DIPLOMARBEIT

Einführung von Enterprise Architecture Management in einem Technologieunternehmen: Eine Fallstudie

ausgeführt zur Erlangung des akademischen Grades
eines Diplom-Ingenieurs unter der Leitung von

Univ.Prof. Dipl.-Ing. Dr. Hermann Kaindl

am

Institut für Computertechnik (E384)
der Technischen Universität Wien

durch

Robert Miksch BSc
Matr.Nr. 0625824
Schadinagasse 10/17, 1170 Wien, Österreich

Wien, am 01. 10. 2014 _____

Kurzfassung

Es wird ein Unternehmen dabei unterstützt, eine zentrale und strukturierte Dokumentation seiner IT-Landschaft zu erstellen. Es werden die Anforderungen der Stakeholder aufgenommen und verschiedene Enterprise Architecture Frameworks betrachtet, die zur Umsetzung geeignet sind. Aufgrund des Fokuses der Anforderungen auf die Applikations-Ebene (im Gegensatz zu einem Fokus auf Infrastruktur-, Daten- oder Geschäfts-Ebene) wird das Framework ArchiMate verwendet. Mit Hilfe des Open-Source-Software-Tools Archi wird ein Modell der IT-Landschaft erstellt. Zum Abschluss werden weitere Schritte für Pflege und Ausbau des Modells diskutiert.

Abstract

A company is supported in creating a central and structured documentation of its IT landscape. The demands of the stakeholders are gathered and several Enterprise Architecture Frameworks are examined. Since the demands are mainly focused on the application layer (instead of on the infrastructure, data or business layer) the framework ArchiMate is used. A model of the IT landscape is created using the open source software Archi. Finally further steps for maintening and expanding the model are discussed.

Danksagung

Zu allererst möchte ich meinem Betreuer Univ.Prof. Dipl.-Ing. Dr. Hermann Kaindl für die Möglichkeit danken meine Diplomarbeit in seinem Fachgebiet zu schreiben. Ich danke ihm für seine produktiven Rückmeldungen und seine fachkundigen Hilfestellungen.
Ein besonderer Dank gebührt dem Partnerunternehmen dieser Arbeit. Insbesondere erwähnt sei Dr. Wolfgang H., dessen Engagement wesentlich dazu beigetragen hat, dass diese Kooperation zustande gekommen ist.
Dr. Selim Erol danke ich für die tatkräftige Unterstützung zum Beginn der Arbeit. Meiner Freundin Andrea danke ich für das Korrekturlesen meiner Arbeit und die stetige emotionale Unterstützung. Ein großes Dankeschön gilt auch meinen Eltern, die mich während meines gesamten Studiums unterstützt haben und stets hinter mir gestanden sind.

Inhaltsverzeichnis

Abkürzungen

EA	Enterprise Architecture, Unternehmensarchitektur
EAM	Enterprise Architecture Management
IT	Informationstechnologie
BPMN	Business Process Modelling Notation
ITIL	IT Infrastructure Library
UML	Unified Modeling Language
DMS	Dokumenten-Management-System

1 Einleitung

Diese Diplomarbeit ist in Kooperation mit einem Partnerunternehmen entstanden. Das Unternehmen entwickelt und vertreibt ein Portfolio technischer Produkte im Business-To-Business-Markt. Die IT-Abteilung des Unternehmens kam auf den Autor dieser Diplomarbeit zu, um ein Überblicksbild ihrer IT-Landschaft zu erarbeiten.

Die IT-Abteilung eines Unternehmens hat die Aufgabe, Systeme bereitzustellen, die Funktionalitäten und Daten liefern, um die Firma beim Umsetzen ihrer Tätigkeiten und Ziele bestmöglich zu unterstützen. Bei dem Partnerunternehmen ist die IT-Abteilung gemeinsam mit dem Unternehmen historisch mitgewachsen. Dadurch ist die IT-System-Landschaft im Laufe der Zeit bedeutend komplexer, vielschichtiger und undurchsichtiger geworden. Die Methoden, um die Systeme zu beherrschen, wurden allerdings nicht in gleichem Maße angepasst.

Viele neue Systeme sind „bei Bedarf“ in das bestehende Geflecht integriert worden, was zu einem verwobenen und stellenweise mangelnd dokumentierten Gesamtsystem geführt hat. Vor allem fehlt es an einem aktuellen, allumfassenden Gesamtbild. Das ist keine ungewöhnliche Situation, sondern ein Resultat aus schnellem Wachstum und hohem Umsetzungsdruck.

Alte System können in so einer Situation oftmals nicht ohne bedeutendem Mehraufwand vollständig ausgegliedert werden, weil die Auswirkung davon auf die anderen Systeme nicht ohne eingehende Untersuchungen bekannt ist. Dies erhöht sowohl Risiko als auch Aufwand eines eventuellen Konsolidierungsprojekts, was wiederum die Wahrscheinlichkeit einer Verschiebung so eines Projekts steigert. Das resultiert letzten Endes in erhöhten Lizenz- und Wartungskosten. Umgekehrt ist es ebenso aufwendig, bestehende Systeme abzuändern oder neue Systeme zu integrieren, wenn nicht restlos klar ist, welche Auswirkungen auf die Gesamt-Landschaft dabei zu berücksichtigen sind.

Um kostengünstig zu bleiben, muss die IT-Strategie den Bedürfnissen des Unternehmens entsprechen. Das sicher zu stellen ist Aufgabe des Managements. Ein gemeinsames Bild der IT-Landschaft und davon, wie die einzelnen Unternehmensprozesse diese nutzen, wäre dabei eine große Hilfe. Wenn das IT-Management und das Management der Fachbereiche so ein gemeinsames Bild zur Verfügung hätten, würde das dazu beitragen Missverständnissen vorzubeugen und Meetings zu verkürzen.

Einer IT-Abteilung muss darüberhinaus stets klar sein, von welchen KollegInnen ihre Systeme benutzt werden, da diese in Systemänderungsprozesse miteinbezogen und über eventuelle Ausfälle informiert werden müssen. Die Nutzerkreise ändern sich jedoch mit jeder Änderung der Geschäftsabläufe. Dieses Wissen gehört ebenfalls festgehalten, da es sonst schwierig und aufwendig ist, stets am Laufenden zu bleiben.

Bisherige Dokumentationsversuche

In dem Unternehmen kam also immer mehr der Wunsch auf, sämtliche Abhängigkeiten der IT-Systeme zentral zu erfassen und darzustellen. Es gab bereits in der Vergangenheit Versuche eine derartige Dokumentation zu erstellen. Keiner war jedoch von nachhaltigem Erfolg geprägt.

Das größte derartige Dokumentationsprojekt des Unternehmens resultierte in einer gigantischen Microsoft Visio-Datei mit Zeichnungen von Systemen und System-Verbindungen. Die Datei war allerdings so unübersichtlich, dass nur in einem Ausdruck im DIN-A0-Format die Beschriftungen noch lesbar waren. Außerdem wurden die verwendeten Symbole nicht beschrieben, sondern einfach als intuitiv angenommen. Es gab keine Vorgabe an das Management oder die TechnikerInnen, das Dokument zu pflegen und es gab keine Verantwortlichen, die die Pflege vorantrieben. Die AkteurInnen, die daraus Nutzen hätten ziehen können, empfanden es als zu unübersichtlich und zu komplex. So wurde es nicht mehr weiter gepflegt und das Dokument veraltete und wurde wertlos.

Die Nutzerkreise der jeweiligen Systeme waren in dem Dokument überhaupt nicht enthalten. Stattdessen gab und gibt es in dem Unternehmen eine weitere, ISO9001-konforme Dokumentation der Geschäftsprozesse, in der auch die im Zuge dieser Prozesse genutzten technischen Systeme festgehalten sind. Diese beiden Dokumentations-Arten waren völlig voneinander isoliert.

Eine angestrebte neue Dokumentation soll aus den gemachten Fehlern lernen und sie vermeiden. Daher wurden folgende Maßnahmen für einen neuen Dokumentationsanlauf festgelegt:

1. *Komplexität für die Akteure so gering wie möglich halten.*
   Wenn die Dokumentation komplex oder die Dokumentationstätigkeit unnötig zeitaufwendig ist, werden die AkteurInnen sie nur ungern pflegen oder zu Rate ziehen. Man sollte sich nicht durch die Gesamtheit der Systeme arbeiten müssen, wenn gerade nur ein Teil-Ausschnitt relevant ist. Die verwendete Symbolik sollte dokumentiert oder wenn möglich sogar standardisiert sein, um den Wiedererkennungswert zu erhöhen und die Grafik dadurch zu vereinfachen.

2. *Die AkteurInnen sollen motiviert werden, die Dokumentation zu nutzen und aktuell zu halten.*
   Die Systeme werden laufend verändert und jede Änderung muss dokumentiert werden. Nur eine Dokumentation, auf deren Korrektheit man sich verlassen kann, ist wertvoll. Daher müssen alle Personen, die die technischen Systeme oder die Geschäftsprozesse ändern können, die Verantwortung tragen, dass jede ihrer Änderungen zentral dokumentiert wird.

3. *Dokumentation von Technik und Geschäftsprozessen soll vereint sein.*
   Dieser Punkt ist wichtig, weil sonst die Daten doppelt gepflegt werden und die Aktualität der Daten unterschiedlich sein kann. Wenn es keine direkte Verknüpung von einem bestimmten System in der technischen Dokumentation mit dem gleichen System in der geschäftsprozessorientierten Dokumentation gibt, muss diese Verknüpfung beim Nachschlagen manuell hergestellt werden, was zeitaufwendig und fehleranfällig ist.

Neuer Dokumentationsversuch

In der Literatur (z.B. [Han09], [Lan09]) wird zu diesem Zweck die Einführung einer Unternehmensarchitektur (Enterprise Architecture, EA) empfohlen:

> „Durch eine Unternehmensarchitektur (Enterprise Architecture) wird eine Gesamtsicht auf das Unternehmen geschaffen. Eine Unternehmensarchitektur beinhaltet alle wesentlichen Business- und IT-Strukturen und deren Verknüpfungen. Auf dieser Basis lassen sich das Business und die IT und deren Zusammenhänge beschreiben. Abhängigkeiten und Auswirkungen von Veränderungen in Business und IT werden transparent.“ [Han09, S. 57f]

Eine EA vereint wie gefordert die Dokumentation der Technik- und Geschäftsebenen. Mit ihren standardisierten Darstellungen und Informationen bietet sie eine gemeinsame Sprache für IT und Fachbereich und sie ermöglicht ein einfaches Einlernen für AkteurInnen, die das Modell pflegen oder nutzen.

Im Zuge dieser Diplomarbeit wird bei dem Partnerunternehmen Enterprise Architecture Management, kurz EAM, eingeführt. In Kapitel 2 wird EAM theoretisch in ihren Grundzügen aufbereitet. Ihr Nutzen wird hervorgehoben und ihr Wesen wird von verwandten Sprachen und Normen wie Business Process Model and Notation (BPMN) [6], IT Infrastructure Library (ITIL) und der Qualitätsmanagement-Norm ISO 9001 abgegrenzt. Außerdem werden verschiedene EAM-Frameworks vorgestellt. Das sind Ansammlungen an Anleitungen zur Realisierung eines EAM.

