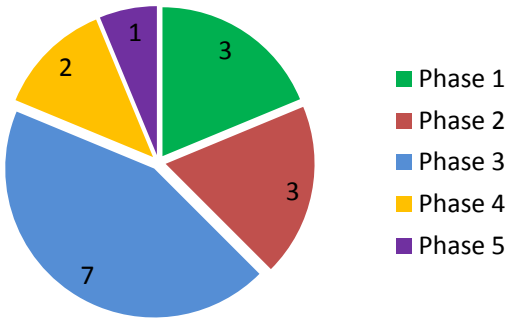

Abbildung 6.6: Umsetzungsdauer der einzelnen Phasen in KW

jedoch die Implementierung und die Durchführung von Load-Tests fand nicht statt. Die Umsetzungdauer der Phase 5 fiel kürzer als geplant aus, und deswegen konnte eine negative Verzögerung festgestellt werden. Es soll allerdings betont werden, dass die Aufgaben der Phase 5 Betrieb und Verwaltung mit dem weiteren Lebenszyklus der Anwendung ausgeführt werden. Wie man der Abbildung 6.6 entnehmen kann, nahm die Durchführung der Phase 3 fast die Hälfte der Zeit ein, die für die Migration von TFS ASAP zur Verfügung stand. Durch

eine genauere Analyse und besser durchdachte Lösungsentscheidungen in der Phase 1, hätte möglicherweise etwas Umsetzungszeit der Phase 3 gespart werden können.

In den Abbildungen sind die Diagramme zu sehen, die die TTO und Durchschnittliche Meilenstein-Verzögerung ADMC für einzelne Phase der Migration von TFS ASAP darstellen. Es fällt auf, dass in der Phase 3 keiner der Meilensteine rechtzeitig erreicht wurde und dass in dieser Phase die durchschnittliche Verzögerung bei der Umsetzung eines Meilensteins 0,5 KW betrug. Eine andere Besonderheit ist, dass für die Phase 4 und 5 alle Meilensteine rechtzeitig erreicht wurden. Das lässt sich damit erklären, dass im Rahmen dieser Phasen jeweils nur ein Meilenstein festgelegt wurde. Bei der Phase 5 fällt außerdem auf, dass der Wert für ADMC negativ ist. Dieser Sachverhalt kommt zustande, weil diese Phase schneller durchgeführt wurde, als geplant.

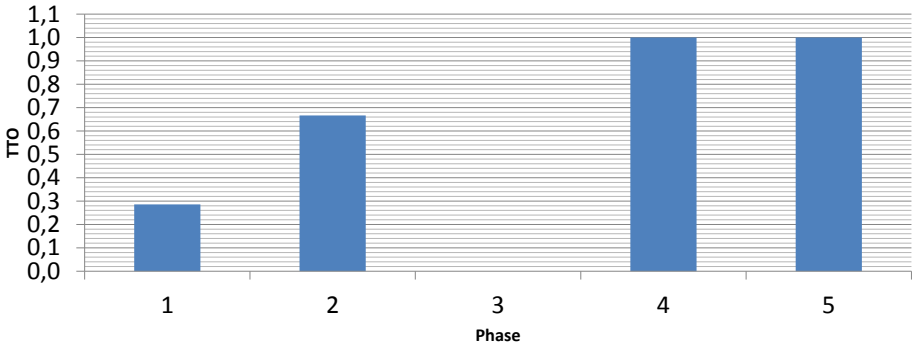

Abbildung 6.7: Prozesszeitplan-Beobachtung TTO für einzelne Phase der Migration von TFS ASAP

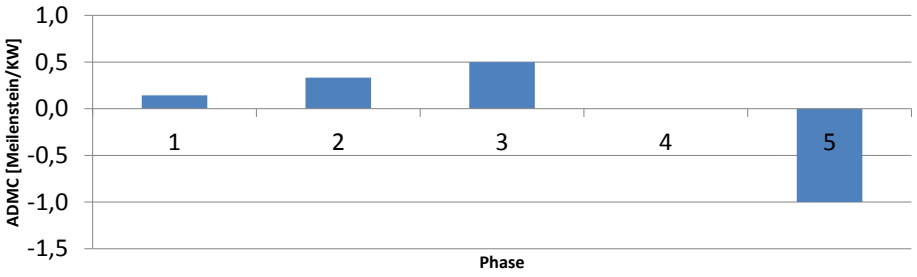

Abbildung 6.8: Durchschnittliche Meilenstein-Verzögerung ADMC für die einzelnen Phasen der Migration von TFS ASAP