In Kapitel 3 werden die Anforderungen der verschiedenen Stakeholder an das zu entwerfende EAM aufgenommen, und das Schema des Modells entwickelt (Kapitel 3.1.3). Im Anschluss wird ein EAM-Framework und ein dazu gehöriges Software-Tool ausgewählt. Das Schema wird dann entsprechend auf das Framework angepasst und in gemeinsamer Arbeit mit den Stakeholdern werden die Daten schließlich erhoben und das Modell realisiert.

Zum Abschluss werden in Kapitel 3.4 die gesammelten Erfahrungen und Erkenntnisse zusammengefasst und nächste Schritte für das Partnerunternehmen empfohlen, um das EAM sinnvoll weiterzuentwickeln und auszubauen.

Das Resultat dieser Arbeit soll dem Unternehmen folgende Mehrwerte generieren:

- Ein standardisiertes Modell seiner IT-Landschaft, der Geschäftsprozesse und deren Interaktionen,
- eine Dokumentation zur Nutzung des Modells für Personen, die es pflegen oder nutzen wollen und
- ein Maßnahmenkatalog, um das Modell zu integrieren und um es aktuell und nutzbar zu halten.

2 Enterprise Architecture Management

Eine Unternehmensarchitektur oder Enterprise Architecture (EA) hat den Zweck, wesentliche Bestandteile des Geschäfts und der IT festzuhalten. Sie liefert ein gesamtheitliches Bild des Unternehmens, weil sie Informationen aus unterschiedlichen Domänen zusammenbringt und verknüpft [Lan09, S. 3].

Eine optimiertes, performantes und kostengünstiges technisches System kann sich beispielsweise als zu unflexibel dafür erweisen, einen agilen Geschäftsprozess zu unterstützen. Ein Tunnelblick rein auf die technische Ebene sieht ein sehr gut funktionierendes System. Nur die gemeinsame Betrachtung der Technologie- und der Geschäftsebene offenbart das Problem, und das ist die Stärke von Enterprise Architecture Management.

2.1 EAM Grundlagen

> „Enterprise Architecture Management stellt Hilfsmittel bereit, um die Komplexität der IT-Landschaft zu beherrschen und die IT-Landschaft strategisch und Business-orientiert weiterzuentwickeln.“ [Han12, S. 12]

Enterprise Architecture Management (EAM) schafft Transparenz in der IT-Landschaft durch Darstellung, wie die IT-Systeme untereinander zusammenarbeiten und wie sie von Geschäftsprozessen benutzt werden. Dies passiert über Tabellen, Grafiken oder Diagramme.

Einer der Mehrwerte, der dadurch entsteht, ist, dass es eine gemeinsame Basis für TechnikerInnen und Nicht-TechnikerInnen schafft. EAM kann dabei helfen, die Sprachbarrieren zu überwinden und Missverständnisse zu vermeiden, indem es ein gemeinsames Bild bereitstellt. Da sowohl IT als auch Business abgebildet werden, finden alle Beteiligten ihren Bereich darin wieder. Dadurch hat man eine bedeutend motivierendere Situation, als wenn beispielsweise Business-Vertreter mit einem rein technischen Abbild konfrontiert werden.

Der Überblick, den EAM über die IT-Landschaft bringt, liefert dem Unternehmen Möglichkeiten, seine Systeme zu konsolidieren, strategisch weiterzuentwickeln und Business-orientiert auszurichten. Wie detailliert ein EAM ist, hängt allein von den Anforderungen des Unternehmens ab. Es gibt nicht den einen korrekten Abstrahierungsgrad. Die Detailtiefe, in der modelliert wird, ist so zu wählen, dass die Anforderungen gerade erfüllt werden. Alles was darüber hinausgeht, ist zu vermeiden, da es die Komplexität des Modells erhöht ohne einen Mehrwert zu generieren.

Obwohl es unterschiedliche Umsetzungen gibt, unterscheiden die meisten EA-Implementierungen vier Ebenen [SMS+09, Kapitel 2.2.1.1]:

1. Die *Geschäfts-Architektur* umfasst fachliche, betriebswirtschaftliche Aktivitäten.
2. Die *Daten-Architektur* umfasst Geschäftsobjekte, Informationen und Daten, welche im Unternehmen entstehen.
3. Die *Applikations-Architektur* umfasst Softwarelösungen, die im Unternehmen in Verwendung sind.
4. Die *Infrastruktur-Architektur* umfasst die IT-Infrastruktur, auf der die Softwarelösungen betrieben werden.

Die Stärke von EA ist es, diese vier unterschiedlichen Architekturdomänen aussagekräftig miteinander zu verknüpfen. Man erhält dadurch ein Gesamt-Modell, das mehr ist als lediglich die Summe seiner vier Teile.

Im Folgenden wird EAM von anderen IT-Management-Disziplinen abgegrenzt, und konkrete Implementierungen werden betrachtet.

2.2 Abgrenzung zu anderen Beschreibungssprachen und Disziplinen des IT-Managements

Es gibt unterschiedlichste Ansätze des IT-Managements mit unterschiedlichen Zielen. EAM mit anderen Management-Methoden zu vergleichen hilft dabei, es besser einzuordnen und zu verstehen. Es folgt eine Abgrenzung von EAM zu drei verbreiteten Methoden.

Business Process Model and Notation (BPMN)

BPMN wurde ursprünglich von der Business Process Modelling Initiative und später von der Object Management Group entwickelt und standardisiert. Es liefert eine graphische Darstellungsmöglichkeit von Geschäftsprozess- und Arbeitsablaufs-Modellierungen. Seit Version 2.0 beinhaltet BPMN außerdem eine Semantik zum Ausführen von Modellen. Während der Fokus von EAM auf Architektur liegt, fokusiert sich BPMN auf die Geschäftsprozesse. Applikationen und technische Infrastruktur werden nicht abgedeckt [Lan09, Kapitel 2.3.2]. EAM hingegen erlaubt sowohl Modellierungen dieser Domänen als auch von Beziehungen zwischen ihnen.

Für die Problemstellung dieser Arbeit reicht BPMN daher nicht aus, weil es nicht dafür entworfen ist eine IT-Landschaft abzubilden. Es gibt Überschneidungen von BPMN mit EAM auf der Geschäfts-Ebene. Eine detaillierte Analyse dieser Überschneidung zwischen BPMN und der EAM-Sprache ArchiMate liefert beispielsweise die Masterarbeit [Usm12].

Information Technology Infrastructure Library (ITIL)

ITIL ist ein weltweit anerkannter Standard für IT-Service-Management. Ursprünglich wurde ITIL vom britischen Office of Government Commerce entwickelt. Es liefert in einer Serie von Dokumenten Anweisungen, wie man IT-Services am besten bereitstellt [Lan09, Kapitel2.1.5].

> „Schon die Definition von ITIL zeigt, dass ITIL nicht unbedingt als Kochbuch entworfen worden ist, um damit eine komplette IT-Funktion zu organisieren, sondern wie der Name sagt, es geht um das Thema ‚Infrastruktur', also um das, was auch als ‚Systembetrieb' bezeichnet wird." [Kel07, S. 69f]

Keller argumentiert also, dass ITIL Anweisungen für den Betrieb einer IT-Landschaft bietet, nicht aber für die Architektur davon. ITIL sollte laut Keller zwar als Nachschlagewerk verwendet werden, ist jedoch nicht hilfreich für ein Prozess-Modell der IT-Unternehmensarchitektur. Mehr Informationen zum Unterschied und einer möglichen Zusammenarbeit zwischen EAM und ITIL ist beispielsweise der Masterarbeit [Rut11] zu entnehmen.

Das Partnerunternehmen bietet seine IT-Services nach dem ITIL-Standard an.

Qualitätsmanagement-Norm ISO 9001

Der ISO-9001-Standard der International Organisation for Standardisation (ISO) gibt die Kriterien für ein gutes Qualitätsmanagement-System (QMS) vor. Der Standard definiert die Aufgaben des QMS und gibt Anforderungen an Personal, Lehrgänge und Arbeitsumgebung vor.

Er schreibt dem Unternehmen vor Schlüssel-Prozesse zu ermitteln und zu dokumentieren. Das sind sämtliche Prozesse, die eine wichtige Rolle in der Wertschöpfungskette haben, und somit für ein QMS von zentraler Bedeutung sind.

Um das entsprechende Zertifikat zu bekommen, muss sich ein Unternehmen einem externen Audit unterwerfen, welches die Einhaltung der Norm sicherstellt. Eine gut entworfene und dokumentierte Unternehmensarchitektur kann einem Unternehmen dabei helfen, die Anforderungen von ISO 9001 umzusetzen. QMS und EAM ergänzen einander sehr gut: QMS gibt vor, was entworfen, dokumentiert, kontrolliert, gemessen und verbessert werden soll, und EAM definiert, wie die Prozesse und Resourcen organisiert und realisiert werden [Lan09, Kapitel 2.1.3].

Das Partnerunternehmen ist seit 1993 in seinem zentralen Standort in Wien ISO 9001 zertifiziert. Die meisten Nebenstandorte sind später ebenfalls zertifiziert worden [1].

2.3 Übersicht über gängige Enterprise Architecture Frameworks

Ein EA-Framework ist eine Sammlung von Thesen, Konzepten, Werten und Praktiken, welche gemeinsam eine Sicht auf die Realität eines Unternehmens durch Betrachten eines Modells ermöglichen [BBL12, S. 106]. Es bietet eine Struktur, wie man EA entwickelt, wartet und nutzt.

Es existieren zur Zeit über 50 EA-Frameworks [Mat11] mit unterschiedlichem Fokus, Reifegrad und Akzeptanz in der Unternehmenswelt. Viele Unternehmen setzen auch ganz auf Eigenentwicklungen oder sie wandeln ein Standard-Framework nach ihren individuellen Bedürfnissen ab.

Das Partnerunternehmen hat keine Expertise auf dem Gebiet und es hat bereits schlechte Erfahrungen damit gemacht einen eigenen Ansatz anzuwenden (siehe Kapitel 1). Eine Eigenentwicklung wird daher ausgeschlossen, und es soll stattdessen ein etablierter Standard ausgewählt werden.

Es gibt bereits einige Arbeiten, welche die Relevanz der einzelnen Frameworks untersuchen und die Vielfalt somit bedeutend einschränken. Im Folgenden werden die Ergebnisse dieser Arbeiten zusammengefasst, um die etabliertesten Methoden zu identifizieren.

2.3.1 Analyse der Verbreitung verschiedener Enterprise Architecture Frameworks

Laut einer Umfrage [LM11] aus 2011 unter 16 ExpertInnen aus unterschiedlichen Branchen und unterschiedlicher Unternehmensgrößen wurden individuell adaptierte Versionen der Frameworks TOGAF (66,7%), ARIS(25%) und Zachman (8,3%) verwendet. Die restlichen 25% verwendeten kein bestehendes Framework, sondern einen eigenen Ansatz. Es wurde also in keinem der 16 Fälle ein Framework in seiner Originalform verwendet.

Eine weitere Umfrage [Ins05] aus dem Jahr 2005 unter 79 Unternehmen aus unterschiedlichen Branchen und Ländern zeigt folgende Verteilung: Zachman (25%), Eigenentwicklungen (22%), TOGAF (11%), DoDAF (11%), FEAF (9%), E2AF (9%), andere (13%).

Lankhorst bezeichnet in seinem Buch [Lan09, Kapitel 2.2] unter anderem folgende EA Frameworks als „renommiert“ („well-known“): IEEE 1471-2000/ISO/IEC 42010 Standard, Zachman, TOGAF, DoDAF/C^{4}ISR, RM-ODP, GERAM, und Nolan Norton Framework.

In „Collaborative Enterprise Architecture“ [BBL12, S. 107] aus dem Jahr 2012 werden die folgenden drei Frameworks hervorgehoben:

- Das Zachman-Framework war der erste Vorstoß in Richtung EA und ist heute noch in Verwendung.
- TOGAF ist als offenes Framework Industrie- und Hersteller-unabhängig und wird von einer Community vorangetrieben.
- Gartner Methodology (oder META Framework) stammt von einer in der Firmenwelt renommierten Beratungsfirma.

Dabei wird explizit darauf hingewiesen, dass diese drei nicht unbedingt die besten Frameworks seien, sondern die am weitesten verbreiteten.

Aus diesen Daten stechen das TOGAF- und das Zachman-Framework besonders hervor, welche laut den Umfragen weit verbreitet sind und auch in den theoretischen Arbeiten hervorgehoben werden. Zusätzlich werden ARIS und DoDAF laut den Umfragen ebenfalls häufig genutzt. Diese vier Frameworks werden daher im Folgenden kurz vorgestellt.

Eine bedeutend umfassendere Betrachtung der aktuell am Markt verfügbaren Frameworks ist [Mat11] zu entnehmen.

2.3.2 Das Zachman Framework

1987 veröffentlichte John A. Zachman im IBM System Journal den Artikel „A framework for information systems architecture“ [Zac87], in dem erstmals ein EA Framework konzipiert wurde. Gemeinsam mit John Sowa entwickelte er es weiter, bis es 1992 in einer zweiten Version [SZ92] veröffentlicht wurde.

Tabelle 2.1 zeigt die typische Darstellung des Zachmann-Frameworks in Matrixform. Die Spalten repräsentieren einzelne *Modelle*, welche Aussagen treffen, die eine Antwort auf die jeweiligen Fragen in Klammern geben. Jede Zeile steht für eine *Perspektive*. In Klammern steht die der Perspektive entsprechende Rolle, zum Beispiel die des Designers, welcher der Perspektive Systemmodell zugeordnet ist.

Jede Zelle ist daher einzigartig, aber keine ist bedeutender als eine andere. Sie liefern Modelle des Unternehmens aus verschiedenen Perspektiven und mit unterschiedlichen Fragestellungen.

	Daten (Was?)	Funktion (Wie?)	Netzwerk (Wo?)	Menschen (Wer?)	Zeit (Wann?)	Motivation (Warum?)
Bereich (Planer)	Liste geschäftsrelevanter Faktoren	Liste Geschäftsprozesse	Liste Unternehmens-Standorte	Liste Geschäftseinheiten	List geschäftsrelevanter Ereignisse	Liste Geschäftsstrategien und -ziele
Unternehmensmodell (Besitzer)	Entity-Relationship-Diagramm	Prozess-Fluss-Diagramm	Logistik-Netzwerk	Organisations-Chart	Zeitablaufplan	Geschäftsplan
Systemmodell (Designer)	Datenmodell	Datenfluss-Diagramm	Architektur verteilter Systeme	Architektur Human Interface	Prozess-Struktur	Wissens-Architektur
Technologiemodell (Erbauer)	Daten-Design	Struktur-Chart	System-Architektur	Mensch/Technik Interface	Kontroll-Strukturen	Wissens-Design
Komponenten (Subunternehmen)	Daten-Definition	Software-Programm	Netzwerk-Architektur	Security-Architektur	Timing-Definitionen	Wissens-Definitionen

Tabelle 2.1: Matrix-förmige Repräsentation des Zachman Frameworks. Quelle: [SZ92, S. 600f]

Lankhorst lobt die Einfachheit des Frameworks, die ganzheitliche Sicht auf das Unternehmen und die Tool-Unabhängigkeit [Lan09, S. 23]. Als Schwäche des Frameworks nennt er die große Anzahl der Zellen, was die praktische Anwendbarkeit erschwert. Außerdem sind laut Lankhorst die Beziehungen der Zellen miteinander nur mangelhaft spezifiziert.

2.3.3 The Open Group Architecture Framework (TOGAF)

TOGAF wird von *The Open Group Architecture Forum* mit ihren über 300 Mitgliedsfirmen entwickelt und gewartet [Har11] und hat aktuell Version 9.1. Die Dokumentation [9] ist frei verfügbar für Lehre, Forschung und für den Eigengebrauch, für kommerzielle Nutzung ist hingegen eine Lizenzierung notwendig.

Die offene, Community-gesteuerte, herstellerunabhängige und werkzeugunabhängige Natur von TOGAF ist seine große Stärke, genauso wie die weltweit über 20.000 TOGAF-zertifizierten ArchitektInnen, die einen großen Markt an möglichen BeraterInnen bilden.

Die Hauptkomponenten von TOGAF sind [Lan09, Kapitel 2.2.4]

- das *Architecture Capability Framework*, welche die zur EA-Einführung notwendige Organisation, Prozesse, Fähigkeiten, Rollen und Verwantwortlichkeiten innerhalb des Unternehmens definiert,

- die *Architecture Development Method ADM*, welche die Arbeitsschritte des Architekten definiert (siehe Abbildung 2.1),
- das *Architecture Content Framework*, welches eine Architektur über das gesamte Unternehmen, also alle vier Domänen (siehe Seite 5) beschreibt,
- und das *Enterprise Continuum*, welches Referenzmodelle anbietet.

Das Herzstück von TOGAF, das ADM, ist sowohl zwischen den Phasen als auch innerhalb davon iterativ aufgebaut. Zwischen zwei Iterationen werden die Vorgaben angepasst, wie beispielsweise die gewünschte Breite und Tiefe des Modells. Dadurch wird das Modell zyklisch an die aktuellen Anforderungen angepasst. Das ist eine Arbeitsweise, die gut in ein Umfeld passt, in dem sich Anforderungen häufig ändern.

Die Phasen B bis D sind die sogenannten Architektur-Entwicklungsphasen und sind von integraler Bedeutung. Jeder Arbeitsschritt wird anschließend mit den Requirements abgeglichen. Das stellt sicher, dass Änderungen in den Requirements gezielt reguliert und in den einzelnen Phasen umgesetzt werden.

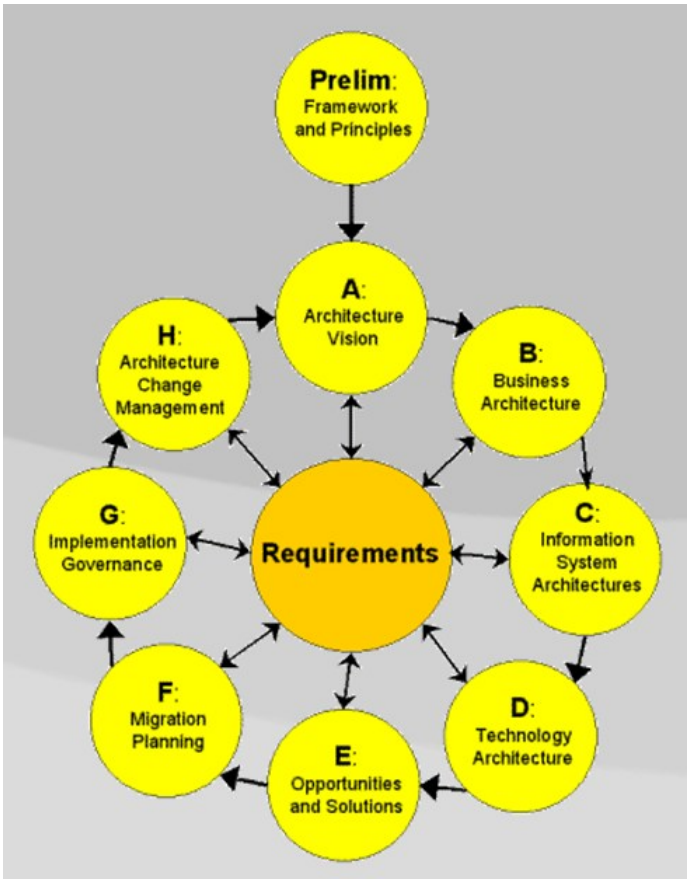


Abbildung 2.1: TOGAF Architecture Development Cycle. Quelle: `https://commons.wikimedia.org/wiki/File:TOGAF_ADM.jpg`, abgerufen am 30.12.2013

ArchiMate und TOGAF

ArchiMate® ist eine Beschreibungssprache im Stile der Unified Modeling Language (UML). Sie wurde ursprünglich unabhängig von TOGAF entwickelt, und im Jahre 2008 an The Open Group übergeben. 2009 wurde ArchiMate in Version 1.0 veröffentlicht, und 2012 in der Version 2.0. Die zweite Version [Ope12, S. 5] ist nicht nur abwärtskompatibel zur ersten, sondern auch kompatibel zum TOGAF ADM. Ende 2013 wurde ein kleineres Update auf Version 2.1 veröffentlicht, welches Kompatibilität zu 2.0 erhält und zusätzlich kleinere Änderung aufgrund von Rückmeldungen von Kunden umgesetzt hat. Näheres ist der offiziellen Dokumentation [Gro13] zu entnehmen, die für Mitglieder von The Open Group frei zur Verfügung steht, und für Nichtmitglieder in Form einer 90-tägigen Evaluierungs-Lizenz.

ArchiMate unterstützt TOGAF durch einen herstellerunabhängigen, graphischen Satz an Konzepten, die dabei helfen ein konsistentes, integriertes Modell zu schaffen, welches in der Form von TOGAF Views dargestellt werden kann [Gro13, Kapitel 2.9].

Die über 180 Seiten starke Spezifikation [Gro13] von ArchiMate definiert drei grundlegende Ebenen: Der Business Layer für Geschäftsobjekte, der Application Layer für Software-Applikationen und der Technology Layer für Hardware. TOGAF deckt einen breiteren Bereich ab als ArchiMate. Folgende Stellen des TOGAF ADM und von ArchiMate sind äquivalent:

1. *Phase B: Business Architecture* ist äquivalent zu *Business Layer*.
2. *Phase C: Information Systems Architecture* ist äquivalent zu *Application Layer*.
3. *Phase D: Technology Architecture* ist äquivalent zu *Technology Layer*.
4. *Phasen E-G* sind äquivalent zu *Implementation and Migration Extension*.

Die weiteren Phasen des ADM werden nicht von ArchiMate abgedeckt. Obwohl sich ArchiMate also sehr gut in TOGAF einfügen kann, wird es von vielen Seiten (z.B. von [Mat11]) als eigenständiges EA Framework betrachtet.

2.3.4 Department of Defense Architecture Framework (DoDAF)

DoDAF wird vom U.S. Department of Defence (DoD) entwickelt, die Dokumentation [U.Sst] ist daher nur auf Englisch verfügbar. Da das Framework von einer Behörde stammt, gibt es keine offizielle Zertifizierung und keinen Herstellersupport [Mat11, Kapitel 4.1.9]. Außerdem ist es explizit auf die Bedürfnisse des DoD zugeschnitten. Mit der Version 2.0 ist DoDAF umstrukturiert worden, und ist nun Daten-zentriert aufgebaut. Das bedeutet, es stellt die Sammlung, Speicherung und Pflege von Daten, welche für effektive Entscheidungen benötigt werden, in den Vordergrund [U.Sst, S. 4].