In den Abbildung 6.9 werden die Werte für die Metriken Lernbarkeit, Verständlichkeit, Schwierigkeitsgrad und Zufriedenheit für jede Phase angezeigt. In der Tabelle 6.25 sind die durchschnittlichen Werte für die ganze Migration dargestellt.

Prinzipiell waren für alle Phasen nicht viele Vorkenntnisse erforderlich. Wenn das schon der Fall war, basierte dies eher auf den gewonnenen Erfahrungen aus den früheren Phasen. Das heißt, dass für diejenigen, die an einer Migration beteiligt sind, Übergänge in weitere Phasen durch den Lerneffekt relativ einfach. Probleme können jedoch entstehen, wenn jemand in das

Projekt einsteigen möchte, der bisher nicht beteiligt war. Alle Phasen waren verständlich und ihre Aufgaben waren klar formuliert. Die Phase 2, 3 und 4 erwiesen sich als die schwierigsten. Man war mit der Durchführung von allen Phasen zufrieden, allerdings weniger bei den Phasen 3 und 4, denn in diesen Phasen musste auf die Umsetzung mancher Anforderungen verzichtet werden.

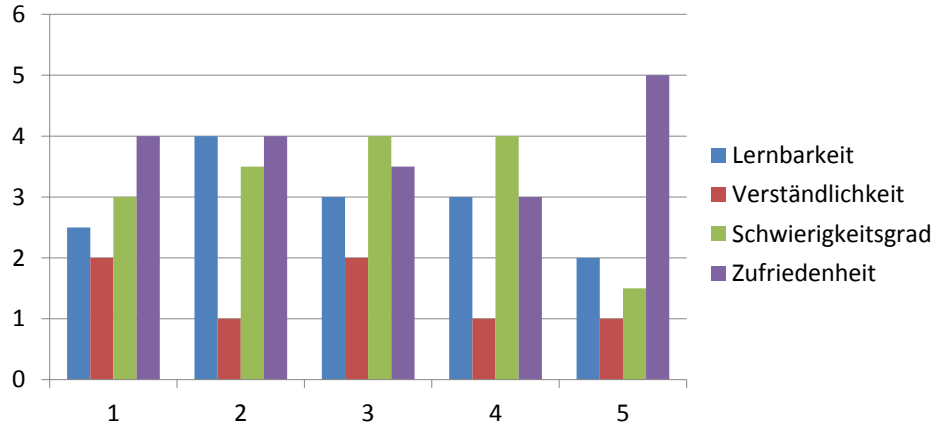

Abbildung 6.9: Die erhobenen Werte für die Metriken Lernbarkeit, Verständlichkeit, Schwierigkeitsgrad und Zufriedenheit für jede Phase

Metrik	durchschn. Wert
Lernbarkeit	2,9
Verständlichkeit	1,4
Schwierigkeitsgrad	3,2
Zufriedenheit	3,9

Tabelle 6.25: Die durchschnittlichen Werte für die Metriken Lernbarkeit, Verständlichkeit, Schwierigkeitsgrad und Zufriedenheit

6.3 Empfehlungen

In diesem Abschnitt werden Empfehlungen dargestellt, die von den Erkenntnissen und Erfahrungen abgeleitet wurden, die während der Erstellung dieser Diplomarbeit gemacht wurden. Die Empfehlungen beziehen sich auf künftige Migrationen von Legacy-Anwendungen wie im Fall von TFS ASAP, sowie auf eine Entwicklung von neuen Anwendungen, die ihre spätere Migration in die Cloud ermöglicht oder vereinfacht. Die beiden Arten von Empfehlungen werden in den zwei folgenden Abschnitten behandelt. Dabei werden zwei Aspekte berücksichtigt: zum einen Prozess-bezogene Empfehlungen, die darauf abzielen, den Migrationsprozess effektiver und effizienter durchzuführen, und zum anderem Empfehlungen, die sich mit den technischen Details der Implementierung auseinandersetzen.