Eine der Besonderheiten von DoDAF sind die gleich acht definierten Sichtweisen (Viewpoints genannt) auf das Unternehmen [U.Sst, S. 105f]:

- Der *All Viewpoint* beschreibt die alle weiteren Viewpoints übergreifenden Architekturaspekte.

- Der *Capability Viewpoint* beschreibt das „Capability Portfolio“ und stellt Ansprüche an die Leistungsfähigkeit des Unternehmens.
- Der *Data and Information Viewpoint* zeigt die Strukturen der Daten-Beziehungen und -Abgleiche.
- Der *Operational Viewpoint* beinhaltet operative Szenarien, Aktivitäten und Ansprüche.
- Der *Project Viewpoint* beschreibt die Verbindung der operativen Ansprüche mit den Leistungsansprüchen, sowie Abhängigkeiten von Systems-Engineering-Prozessen, System-Design und Service-Design zum „Defense Acquisition System Process“.
- Der *Services Viewpoint* drückt die Leistung von Akteuren, Aktivitäten, Services und deren Interaktionen hinsichtlich der Unterstützung von operativen und Leistungs-Funktionen aus.
- Der *Standards Viewpoint* repräsentiert die operativen, geschäftlichen, technischen und industriellen Strategien, Standards, Richtlinien, Beschränkungen und Prognosen.
- Der *Systems Viewpoint* repräsentiert den Legacy-Support zu vorherigen DoDAF-Versionen und beschreibt die Systeme, ihre Zusammensetzung, ihre Interkonnektivität und mehr.

Diese Viewpoints werden jeweils erst dann umgesetzt, wenn die Entscheidungsträger Information von ihnen brauchen. Dieser Zugang soll die Daten-zentrierte Funktionsweise von DoDAF unterstreichen [U.Sst, S. 108].

2.3.5 Architektur integrierter Informationssysteme (ARIS)

ARIS wurde ursprünglich im Jahre 1991 an der Universität des Saarlandes von Professor August-Wilhelm Scheer entwickelt und ist heute in der Hand der IDS Scheer AG [Mat11, Kapitel 4.1.4]. Auf Anfrage bei der IDS Scheer AG erhält man die 2800 Seiten starke Dokumentation kostenlos. Außerdem bietet sie kostenpflichtige Trainings- und Supportdienstleistungen an. ARIS ist hauptsächlich in Europa verbreitet [Mat11, S. 73].

Abbildung 2.2 zeigt das „ARIS-Haus“, eine graphische Darstellung der Architektur von ARIS. Der Hauptfokus von ARIS liegt in den Geschäftsprozessen, welche auf fünf verschiedene Sichtweisen betrachtet werden. Diese fünf Sichtweisen bilden die Bausteine des ARIS-Hauses [Mat11, S. 73f]:

1. Organization View für sämtliche Organisationseinheiten (Menschen und Geräte)
2. Data View für einen logischen und zeitlichen Ablaufplan der Sichten
3. Control View für Informationsobjekte
4. Function View für Unternehmensziele und Tätigkeiten um diese zu erreichen
5. Product/Service View für sämtliche Produkte und Dienstleistungen des Unternehmens

Jede der fünf Sichten kann in die drei Elemente des *Lifecycle-Modells* aufgeteilt werden [Mat11, S. 74]:

1. Anforderungs- bzw. Fachkonzept

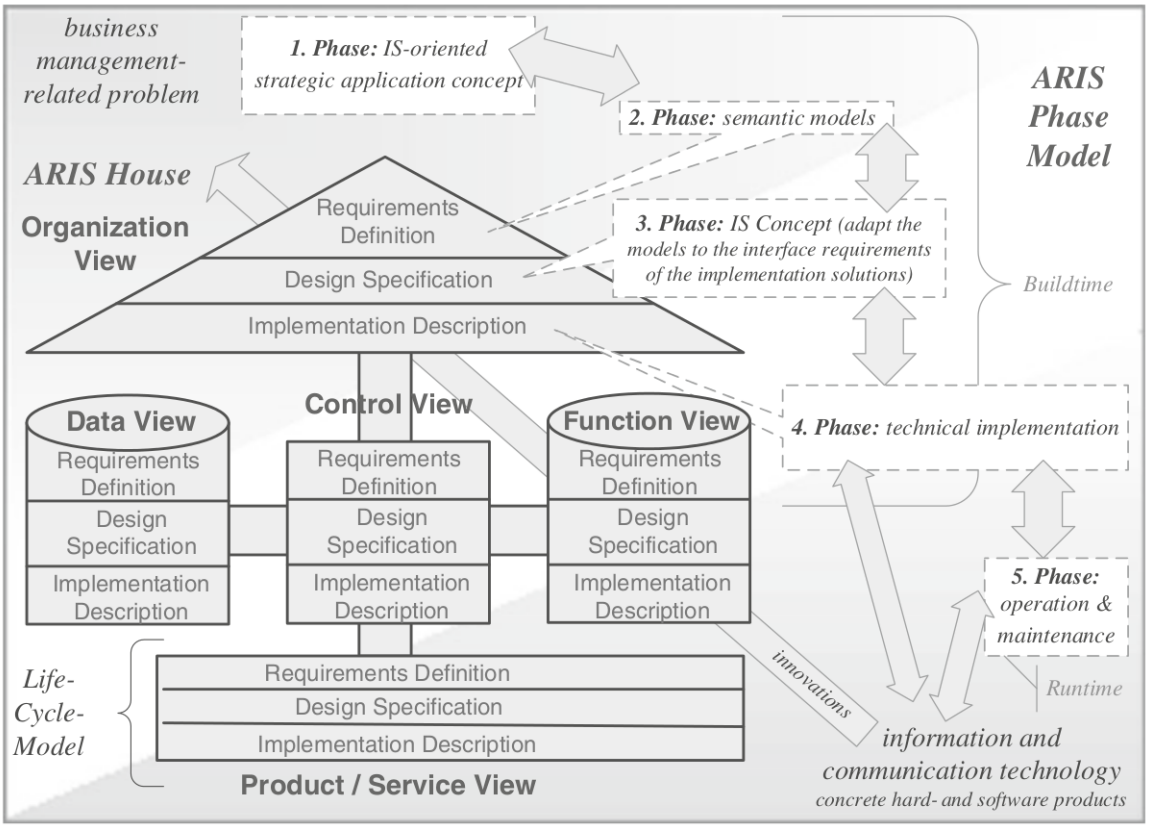


Abbildung 2.2: Das ARIS-Haus und -Phasenmodell. Quelle: [Mat11, S. 75]

2. Design-Spezifikation
3. Implementierung

Der letzte Baustein, das *ARIS-Phasenmodell*, ist ein Referenzmodell zur Vorgehensweise bei ARIS. Die Phasen dieses Modells starten mit einem globalen, sichtenübergreifenden Anwendungskonzept, gefolgt von den drei Lifecycle-Phasen aller Sichten und enden mit der Run-Time-Phase, die den Betrieb des Informationssystems umfasst.

Es existiert keine explizite Ebenentrennung, wie sie in Kapitel 2.1 vorgestellt wurde. Die graphische Notation von ARIS ist gemäß Lankhorst zwar sehr solide, aber auch sehr umfangreich und schwer zu lernen [Lan09, S. 33]. Außerdem ist ARIS laut Lankhorst nur im Bereich der Business-Modellierung flexibel, nicht erweiterbar und die Integration der unterschiedlichen Sichten ist nur mangelhaft umgesetzt [Lan09, S. 34].

2.4 EAM in der wissenschaftlichen Literatur

EAM ist eine noch relativ junge Disziplin. Dieses Unterkapitel stellt einen Auszug aktueller Forschungsarbeiten rund um EAM vor.

Die Masterarbeit [Usm12] untersucht die Struktur der beiden Modellierungssprachen BPMN 2.0 und ArchiMate. Zu diesem Zweck werden BPMN und der Business Layer von ArchiMate in Petri-

Netze transferiert um semantische Analysen durchzuführen. Auf diese Weise wird bewiesen, dass sich ArchiMate-Business-Modelle in BPMN-Modelle überführen lassen und umgekehrt.

In der Masterarbeit [Ett05] wird untersucht, ob ArchiMate oder UML als Architekturbeschreibungssprache für ein bestimmtes Projekt verwendet werden soll. Dafür werden die beiden Sprachen einer Qualitäts- und Geschäftspotenzial-Analyse unterzogen. Die Qualitäts-Analyse erfolgt mit einer Methode zur Modell-Komplexitäts-Analys namens „Method Points Analysis“ und mit Hilfe von „Semiotic Theory“. Die Geschäftspotenzial-Analyse erfolgt anhand eine Fallstudie. Die Auswertung der Analysen ergibt eine klare Empfehlung für ArchiMate.

Die Masterarbeit [Rut11] der Technischen Universität München vergleicht die IT Governance Frameworks ITIL, COBIT mit der Universitäts-eigenen Entwicklung BEAMS – einem Bausteinsystem für EAM. Es wird auf Unterschiede und Berührungspunkte eingegangen, sowie eine mögliche Koexistenz ausgearbeitet.

Die Masterarbeit [AT09] modelliert mit einem EA Framework die Arbeitsabläufe eines Dokumentenmanagement Systems (DMS). Zuerst werden die Konzepte verschiedener EA Frameworks betrachtet. Von diesen Frameworks wird ArchiMate zur Modellierung des DMS ausgewählt. Sämtliche Arbeitsabläufe mit dem DMS werden mit ArchiMate modelliert und schließlich auf der Plattform Microsoft Sharepoint umgesetzt.

Das Buch [HP13] umfasst acht Papers der Tagung „Practice-Driven Research on Enterprise Transformation“ im Jahr 2013. Es beinhaltet praktische Erfahrungen und Fallbeispiele rund um Enterprise Architecture und Enterprise Transformation. Diese Tagung war bereits die sechste ihrer Art und es gibt auch Bücher der vorhergehenden.

Das Buch [LPW+09] betrachtet das Thema Enterprise Architecture aus einem allgemeineren Blickwinkel. Es beleuchtet die Historie von EAM und den Mehrwert, den es generieren kann. Außerdem werden Frameworks analysiert und generelle Arbeitsweisen zum Durchführen von EAM vorgeschlagen.

Der Tagungsbericht [LA14] betrachtet EAM aus dem Blickwinkel einer *Enterprise Transition*, also dem Prozess einer Unternehmensänderung. EAM kann nützliche Information für diesen Prozess bereitstellen. In dem Bericht wird untersucht, welche Informationen erfolgskritisch für einen erfolgreichen Änderungsprozess sind. Er kommt zu dem Schluss, dass EAM diese Informationen liefern und somit Enterprise Transition unterstützen kann.

3 Fallstudie

In diesem Kapitel wird die Einführung von Enterprise Architecture Management (EAM) im Partnerunternehmen beschrieben. Dazu werden zuerst die beteiligten Stakeholder-Rollen und ihre Wünsche an das Modell genannt. Danach wird daraus ein werkzeugunabhängiges Schema entwickelt und die dadurch gewonnenen Informationen werden genutzt, um ein Enterprise Architecture Framework auszuwählen, in welchem das Modell realisiert ist.

3.1 Werkzeugunabhängige Modellentwicklung

In der ersten Phase der Modellierung werden die Wünsche der Stakeholder an das Modell aufgenommen und aufbereitet. Wenn das Modell erstellt ist und von den bisherigen Stakeholdern aktiv verwendet wird, ist es wahrscheinlich, dass die bisherigen und weitere Stakeholder weitere Anforderungen an das Modell stellen werden. Soll sich das EAM in der Firma ausbreiten und etablieren, also von möglichst vielen Personen verwendet werden, ist es daher essentiell, auch nach Beendigung dieser schriftlichen Arbeit das Modell ständig weiterzuentwickeln und zu betreuen.

3.1.1 Stakeholder-Rollen

Verschiedene Personen aus unterschiedlichen Abteilungen haben Interesse an dem Ergebnis des Modells. Diese Stakeholder repräsentieren folgende Rollen im Unternehmen:

ManagerIn

ManagerInnen gehören der mittleren und höheren Management-Ebene an und bestimmen die Entwicklung des großen Gesamtbilds beziehungsweise der Gesamtstrategie des Unternehmens und der IT-Abteilung. Sie müssen Sorge tragen, dass die IT-Systeme die Geschäftsprozesse bestmöglich unterstützen und sie müssen ihre Entscheidungen gegenüber ihren Vorgesetzten und KollegInnen rechtfertigen können.

Prozess-VerantwortlicheR

Prozess-Verantwortliche haben die Kontrolle über die Geschäftsprozesse, wie beispielsweise *Projektmanagement* oder *Sales*. Ein Prozess besteht aus definierten und dokumentierten Tätigkeiten und Tätigkeitsabfolgen, welche von AkteurInnen in der Regel mithilfe von IT-Systemen ausgeführt werden. Mit EAM kann der Ist-Zustand, ein Soll-Zustand und ein Übergang von Ist zu Soll eines solchen Prozesses dargestellt werden. Im Partnerunternehmen sind diese Prozesse bereits gemäß dem Qualitätsmanagementstandard ISO 9001 definiert. Diese Definitionen werden so weit ins EAM übernommen, wie es für die Stakeholder von Interesse ist.

IT-MitarbeiterIn

Vertreter dieser Rolle haben Verantwortung über gewisse Applikationen oder Server. Die Kenntnis der Datenquellen und der anderen Abhängigkeiten ihrer betreuten Systeme ist für sie essentiell. EAM kann ihnen helfen, ihre Systeme anderen Personen zu erklären und sich mit ihren KollegInnen über Umstrukturierungen abzustimmen. Besonders hilfreich ist EAM im Falle von Urlaubsvertretung oder Krankheitsfällen. Der/die vertretende KollegIn kann sich im Modell schnell orientieren und den Ist-Zustand in Erfahrung bringen.

3.1.2 Anforderungen der Stakeholder

Für die erste Iteration des Modells haben die Stakeholder in einführenden Gesprächen folgende Wünsche geäußert.

Die *ManagerInnen* wünschen sich als Ergebnis der Arbeit einen klar verständlichen Überblick über die IT-Systeme für die Management-Ebene. Sie wünschen sich ein übersichtliches Bild über die bestehende System-Landschaft. Das soll dem Management ein gemeinsames Verständnis der Ist-Situation als Ausgangsbasis für Diskussionen liefern.

Das Modell soll ihnen außerdem klar ersichtlich machen, welche die „führenden“ Systeme der Systemlandschaft sind. Man soll damit erkennen können, welche Systeme die meisten Daten halten und verteilen.

Ein weiterer Wunsch der Manager ist es, sehen zu können, wie standardisiert die Applikationslandschaft aktuell ist. Das Unternehmen hat eine Vielzahl an Applikationen in Verwendung. Einige davon sind Eigenentwicklungen und einige sind Standardprodukte, wie beispielsweise Microsoft Exchange Server. Welche Applikationen sind Industriestandard? Welche sind Eigenentwicklungen?

Darüberhinaus soll ersichtlich werden, in welchen Niederlassungen eine Applikation in Verwendung ist.

Wenn das Modell in der ersten Iteration abgeschlossen ist, möchten die ManagerInnen aufgezeigt bekommen, welchen zusätzlichen Nutzen das EAM durch weitere Ausbauschritte bringen kann. Außerdem sollen Vorschläge ausgearbeitet werden, wie die fortlaufende Pflege und Nutzung des EAM in das Tagesgeschäft integriert werden kann und wie Engagement bei den MitarbeiterInnen erzeugt werden kann.

Die Anforderungen der *IT-MitarbeiterInnen* an das Modell sind in mehreren Gesprächen aufgenommen und konkretisiert worden. Das Modell soll ihnen sämtliche für sie bedeutende Informationen über die Applikationslandschaft zur Verfügung stellen. Dies kann insbesondere in Krisen- und in Vertretungssituationen wertvolle Zeit bei einer Fehlerursachen-Suche ersparen. In der folgenden Liste sind die Anforderungen des Teams zu finden:

1. Welche Applikationen und technische Services existieren und von wem werden sie technisch bzw. fachlich betreut?
2. Welche Schnittstellen zwischen Applikationen existieren, wie sind sie realisiert und welche Daten tauschen sie aus? Wie wird der Austausch gesteuert?
3. Welche Daten existieren, und wo werden sie gehalten?
4. Wann finden regelmäßige Datentransfers/-abgleiche statt?
5. Welche Systeme müssen angepasst werden, wenn eine Applikation abgelöst wird?
6. Welche Systeme sind betroffen, wenn ein Server, eine Datenbank oder eine Applikation ausfällt?

Die IT-MitarbeiterInnen haben dadurch essenzielle Informationen zentral dokumentiert. Neue KollegInnen sehen so beispielsweise auf einen Blick, wer in welchen Fragen zu konsultieren ist. Man kann ablesen, wo bestimmte Daten gehalten werden oder welche Server in Verwendung sind, wenn eine bestimmte Geschäfts-Tätigkeit durchgeführt wird.

Die *Prozess-Verantwortlichen* möchten in dem Modell abgebildet haben, welche Tätigkeiten eines Geschäftsprozess von welcher Rolle und mit welcher IT-Applikation durchgeführt wird. Ein Geschäftsprozess umfasst mehrere Tätigkeiten zum Erzeugen einer geschäftsrelevanten Leistung.

3.1.3 Erstellung eines werkzeugunabhängigen Schemas

Aus den freisprachlichen Anforderungen des vorigen Kapitels werden nun strukturiertere, semi–formale Aussagen ausgearbeitet. Auf Basis dieser Aussagen wird danach das Schema des Modells entworfen.

3.1.3.1 Modell-Aussagen

Im Folgenden werden die Aussagen des Modells textuell formuliert. Diese Aussagen dienen dazu, die Wünsche und Anforderungen der Stakeholder zu konkretisieren, und sie bilden die Basis für die Modellierung. Diese Aussagen sind zuvor aber noch von den Stakeholdern zu validieren.

Applikation

Applikationen sind die zentralsten Elemente der Anforderungen. Eine Applikation ist zum Beispiel eine Server-Applikation, eine Web-Applikation, oder eine lokale Client-Applikation. Sie besitzt einen eindeutigen Namen und eventuell weitere in der Firma geläufige Bezeichnungen, die als Synonyme angeführt werden sollen.

Die grundlegendste Aussage des Modells über eine *Applikation* lautet:

> Das Modell repräsentiert die Applikation mit Namen X.
> Optional: Die Applikation X hat die Synonyme Y und Z.

Die Grenzen einer Applikation sind nicht immer eindeutig, sondern oftmals von der Modellierung abhängig. Zwei unterschiedliche Software-Produkte können zum Beispiel eine gemeinsame Code-Basis haben. Aus technischer Sicht sind sie eine einzige Applikation mit zwei unterschiedlichen Oberflächen, aus Nutzersicht sind es zwei unterschiedliche Applikationen. Da für diese Modellierung beide Sichten von Belang sind, würde dieses Beispiel wie folgt modelliert werden: Die zwei Applikationen, die der Nutzer sieht, sind eingebettet in eine weitere Applikation, welche die gemeinsame Code-Basis repräsentiert.

Um solche komplexeren Gegebenheiten darstellen zu können, wird folgende Aussage benutzt:

> Die Applikation X ist eingebettet in die (Host-)Applikation Y.

Server

Jede Applikation muss auf einem Server beziehungsweise auf lokalen Arbeitsstationen laufen. Es kann auch auf mehrere Server aufgeteilt sein (zum Beispiel Applikations- und Datenbankserver).

Die Modell-Aussage über Server lautet:

> Die Applikation X läuft auf den Servern Y und Z.

Standardisierungsklasse

Um die Standardisierungsklasse der Applikations-Landschaft zu erkennen, soll jede einzelne Applikation bezüglich ihrer jeweiligen Standardisierung bewertet werden. Der gewünschte Effekt ist es, erkennen zu können, wieviel Eigenleistung in einem Produkt steckt, ob eine hohe Herstellerabhängigkeit vorherrscht und ob neue MitarbeiterInnen im Fachbereich damit bereits vertraut sein können oder Schulungen benötigen.

Wenn ein Standardprodukt für ein Unternehmen stark angepasst wurde, können bei einem Update Schwierigkeiten auftreten und es muss entsprechend mehr Aufwand und Zeit dafür eingeplant werden. Die Vorteile eines Standard-Produkts gegenüber einer Eigenentwicklung sind der vergleichsweise geringere Aufwand an Eigenleistung seitens der internen IT, ein professioneller Support der Hersteller-Firma, sowie ein hoher Wiedererkennungswert bei neuen MitarbeiterInnen.

Demgegenüber stehen neben einer gewissen Hilflosigkeit der IT-Abteilung bei Fehlern in der Software oftmals noch die Lizenzkosten, die je nach Vertrag auch nachträglich erhöht werden können. Außerdem können Software-Anpassungswünsche aus dem Fachbereich oft nicht oder nur mit sehr hohem Aufwand umgesetzt werden. Hinzu kommt, dass nicht jedes Standardprodukt mit geringem Aufwand im Unternehmen eingeführt werden kann. SAP ist zum Beispiel ein Standardprodukt, das mit hohem Aufwand konfiguriert werden muss.

Eigenentwicklungen sind sehr gut an die individuellen Wünsche des Fachbereichs anpassbar. Es ist möglich, Bugs selbst zu finden und zeitnah auszubessern und die Kosten von Wartung und Weiterentwicklung sind gut planbar. Andererseits werden die in internen IT-Abteilungen entwickelten Produkte oftmals von bedeutend weniger EntwicklerInnen programmiert als Standardprodukte. Eine geringere Anzahl an Software-Features ist also wahrscheinlich. All diese Merkmale gelten natürlich nicht universell, sondern sind eher als ungefähre Richtlinien zu sehen, die individuell bewertet werden müssen.

Standard-Produkte und Eigenenentwicklungen haben beide Stärken und Schwächen. Da es keine Musterlösung gibt, muss individuell entschieden werden, welchen Weg man einschlägt.

Wenn allerdings ein Standardprodukt so weit verändert wird, dass weder die Stärken von Eigenentwicklungen noch die von Standardprodukten noch greifen, sollten die Gründe dafür herausgefunden und die Produktwahl überdacht werden.

Gemeinsam mit den IT-MitarbeiterInnen wurden die fünf Standardisierungsklassen aus Tabelle 3.1 festgelegt, nach denen die Applikationen bewertet werden sollen.