6.3.1 Anwendungsentwicklung

In diesem Unterabschnitt werden Empfehlungen abgeleitet, die Ihre Verwendung finden können, wenn eine On-Premise-Anwendung entwickelt wird. Der Einsatz von diesen Empfehlungen bei der Entwicklung soll eine zukünftige Migration der Anwendung vereinfachen.

Cloud Assessment

Bei der Entwicklung einer Anwendung soll vor allem die Frage beantwortet werden, ob ihre spätere Migration in die Cloud überhaupt Sinn macht. Nicht alle Anwendungsarten können in die Cloud migriert werden und nicht alle geforderten Szenarien lassen sich in der Cloud umsetzen. Zusätzlich können rechtliche und ökonomische Gründe gegen eine Cloud-Migration sprechen. Deswegen besteht der Bedarf, noch in der Planungsphase zu prüfen, ob das der Fall ist und damit ob sich der Aufwand lohnt, die weiteren Empfehlungen zu verwenden, wenn eine Cloud-Migration in der Zukunft ausgeschlossen ist.

Architektur

Die Anwendung sollte eine saubere Architektur aufweisen. Empfehlenswert ist der Einsatz von SOLID-Prinzipien[1] bei dem Design und bei der Entwicklung. Eine Schichten-Architektur, in welcher Applikation nach konzeptionellen Aspekten strukturiert und aufgeteilt ist, wie z.B. 3-Schichten-Architektur [FRF02], ist zu empfehlen. Die Applikation sollte aus eigenständigen Komponenten bestehen. Solche Vorgehensweise wird eine Zuordnung von bestimmten Komponenten zu einzelnen Cloud-Diensten vereinfachen. Die Abhängigkeiten zwischen diesen sowie zu den externen Komponenten sollten durch Schnittstellen abstrahiert sein, um ein einfaches Umschalten zwischen unterschiedlichen Implementierungen zu vereinfachen und die Testbarkeit der Komponenten zu erhöhen.

Dokumentation

Die Anwendung soll gut dokumentiert sein und die Dokumentation soll die wichtigsten Aspekte berücksichtigen. Es soll ein Dokument zur Verfügung stehen, in dem der Aufbau der Anwendung zu sehen ist. Dazu eignet sich ein in [BLS11] beschriebenes Anwendungsmodell, dass alle Komponenten der Anwendung und ihre Implementierungen und Abhängigkeiten beinhaltet. Dieses Dokument verkürzt die Zeit für die Analyse der Applikation im Rahmen der ersten Phase der Migrationsmethodologie und ist auch ein Ausgangspunkt für das Redesign der Anwendungsarchitektur.

[1] The Principles of OOD: http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Skalierbarkeit

Die parallele Arbeit soll ermöglicht werden. Zu diesem Zweck sollen die Datenverarbeitungs- und Datenhaltungslogik voneinander getrennt werden. Die Datenverarbeitungsroutinen können beliebig skaliert werden und stören sich nicht gegenseitig.

Auf Web-geeignete Protokolle setzen

Es sollen ausschliesslich Netzwerkprotokolle eingesetzt, die im Web oder durch Firewalls gut funktionieren. Für synchrone Dienstschnittstellen eingnet sich am besten REST (oder SOAP). Für asynchrones Messaging, sollen Messaging-Protokolle eingesetzt werden. Hier bietet sich zum Beispiel MQTT[2] an. Für das Web-basierte Messaging eignen sich SignalR[3] oder WebSockets[4].

6.3.2 Migration

Aus den Erfahrungen und Erkenntnissen, die während der Migration von TFS ASAP gewonnen wurden, werden im Folgenden Empfehlungen für weitere, zukünftige Cloud-Migrationen von On-Premise-Anwendungen abgeleitet.