Die Aussage des Modells über die *Standardisierungsklasse* lautet:

> Die Applikation X hat die Standardisierungsklasse Y.

Diese Aussage muss für jede Applikation getroffen werden, wobei Y genau einem Element aus Tabelle 3.1 entspricht.

Applikations-Betreuung

Es gibt zwei Arten, auf die eine Applikation betreut werden muss: technisch und fachlich. Die technische Betreuung umfasst die Wartung, Umsetzung der Weiterentwicklung und den technischen Support in Fehlerfällen. Die fachlichen oder auch inhaltlichen BetreuerInnnen vertreten den NutzerInnenkreis der Applikation. Sie pflegen die Daten und stellen Anforderungen an die Weiterentwicklung der Applikation. Jede Applikation hat mindestens eine(n) technische(n) und mindestens eine(n) fachliche(n) BetreuerIn. Ein(e) BetreuerIn kann entweder eine Person, eine Abteilung oder eine andere Form von Team sein.[1]

Die Aussage des Modells über *Applikations-Betreuung* lautet:

> Die Applikation X hat Person/Abteilung/Team Y als technische Betreuung.
> Die Applikation X hat Person/Abteilung/Team Z als fachliche Betreuung.

[1] Erst zu einem späteren Zeitpunkt ist aufgefallen, dass eine Rollen-Definition vorzuziehen gewesen wäre. Personen oder Teams in so einem Modell abzubilden hat den Nachteil, dass Personen das Unternehmen verlassen, Teams sich auflösen oder mit anderen Teams fusionieren können. Diese Änderungen müssen im Modell dann ebenfalls nachgebessert werden.

Nr	Standardisierungsklasse	Beschreibung
1	Standardprodukt	Die Software stammt von einem Dritt-Hersteller und ist für ihre Verwendungszwecke allgemein weit verbreitet. Sie wurde nur zu einem für den Betrieb erforderlichen Mindestmaß angepasst. Das bedeutet, dass Personen, die in einem anderen Unternehmen damit gearbeitet haben, praktisch keinen Unterschied bemerken.
2	Standardprodukt mit geringer Anpassung	Die Software stammt von einem Dritt-Hersteller und ist für ihre Verwendungszwecke allgemein weit verbreitet. Sie wurde in manchen Aspekten verändert und die Anpassungen beeinflussen die Usability, Arbeitsabläufe oder bringen neue Features.
3	Standardprodukt mit wesentlicher Anpassung	Die Dritt-Hersteller-Software ist in wesentlichen Aspekten abgewandelt. Neue Benutzer werden sich wenig vertraut fühlen, und Updates des Kernprodukts ziehen einen Arbeitsaufwand bei der Anpassung mit sich.
4	Fremdentwicklung	Die Software ist von einem Dritt-Hersteller und ist kein üblicher Industriestandard. Typischerweise ist es eine individuell an die Anforderungen des Partnerunternehmens angepasste (oder sogar von Grund auf erstellte) Entwicklung.
5	Eigenentwicklung	Die Software ist eine Eigenentwicklung des Partnerunternehmens. Oder sie wurde bei einem Dritt-Hersteller in Auftrag gegeben, wobei das Partnerunternehmen aber alle Rechte für den Quellcode besitzt.

Tabelle 3.1: Standardisierungsklassen der verwendeten Softwareprodukte

Schnittstellen und Daten

Applikationen können über Schnittstellen Daten mit anderen Applikationen austauschen. Dabei gibt es immer einen Auslöser (Trigger), der den Austausch startet. Es werden zwei Typen von Trigger unterschieden: Ein Trigger kann entweder die Aktion eines Benutzers sein („User“), oder zu bestimmten Zeitpunkten bzw. in bestimmten Zeitintervallen ausgelöst werden („Schedule“). Zusätzlich zum Trigger-Typ ist die jeweilige Bedingung anzugeben. Bei Schedule-Typen ist das zum Beispiel die Uhrzeit, bei User-Triggern die jeweilige User-Aktion.

Die Aussage des Modells über *Schnittstellen* lautet:

> Über die Schnittstelle S wird das Datenobjekt D von Applikation X zu Applikation Y übertragen. Optional (wenn bekannt): Die Schnittstelle S wird von Trigger T gestartet. T ist entweder vom Typ „User“ oder vom Typ „Schedule“, geht von Applikation Z aus und hat die Bedingung Y.

Jedes *Datenobjekt* kann zwischen Applikationen ausgetauscht werden. Es gibt jedoch immer genau eine Masterapplikation welchem die Hoheit über dieses Datenobjekt obliegt. Daten können aber auch in Form von Dateien vorliegen (typischerweise .csv- oder .txt-Formate), in diesem Fall hätte die Datei die Hoheit über dieses Datenobjekt. Nur eine der folgenden beiden Aussagen kann pro Datenobjekt verwendet werden, je nachdem ob es einer Applikation oder einer Datei zugeordnet ist.

Das Datenobjekt D hat Applikation X als Masterapplikation. Das Datenobjekt D hat Datei X als Masterdatei.

Abhängigkeiten von Applikationen

Zwischen zwei Applikationen kann es Abhängigkeiten geben. Das ist beispielsweise der Fall, wenn eine Applikation Daten mit einer anderen austauscht. Wenn eine Applikation A Daten von einer nicht verfügbaren Applikation B benötigt, tritt sofort eine Fehlfunktion irgendeiner Art auf. Wenn es Daten an B sendet, ist es abhängig von der Programmierung, ob bei A ein Fehler auftritt, ob der inkonsistente Datenstand in B berücksichtigt wird, und ob die Nicht-Verfügbarkeit von B an die Benutzer kommuniziert wird.

Es würde den Aufwand der Modellierung vervielfachen, wenn man das für jede Schnittstelle testen müsste. Aus diesem Grund werden beide Applikationen einer Schnittstelle in dem Modell als voneinander abhängig definiert.

Eine Applikation kann auch in ein anderes integriert sein, wodurch ebenfalls eine Abhängigkeit der eingebetteten Applikation zur Host-Applikation und umgekehrt entsteht. Ein Beispiel dafür sind Sharepoint[2]-WebParts. Das sind wiederverwendbare, programmierbare „Ausschnitte“ einer Sharepoint-Website, welche eine bestimmte Funktionalität zur Verfügung stellen. Ohne dem Sharepoint-System kann der WebPart klarerweise nicht funktionieren und ohne dem WebPart fehlt Sharepoint die entsprechende Funktionalität.

Darüber hinaus sind Applikationen auch von den verwendeten Servern abhängig (z.B. Datenbank-Server, Web-Server, etc.).

Zusammenfassend gibt es für eine Applikation *drei mögliche Abhängigkeiten*:

- Schnittstellen zu anderen Applikationen
- Einbettungen in andere Applikationen
- verwendete Server

Alle drei Punkte werden von bereits definierten Aussagen abgedeckt, es ist also für die Darstellung von Abhängigkeiten einer Applikationen keine zusätzliche Aussage notwendig.

Geschäftsprozesse und Tätigkeiten

Geschäftsprozesse sind thematische Gruppierungen von Tätigkeiten im Partnerunternehmen. Ein Beispiel dafür ist der Sales-Prozess, welcher sämtliche Sales-bezogene Tätigkeiten umfasst. Eine Sales-Tätigkeit ist beispielsweise das Pflegen von Kontaktdaten in der Customer Relationship Management (CRM) Software.

Die Tätigkeit T ist Teil des Geschäftsprozesses P.

Geschäftsprozesse können untergliedert werden in mehrere Sub-Geschäftsprozesse.

[2]Sharepoint ist ein Tool für Content- und Dokumenten-Management der Firma Microsoft.

Der Geschäftsprozess P ist untergliedert in Sub-Geschäftsprozesse SP1 und SP2.

Eine Tätigkeit wird von einer Rolle ausgeführt, welche dazu meistens eine IT-Applikation verwendet. Es gibt auch Tätigkeiten, für die keine Applikation gebraucht wird, diese werden in dem Modell allerdings nicht berücksichtigt, in der ISO9001-Dokumentation hingegen schon. Die Rollen werden von der vorhandenen Dokumentation übernommen.

Die Tätigkeit T wird ausgeführt von den Rollen A, B und C.
Die Tätigkeit T wird ausgeführt unter Verwendung der Applikationen X, Y und Z.

Niederlassungen

Das Partner-Unternehmen hat weltweit Niederlassungen von unterschiedlicher Größe und mit unterschiedlichen Aufgabengebieten. Daher verwenden nicht alle Niederlassungen alle IT-Applikationen:

Die Niederlassung X verwendet die Applikation Y.

Diese Aussagen sind in Tabelle 3.2 zusammengefasst. Abbildung 3.1 zeigt die Zusammenhänge der durch diese Aussagen definierten Elemente in Form eines Entity-Relationship-Diagramms in der Bachman-Notation. Dieses Diagramm repräsentiert das Werkzeug-unabhängige Schema, welches die Basis für die Datenerhebung und für das Werkzeug-spezifische Modell ist.

1.	Das Modell repräsentiert die Applikation mit Namen X.
2.	Die Applikation X hat die Synonyme Y und Z. (optional)
3.	Die Applikation X ist eingebettet in die (Host-)Applikation Y.
4.	Die Applikation X läuft auf den Servern Y und Z.
5.	Die Applikation X hat die Standardisierungsklasse Y.
6.	Die Applikation X hat Person/Abteilung/Team Y als technische Betreuung.
7.	Die Applikation X hat Person/Abteilung/Team Z als fachliche Betreuung.
8a.	Das Datenobjekt D hat Applikation X als Masterapplikation.
8b.	Das Datenobjekt D hat Datei X als Masterdatei.
9.	Über die Schnittstelle S wird das Datenobjekt D von Applikation X zu Applikation Y übertragen.
10.	Die Schnittstelle S wird von Trigger T gestartet. T ist entweder vom Typ „User“ oder vom Typ „Schedule“, geht von Applikation Z aus und hat die Bedingung Y.
11.	Die Niederlassung X verwendet die Applikation Y.
12.	Die Tätigkeit T ist Teil des Geschäftsprozesses P.
13.	Der Geschäftsprozess P ist untergliedert in Sub-Geschäftsprozesse SP1 und SP2.
14.	Die Tätigkeit T wird ausgeführt von den Rollen A, B und C.
15.	Die Tätigkeit T wird ausgeführt unter Verwendung der Applikationen X, Y und Z.