Mehr Zeit für die Analyse

Während der Migration wurden oft Entscheidungen aus der *Analyse und Planung* Phase geändert. Dies ist teilweise auf die Unreife und mangelhafte Dokumentation der gewählten oder verfügbaren Technologien zurückzuführen. Es deutet jedoch auch darauf hin, dass zu wenig Zeit für die *Analyse und Planung* und Proof-of-Concept Phasen investiert wurde. Bei künftigen Migration sollte für diese Aufgaben mehr Zeit in Anspruch genommen werden und die Entscheidungen sowie die Alternativen sollten mit der Begründung, mit Vor- und Nachteilen dokumentiert werden. Die Zeitabschätzung sollte nicht zu optimistisch sein.

Prozessverbesserungen

Die durchgeführten Tätigkeiten sollten täglich dokumentiert werden, damit im Nachhinein der Prozessfortschritt bewertet und analysiert werden kann. Alle Entscheidungsänderungen, die während der Umsetzungsphase vorgenommen werden, sind mit der Begründung zu dokumentieren und mit dem Projektverantwortlichen abzusprechen. Bei einer Designänderungen sollen Maßnahmen ergriffen werden, die es ermöglichen, den Zustand vor der Designänderung wiederherzustellen.
Um den Fortschritt besser zu kontrollieren und bei aufgetretenen Problem einzugreifen,

[2]MQTT: http://mqtt.org/
[3]ASP.NET SignalR: www.signalr.net
[4]The WebSocket API: http://dev.w3.org/html5/websockets/

empfiehlt es sich täglich ein *Daily Scrum* mit allen Projektbeteiligten durchzuführen. In seinen Rahmen sollt berichtet werden, was man seit dem letzten Mal erreicht hat, was man bis zum nächsten Mal zu erreichen plant und was hindert einen bei der Arbeit hindert.

Cloud Service

Cloud Service - Es wird keine Wartung des Betriebssystems erforderlich. Man kann automatische Skalierung vom Azure Management Portal aus einstellen. Durch die während des Deployments ausgeführten Tasks lassen sich Tätigkeiten zum individuellen Einstellen der Instanzen definieren. Somit kann zum Beispiel die IIS von der Web Role beeinflusst oder eine zusätzliche, erforderliche Software installiert werden. Das sowohl lokale als auch in der Cloud Debuggen sowie eine Remote-Desktop-Verbindung sind möglich. Diese Funktionalitäten sollten für den Test und Fehleridentifizierung eingesetzt werden. Cloud Service wird empfohlen, wenn eine *Cloudify*-Migration durchzuführen ist.

Anwendungsmodell

Falls nicht vorhanden, soll ein Anwendungsmodell der analysierten Anwendung wie im früheren Unterabschnitt beschrieben erstellt werden, das als Ausgangspunkt für die Analyse und das Redesign dient.

Skalierbarkeit

Bei der Analyse der Anwendung sollte man Rücksicht auf die Skalierbarkeit nehmen und Stellen im Quellcode erkennen, deren Ausführung sich parallelisieren lässt.

Zugriffsschnittstellen für Legacy-Komponenten sicherstellen

Es muss geprüft werden, welche Schnittstellen der Anwendung in der Cloud genutzt werden können. Schnittstellen die nicht genutzt werden können, sollte auf sie über Adapter-Komponenten zugegriffen werden können, wenn z.B. eine GUI oder nicht für das Web geeignete Protokolle angesprochen werden müssen. Am besten eignet sich REST/HTTP für diese Adapter-Komponenten.

Sicherheit

Da die Anwendungen im Web eingesetzt werden, muss unbedingt darauf geachtet werden, dass die übertragenen Daten nicht abgehört werden können, deswegen muss die Transportverschlüsslung (SSL) erfolgen. Bei Legacy-Anwendungen werden bisher mit hoher Wahrscheinlichkeit Protokolle eingesetzt, die für geschützte Firmennetzwerke ausgelegt sind.

7 Zusammenfassung und Ausblick

In diesem Kapitel wird zusammengefasst, was im Rahmen dieser Diplomarbeit gemacht wurde. Der aktuelle Stand der Arbeiten an TFS ASAP Online wird dargestellt. Im Bezug auf die entwickelte Cloud-Migrations-Methodologie sowie die Anwendung TFS ASAP Online wird auf mögliche weitere Aufgaben in diesem Kontext eingegangen.

Zuerst wurde die Ausgangssituation und Motivation dieser Diplomarbeit beschrieben. Dann wurden die Grundlagen in Bereichen Cloud Computing und Cloud Computing vorgestellt. Mit den Grundlagen erfolgte die Darstellung der Anwendung TFS ASAP Online der Firma AIT, die unter Einsatz der Microsoft Azure Plattform in die Cloud migriert werden sollte. TFS ASAP ist ein Plug-In für Team Foundation Server. Da Team Foundation eine Cloud-Version namens Visual Studio Online hat, sollte die Cloud-Version von TFS ASAP Visual Studio Online auf solche Weise erweitern, wie das TFS ASAP mit Team Foundations Server macht. Darüber hinaus wurde etwas Zeit in Anspruch genommen, um die Microsoft Azure Plattform und die gängigen Technologien aus der Microsoft-Welt näher zu bringen. Darauffolgend wurde der Kenntnisstand im Bezug auf die Cloud-Migration untersucht. Der Blickwinkel der Wissenschaft und der Industrie wurde in der Recherche berücksichtigt. Darüber hinaus wurden Methoden zur Evaluierung von Prozessen erforscht. Metriken für Prozesse wurden betrachtet. Basierend auf diesem Wissen wurde eine Methodologie für Cloud-Migration entwickelt, die für die Migration von TFS ASAP eingesetzt werden sollte. Die Methodologie entstand nach dem *best-of-breed*-Ansatz, wo die nach der objektiven Meinung des Autors besten Teile der Cloud-Methodologien von Microsoft und Amazon übernommen wurden. Zusätzlich wurden die für die Migration zu erhebenden Daten gewählt. Daraufhin wurde diese Methodologie für die Migration von TFS ASAP in die Microsoft Azure Plattform verwendet. Nach der Durchführung der Migration erfolgte die Analyse und Beurteilung des Prozesses unter Verwendung der gesammelten Daten. Am Ende wurden auf der Basis der gewonnenen Erfahrungen und Erkenntnisse Empfehlungen für weitere Cloud-Migrationen, sowie für die Entwicklung von Anwendungen, die zukünftig in die Cloud migriert werden könnten, abgeleitet.

Die Migration von TFS ASAP ist als erfolgreich einzustufen, obwohl nicht alle Anforderungen in Bezug auf die umzusetzende Funktionalität und den Test erfüllt wurden. TFS ASAP Online wurde in der Cloud von Microsoft ausgeliefert und kann heute von allen Leuten in der Welt in Anspruch genommen werden. Die Anforderungen für geforderte Grundfunktionalität wurden umgesetzt. Während der Migration wurden Kontakte mit Mitarbeitern von Microsoft verknüpft. Die Kontakte resultierten in der Eintragung von TFS ASAP Online in die Liste der offiziellen Erweiterungen von Visual Studio Online. Durch Verfassung von Blog-Artikeln und Teilnahme an Diskussionen an Foren für Visual Studio Online wurde ein Beitrag für die Community geleistet. TFS ASAP Online, anfangs ein Projekt, das im Rahmen einer

Diplomarbeit realisiert werden sollte, wurde inzwischen zu einem richtigen Produkt im Portfolio von AIT. Die während des Projekts gewonnene Erfahrung ist nicht zu unterschätzen. Während diese Diplomarbeit geschrieben wird, wurde die Entwicklung von TFS ASAP Online von anderen AIT-Mitarbeitern übernommen. Es wird immer noch an der Anwendung gearbeitet.

Leider konnten nicht alle geplanten Aufgaben realisiert werden. Was die technischen Aufgaben betrifft, wurden die Scheduled Automations nicht implementiert. Das gleiche gilt für die Load-Test, obwohl man eine Lösungskonzeption erarbeitete. Nur ein Durchlauf mit Hilfe der entwickelten Migrationsmethodologie wurde durchgeführt. Deshalb konnte diese Methodologie nicht richtig evaluiert werden, indem die während der Durchführung dieser Migration gesammelten Daten mit den während der Durchführung einer Migration unter Einsatz einer anderen Migrationsmethodologie gesammelten Daten verglichen wurden.