Tabelle 3.2: Zusammenfassung der Aussagen des Modells

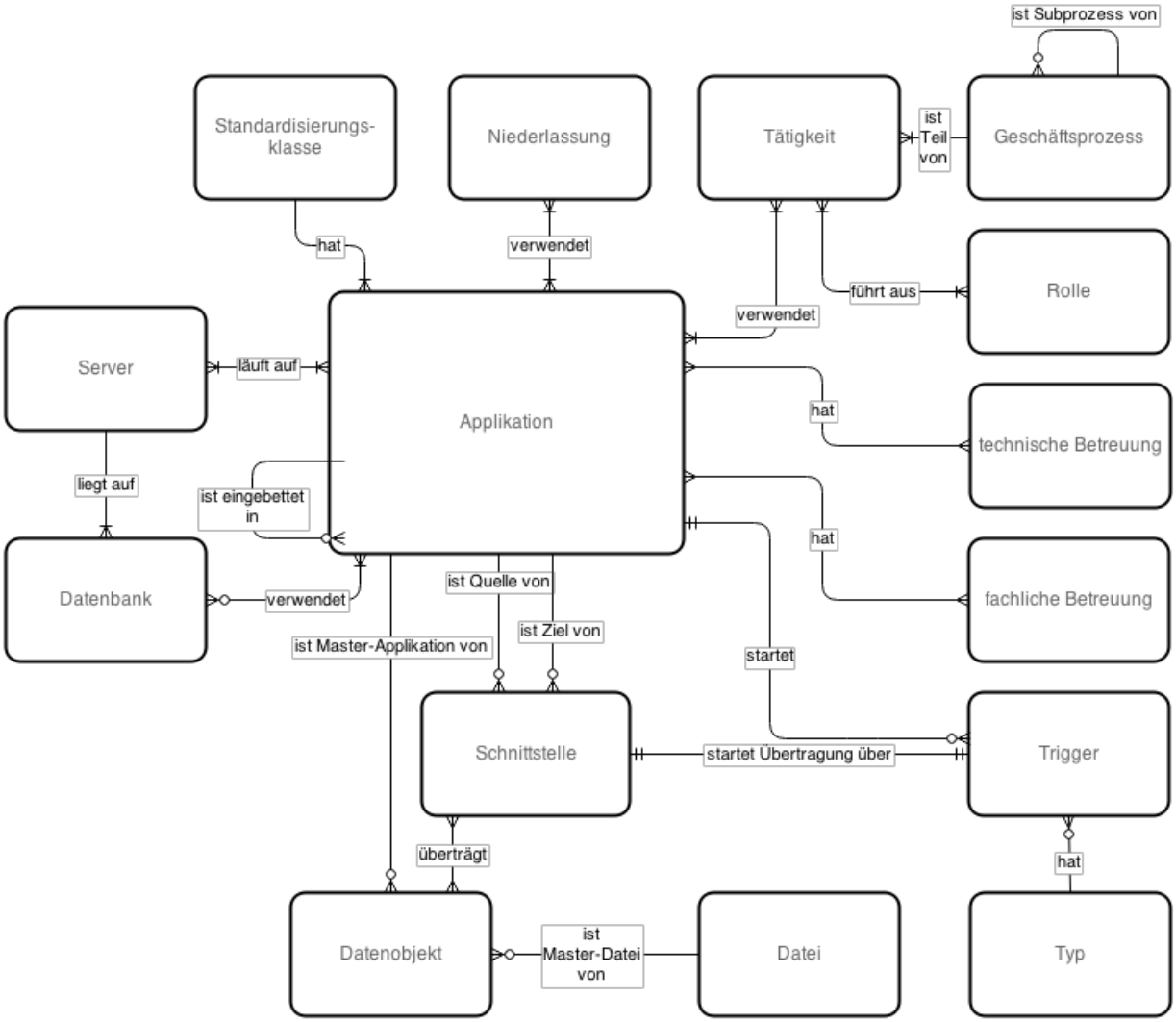


Abbildung 3.1: Werkzeug-unabhängiges Schema des Modells

3.1.3.2 Datenerhebung

Nachdem die Struktur des Modells nun definiert war, wurden als nächsten Schritt die Daten in Gesprächen mit den Stakeholdern erhoben und in einfachen Listen festgehalten. Die Listen entsprechen den Modell-Aussagen aus Tabelle 3.2. Die erste Liste beinhaltet das zentralste Element in dieser Modellierung, die Applikation.

Zu jeder *Applikation* werden die folgenden Eigenschaften (und Beziehungen) erhoben (In runder Klammer steht die erlaubte Anzahl der Einträge pro Spalte, in geschwungener Klammer die Zeilennummer der korrespondierenden Modell-Aussage aus Tabelle 3.2):

1. Name (1) {1}
2. Synonyme (0 bis a) {2}
3. Host-Applikationen (0 bis 1) {3}

4. Server (1 bis b) {4}
5. Standardisierungsklasse (1) {5}
6. technische Betreuung (1 bis c) {6}
7. fachliche Betreuung (1 bis d) {7}
8. Niederlassung (1 bis e) {11}

Schnittstellen besitzen ebenfalls einige Eigenschaften und werden in eigenen Listen erhoben:

1. Datenobjekt (1 bis f) {9}
2. Quell-Applikation (1) {9}
3. Ziel-Applikation (1) {9}
4. Trigger-Typ (0 bis g) {10}
5. Triggernde Applikation (0 bis h) {10}
6. Trigger-Bedingung (0 bis i) {10}

Datenobjekte werden mit folgenden Eigenschaften erhoben:

1. Name (1) {8}
2. Masterapplikation (1) {8}

Geschäftsprozesse werden ebenfalls in eigenen Listen erhoben:

1. Name (1) {12}
2. Eltern-Geschäftsprozess (0 bis 1) {13}
3. Tätigkeit (0 bis j) {12}
4. Rolle (1 bis k) {14}
5. Verwendete Applikationen (1 bis l) {15}

Die Gespräche fanden über einen Zeitraum von mehreren Wochen statt. Es hat sich gezeigt, dass bei einem einzigen Gespräch oftmals wichtige Details übersehen werden. Um dem gegenzusteuern wurden den Gesprächspartnern die Ergebnisse des ersten Gesprächs später in der EAM-Sprache erneut gezeigt. Das hatte einerseits den Effekt, dass sie die Konzepte der Sprache anhand von etwas Vertrautem schnell erfassten und andererseits ist ihnen oft durch die graphische Darstellung aufgefallen, wenn etwas fehlte. Die Ergebnislisten sind im Besitz des Partnerunternehmens. Eine Abbildung der Liste hätte zur Folge, dass diese Diplomarbeit gesperrt werden müsste. Aus diesem Grund wird davon abgesehen.

3.2 Auswahl geeigneter Tools

In diesem Kapitel werden ein Architektur-Framework und ein Software-Tool zur Realisierung des EAM ausgewählt und näher beschrieben.

3.2.1 Wahl des Frameworks

In Kapitel 2.3 wurden einige Enterprise Architecture (EA) Frameworks vorgestellt und in Kapitel 3.1 wurden die Anforderungen an das Modell der Fallstudie ausgearbeitet und das daraus resultierende Schema entwickelt.

Betrachtet man das Schema (Abbildung 3.1), dann erkennt man, dass die Elemente der Applikationsebene (Applikation, Standardisierungsklasse, Schnittstelle, Trigger, Typ) den Hauptfokus bilden und die Geschäftsebene, die Datenebene und die Infrastrukturebene weniger stark ausgeprägt sind.

Im Gegensatz dazu legt das EA-Framework ARIS, wie in Kapitel 2.3.5 beschrieben, seinen Fokus auf die Geschäftsebene. Es ist daher für diese Zwecke keine gute Wahl. Ebenso verhält es sich mit dem Framework DoDAF (siehe Kapitel 2.3.4) und seinem Daten-zentrierten Ansatz. Daher wurden diese beiden Frameworks als Kandidaten verworfen.

Das Zachman-Framework liefert eine große Menge an einfachen Modellen (Zellen in der Zachman-Matrix), die allerdings nur mangelhaft miteinander verknüpft sind (siehe Kapitel 2.3.2). Der Autor dieser Arbeit schätzt die Flexibilität des Frameworks allerdings als gering ein. Es wirkt unrealistisch, dass für jede möglicherweise aufkommende Frage ein einziges Zachman-Modell die Antwort liefern kann. Sobald allerdings mehrere der schwach verknüpften Modelle für eine Fragestellung/Aussage notwendig sind, wird die Schwäche des Frameworks offensichtlich. Daher wird es ebenfalls nicht für diese Modellierung verwendet.

Im Gegensatz dazu entwickelt man mit ArchiMate ein einziges Gesamt-Modell mit individuell und flexibel definierbaren Ausschnitten. Es weist in dieser Hinsicht also eine größere Flexibilität auf als das Zachman-Framework. Die Elemente des Schemas können vollständig mit den drei Ebenen von ArchiMate abgebildet werden, wie in Kapitel 3.3 gezeigt wird.

Wenn sich die Anforderungen an das EAM in Zukunft so sehr ausweiten sollten, dass sie mit ArchiMate nicht mehr abbildbar sind, kann das EAM mit TOGAF realisiert werden, ohne die bisherige Arbeit zu verlieren, da sich ein ArchiMate-Modell in TOGAF einfügt (siehe Kapitel 2.3.3).

Aus diesen Gründen wird das EAM für dieses Fallbeispiel mit den Konzepten von ArchiMate realisiert. Sollte sich herausstellen, dass ArchiMate nicht ausreicht, kann das Modell mit den Methodiken von TOGAF erweitert werden, unter Weiterverwendung der bisherigen Arbeit.

3.2.2 Wahl des Software-Tools

ArchiMate ist offen und herstellerunabhängig standardisiert und wird daher von mehreren Produkten verschiedener Hersteller unterstützt. Kommerzielle, von The Open Group zertifizierte Software, wie zum Beispiel wie BiZZdesign Architect [5], sind unter [4] zu finden.

Neben diesen kommerziellen Anbietern gibt es noch ein Open-Source-Produkt namens Archi [2]. Ursprünglich wurde Archi von Jisc [8] entwickelt und wird heute von einem Entwickler namens Phil Beauvoir als Open-Source-Projekt geführt [3].

Während die kommerziellen Tools allesamt mächtiger sind und nicht nur die ArchiMate-Sprache unterstüzen, ist Archi rein auf ArchiMate-Modellierung ausgelegt. Dieser Fokus auf ArchiMate anstatt gleichzeitiger Unterstützung mehrerer Standards ist eine Erleichterung für EinsteigerInnen. Darüberhinaus bietet es viele weitere einsteigerfreundliche Features. Abbildung 3.2 zeigt zum Beispiel den Ausschnitt eines Screenshots von Archi, auf dem die Hinweise („hints“) zu sehen sind, die stets zum aktuell ausgewählten ArchiMate-Objekt (in diesem Fall das *Data Object*) eingeblendet werden. Das ist eine gute Hilfe für AnfängerInnen, welche das regelmäßige Nachschlagen der Bedeutung der Objekte erleichtert. Diese kostenlose, einsteigerfreundliche Software wird