Für die künftigen Arbeiten, die im Bezug auf diese Arbeit geleistet werden können, bietet sich vor allem eine richtige Evaluierung der Methodologie an. Man sollte möglicherweise ein einfacheres Szenario auswählen und mehrere Migrationen mit Hilfe von mehr als einer Methodologie durchführen, damit man eine Vergleichsbasis hat. Im weiteren sollte die Liste der Empfehlungen für weitere Migrationen und für die Entwicklung von in der Zukunft zu migrierenden Anwendungen validiert werden. Vorhandene Empfehlungen sollten gegebenenfalls verbessert werden, neue Empfehlungen sollten hinzugefügt werden und die Empfehlungen, die sich als ungültig erweisen, sollten gelöscht werden. In der Cloud-Migration-Methodologie wurde keine Datenmigration berücksichtigt, da das in diesem Fall nicht notwendig war. Es muss nicht in alle Fällen so sein. Deswegen könnte die Migrationsmethodologie um ein Mögliche Datenmigration erweitert werden. Vor allem aus dem Grund, dass die Methodologien, auf den bei der Entwicklung der Methodologie aus dieser Diplomarbeit zurückgegriffen wurde, die Datenmigration als einen zusätzlichen Schritt im Rahmen einer Migration vorsehen. Darüber hinaus sollte die mögliche Werkzeugunterstützung bei einer Cloud-Migration detaillierter untersucht werden. Was die technische Seite betrifft - da sollten die Anforderungen, deren Implementierung storniert wurde, implementiert werden. Bei manchen ist die Entwicklung im Moment in vollem Gange, da die Ergebnisse dieser Diplomarbeit bei AIT intern Wiederverwendung fanden und ständig weiterentwickelt werden.

Literaturverzeichnis

[ABLS13] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch. How to Adapt Applications for the Cloud Environment. *Computing*, 95(6):493–535, Juni 2013.

[AFG+09] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[AITa] AIT GmbH & Co. KG. Automation: Aggregate Work Item State (CMMI). `https://tfsasaponline.aitag.com/FrontDesk/en-US/Home/Automations/0fed142f-faa4-4de2-9d96-e88bf397571f`.

[AITb] AIT GmbH & Co. KG. *Automation Service Developer Guide*.

[AITc] AIT GmbH & Co. KG. *TFS ASAP - Quick-Start: Installation, Setup and TFS ASAP*.

[AITd] AIT GmbH & Co. KG. *TFS ASAP - User Guide*.

[Bal98] H. Balzert. *Software-Management, Software-Qualitaetssicherung, Unternehmensmodellierung*. Spektrum Akad. Verl., Heidelberg, 1998.

[BBB+11] D. Baier, V. Bertocci, K. Brown, S. Densmore, E. Pace, and M. Woloski. *A Guide to Claims-based Identity and Access Control: Authentication and Authorization for Services and the Web*. Microsoft, 2011.

[BBG11] R. Buyya, J. Broberg, and A. Goscinski. *CLOUD COMPUTING: Principles and Paradigms*. John Wiley & Sons, Inc., 2011.

[BHJ+12] D. Betts, A. Homer, A. Jezierski, M. Narumoto, and H. Zhang. *Developing Multitenant Applications for the Cloud*. Microsoft, 2012.

[BLS11] T. Binz, F. Leymann, and D. Schumm. CMotion: A Framework for Migration of Applications into and between Clouds. In *Proceedings of the 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. IEEE Computer Society Conference Publishing Services, Dezember 2011.

[BWHK13] E. Blankenship, M. Woodward, G. Holliday, and B. Keller. *Professional Team Foundation Server 2012*. John Wiley & Sons, 2013.

[CS11] G. Case and G. Spalding. *ITIL Version 3 - Continual Service Improvement*. The Stationery Office, 2011.

[CTG+12] K. Cheng, S. Turkarslan, N. Garcia, S. Howard, S. Tinline-Jones, S. Pelluru, S. Coriani, and J. A. Bravo. *Migrating Data-Centric Applications to Windows Azure*. MSDN Library, 2012.

[Day14] B. Day. Walkthrough: ASP.NET MVC Identity with Microsoft Account Authentication, Februar 2014. `http://goo.gl/7Bk81M`.

[DHN+12] S. Densmore, A. Homer, M. Narumoto, J. Sharp, and H. Zhang. *Building Hybrid Applications in the Cloud on Windows Azure*. Microsoft, 2012.

[FRF02] M. Fowler, D. Rice, and M. Foemmel. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, November 2002.

[Gal04] D. Gallin. *Software Quality Assurance: From Theory to Implementation*. Pearson/AddisonWesley, 2004.

[Gec06] B. Geck. Übersicht der Prozessmodelle CMMI, SPICE, ITIL. gebeCom GmbH., 2006.

[Hin14] M. Hinshelwood. Getting a Service Account for VSO with TFS Service Credential Viewer, 2014. `http://nakedalm.com/getting-service-account-vso-tfs-service-credential-viewer/`.

[HSB+14] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson. *Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications*. Microsoft, 2014.

[JAP13] P. Jamshidi, A. Ahmad, and C. Pahl. Cloud Migration Research: A Systematic Review. *Cloud Computing, IEEE Transactions on*, 1(2):142–157, July 2013.

[Kan04] S. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, 2004.

[LL07] J. Ludewig and H. Lichter. *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.Verlag GmbH, 2007.

[LR00] F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, 2000.

[Mica] Microsoft. Azure Execution Models. `http://azure.microsoft.com/en-us/documentation/articles/fundamentals-application-models/`.

[Micb] Microsoft. Azure Service Bus. `http://azure.microsoft.com/en-us/documentation/articles/fundamentals-service-bus-hybrid-solutions/`.

[Micc] Microsoft. Azure Websites, Cloud Services, and Virtual Machines comparison. `http://azure.microsoft.com/en-us/documentation/articles/choose-web-site-cloud-service-vm/`.

[Micd] Microsoft. Enabling Diagnostics in Azure Cloud Services and Virtual Machines. `http://azure.microsoft.com/en-us/documentation/articles/cloud-services-dotnet-diagnostics/`.

[Mice] Microsoft. How to use Blob Storage from .NET. `http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs/`.

[Micf] Microsoft. How to Use Service Bus Queues. http://azure.microsoft.com/en-us/documentation/articles/service-bus-dotnet-how-to-use-queues/.

[Micg] Microsoft. How to Use Service Bus Topics/Subscriptions. http://azure.microsoft.com/en-us/documentation/articles/service-bus-dotnet-how-to-use-topics-subscriptions/.

[Mich] Microsoft. How to use Table Storage from .NET. http://azure.microsoft.com/en-us/documentation/articles/storage-dotnet-how-to-use-tables/.

[Mici] Microsoft. Introducing Microsoft Azure. http://azure.microsoft.com/en-us/documentation/articles/fundamentals-introduction-to-azure/.

[Micj] Microsoft. Manage Accounts, Subscriptions, and Administrative Roles. http://msdn.microsoft.com/en-us/library/azure/hh531793.aspx.

[Mick] Microsoft. .NET On-Premises/Cloud Hybrid Application Using Service Bus Relay. http://azure.microsoft.com/en-us/documentation/articles/cloud-services-dotnet-hybrid-app-using-service-bus-relay/.

[Micl] Microsoft. Virtual Machines. http://azure.microsoft.com/en-us/services/virtual-machines/.

[Micm] Microsoft. What is a Cloud Service? http://azure.microsoft.com/en-us/documentation/articles/cloud-services-what-is/.

[Mic14] Microsoft. Authorize Access with OAuth 2.0, 2014. http://www.visualstudio.com/en-us/integrate/get-started/get-started-auth-oauth2-vsi.

[MS14] S. Manheim and R. Squillace. *Windows Azure Service Bus Reference*. Microsoft, 2014.

[NIS11] National Institute of Standards and Technology. The NIST Definition of Cloud Computing, 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[Par57] C. N. Parkinson. *Parkinson's Law, and Other Studies in Administration*. Boston, Houghton Mifflin, 1957.

[Pel14] J. Pelser. Using Microsoft Live authentication in ASP.NET MVC, 2014. http://www.beabigrockstar.com/guides/aspnet-oauth/aspnet-mvc-microsoft-login.

[PXW13] C. Pahl, H. Xiong, and R. Walshe. A Comparison of On-Premise to Cloud Migration Approaches. pages 212–226, 2013.

[Rü14] T. Rümmler. TFS ASAP is an official Visual Studio Online Integration, Dezember 2014. http://blog.aitgmbh.de/2014/12/12/tfs-asap-is-an-official-visual-studio-online-integration/.

[Sab14a] J. Sabacinski. Authentifizierungsmöglichkeiten bei der Visual Studio Online REST API, 2014. http://blog.aitgmbh.de/2014/08/06/authentifizierungsmglichkeiten-bei-der-visual-studio-online-rest-api/.

[Sab14b] J. Sabacinski. How to Change Work Item State Using Visual Studio Online REST API, 2014. http://blog.aitgmbh.de/2014/08/26/how-to-change-work-item-state-using-visual-studio-online-rest-api/.

[Sab14c] J. Sabacinski. Spezifikation TFS ASAP Online User Interface. AIT Gmbh & Co. KG, 2014.

[Sab14d] J. Sabacinski. TFS ASAP Online - Anforderungsspezifikation. AIT GmbH & Co. KG, Juli 2014.

[SABL13] S. Strauch, V. Andrikopoulos, T. Bachmann, and F. 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 36–46. SciTePress, Mai 2013.

[SALM12] S. Strauch, V. Andrikopoulos, F. Leymann, and D. Muhler. ESB^{MT}: Enabling Multi-Tenancy in Enterprise Service Buses. In *Proceedings of the 4th IEEE International Conference on Cloud Computing Technology and Science (CloudCom'12)*, pages 456–463. IEEE Computer Society Press, Dezember 2012.

[SKLU11] S. Strauch, O. Kopp, F. Leymann, and T. 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.

[SS14] H. Schwichtenberg and M. Steyer. *Moderne Webanwendungen mit ASP.NET MVC und JavaScript: ASP.NET MVC im Zusammenspiel mit Web APIs und JavaScript-Frameworks*. O'Reilly, 2014.

[Sto02] R. Stockmann. Was ist eine gute Evaluation. Centrum für Evaluation, 2002.

[Sto11] A. Stone. How to Write a Robust TFS Server Plugin, with Job Extensions, 2011. http://blogs.microsoft.co.il/assafstone/2011/07/29/how-to-write-a-robust-tfs-server-plugin-with-job-extensions/.

[TUS11] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam. To Move or Not to Move: The Economics of Cloud Computing. *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, pages 5–5, 2011.

[VA12] Q. H. Vu and R. Asal. Legacy Application Migration to the Cloud: Practicability and Methodology. In *Services (SERVICES), 2012 IEEE Eighth World Congress on*, pages 270–277. IEEE, 2012.

[Var10] J. Varia. Migrating Your Existing Applications to the AWS Cloud: A Phase-driven Approach to Cloud Migration. Amazon Web Services, October 2010.

[WAB+09] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, and J. Stößer. Cloud-Computing. *WIRTSCHAFTSINFORMATIK*, 51(5):453–462, 2009.

[Weh14a] B. Wehrle. TFS ASAP meets VisualStudio.com, Oktober 2014. `http://blog.aitgmbh.de/2014/10/10/tfs-asap-meets-visualstudio-com/`.

[Weh14b] B. Wehrle. TFS ASAP Online Preview goes public, November 2014. `http://blog.aitgmbh.de/2014/11/02/tfs-asap-online-preview-goes-public/`.

Alle Verweise wurden zuletzt am 7. Januar 2015 abgerufen.

Danksagung

Ich möchte mich an dieser Stelle bei meinem Betreuer aus der Wissenschaft Steve Strauch herzlich für seine Unterstützung bedanken. Ein Dank gilt auch an meinen Betreuer aus der Industrie Boris Wehrle.

Jakub Sabaciński

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, 9. Januar 2015 _____

(Jakub Sabaciński)