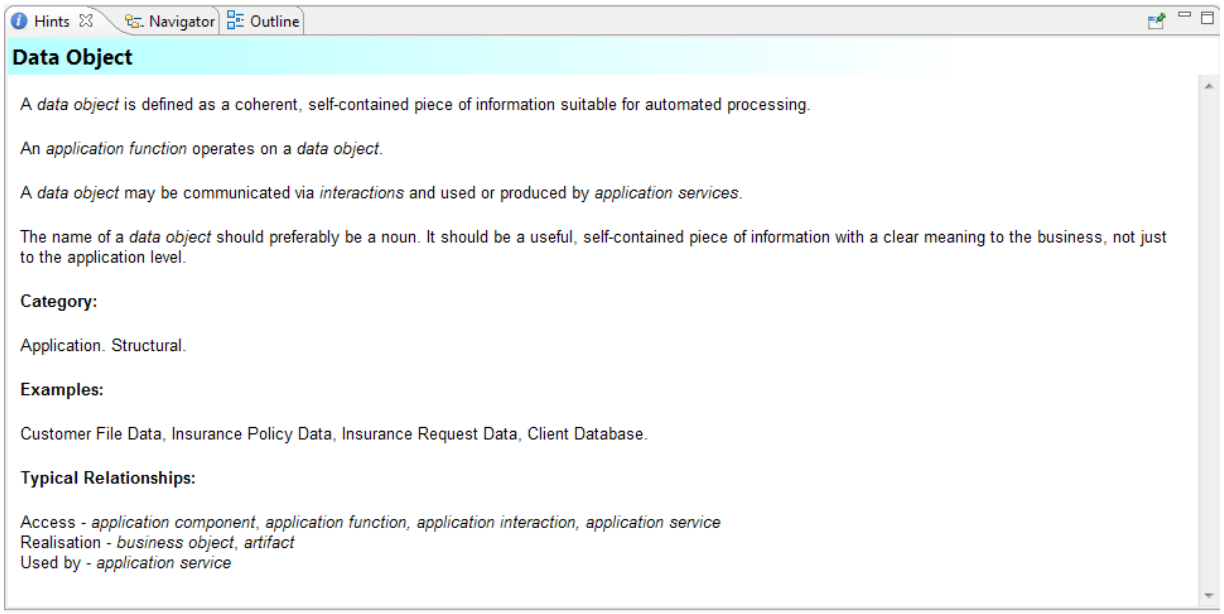


Abbildung 3.2: Screenshot der Hinweise („Hints“) von Archi zum im Tool ausgewählten ArchiMate-Objekt „Data Object“.

für dieses Projekt verwendet um erste Modellierungserfahrungen zu sammeln, ArchiMate besser kennenzulernen und im Zuge dessen die eigenen Anforderungen an die Software-Unterstützung zu konkretisieren. Falls es sich als notwendig erweisen sollte, kann im späteren Verlauf auf ein kommerzielles, mächtigeres Produkt umgestiegen werden (siehe Kapitel 3.4.2).

Archi: Funktionen und Übersicht

Abbildung 3.3 zeigt einen weiteren Ausschnitt der Archi-Oberfläche. Er befindet sich im linken, oberen Quadranten des Fensters und listet in einer Baum-Struktur sämtliche Elemente des aktuellen Modells. Das umfasst sowohl *Views* als auch Objekte wie *Applications* oder *Relations*.

Im Quadranten rechts davon findet man die aktuell geöffnete View mit ihren ArchiMate-Elementen und -Verbindungen (siehe Abbildung 3.4). In dieser Ansicht kann die View editiert werden. Noch weiter rechts ist die Palette, eine Auswahlmöglichkeit für die Elemente und Verbindungen.

Der linke untere Quadrant (Abbildung 3.5) ist in Tabs unterteilt. Der aktive Tab *Outline* zeigt die Gesamtübersicht über die aktuelle View. Der *Hints*-Tab wurde bereits in Abbildung 3.2 dargestellt und bietet Erklärungen. Der letzte Tab, *Navigation*, zeigt die Beziehungen des aktuellen Elements in Baumform. Wenn man durch diesen Baum navigiert, kann man dadurch Abhängigkeiten verfolgen.

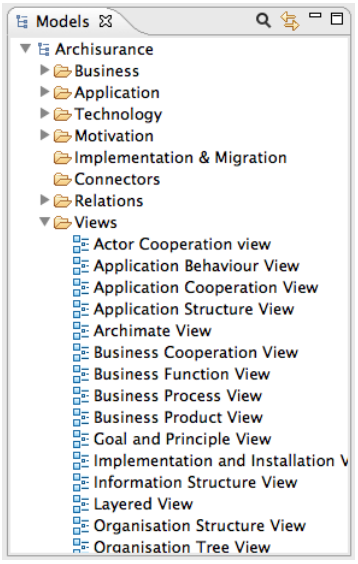


Abbildung 3.3: Teil des User Interfaces von Archi. Quelle: [2]

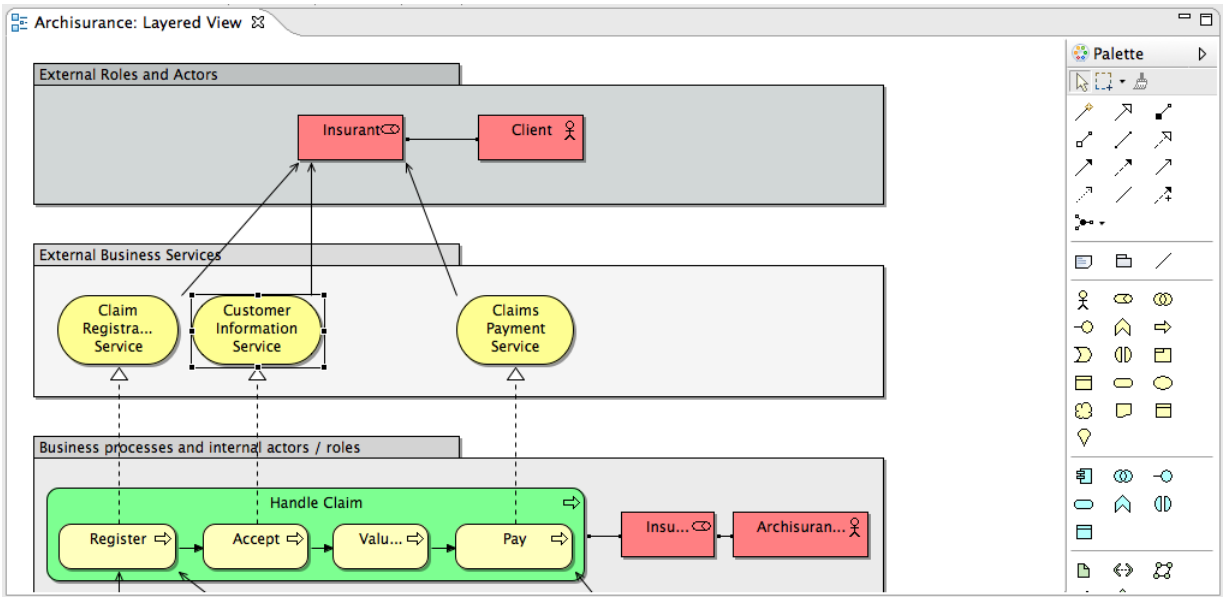


Abbildung 3.4: Teil des User Interfaces von Archi. Quelle: [2]

Im letzten Quadranten (Abbildung 3.6) ist der *Visualiser* zu finden, eine dynamische Sternförmige Darstellung der Verbindungen des aktuellen Elements. Diese Ansicht ist graphisch ansprechender und übersichtlicher als die Baumform des Navigations-Bereichs, allerdings lassen sich Abhängigkeiten nicht wie bei der Navigation über mehrere Elemente verfolgen. Die Visualiser-Ansicht hat dennoch ihren Zweck, und zwar deshalb, weil sie im Gegensatz zur aktuellen View immer sämtliche Verbindungen des Elements anzeigt. Eine View hingegen zeigt nur einen Ausschnitt des Modells mit den dafür relevanten Beziehungen. Die Visualiser-Ansicht kann hilfreich sein, um nachzuprüfen, ob eine gewisse Beziehung bereits in einer anderen View zur Ansicht gebracht wird oder ob dieser Aspekt noch nicht modelliert wurde.

Ein wichtiges Fenster ist in den Screenshots nicht zu sehen, das *Properties*-Fenster. Dieses Fenster

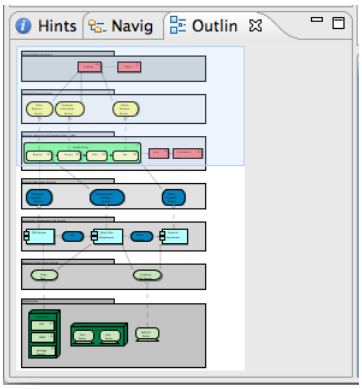


Abbildung 3.5: Teil des User Interfaces von Archi. Quelle: [2]

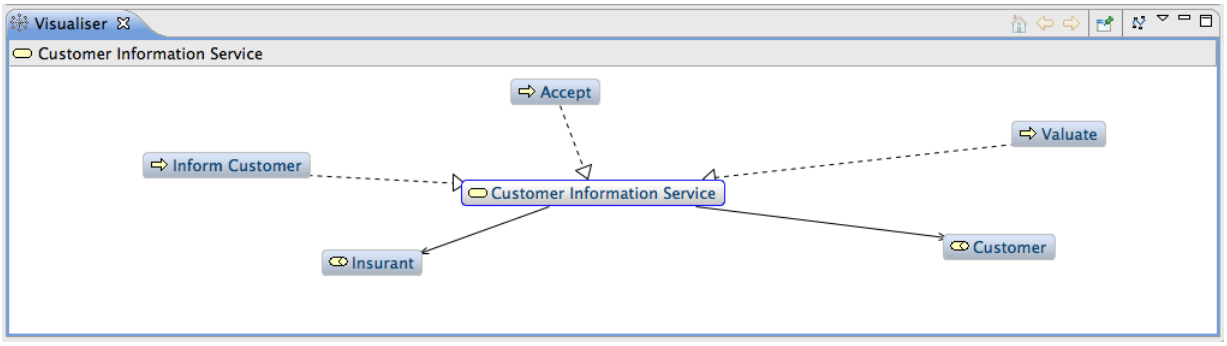


Abbildung 3.6: Teil des User Interfaces von Archi. Quelle: [2]

dient zur Eingabe von Eigenschaften des aktuellen Elements. Dazu zählen Standard-Eigenschaften wie Name und Beschreibung sowie selbst-definierbare Eigenschaften in Form von Schlüssel-Wert-Paaren zur Realisierung von *Profilen* (siehe Seite 28).

3.3 Werkzeug-spezifische Modellentwicklung

In diesem Kapitel werden die grundlegenden Konzepte von ArchiMate erklärt, um das Modell zu verstehen ohne zuerst die gesamte Dokumentation der Sprache lesen zu müssen. Ziel ist es, dass der Leser ArchiMate soweit überblickt, wie es zum Verständnis des Modells notwendig ist.

Im Anschluss wird das werkzeugunabhängige Schema aus Kapitel 3.1.3 in ein ArchiMate–Schema übersetzt. Danach wird exemplarisch ein Teilbereich des erstellten Gesamt-Modells dargestellt und erläutert.

3.3.1 Einführung in ArchiMate zum Verständnis des Modells

In diesem Kapitel folgt eine kurze Einführung in ArchiMate. Sie soll lediglich die Konzepte abdecken, welche zum Verständnis der Fallstudie notwendig sind. Zum Nachschlagen weiterführender Informationen wird auf die offizielle Dokumentation von ArchiMate [Gro13] verwiesen.

Um die Einstiegshürde für alle Stakeholder möglichst gering zu halten, sollte ein Modell gut lesbar und benutzbar sein. Das erreicht man, indem man nur so wenig Elemente und Elementtypen wie

unbedingt notwendig benutzt [Lan09, S. 137]. ArchiMate kennt aus diesem Grund im Gegensatz zu der weitaus bekannteren graphischen Sprache UML viel weniger Elemente [Gro13, Kapitel 2.1].

Ebenen

Wie auf Seite 10 bereits kurz erörtert, lassen sich die Elemente von ArchiMate drei Ebenen zuordnen:

- *Business Layer* für Geschäftsobjekte,
- *Application Layer* für Software–Applikationen und
- *Technology Layer* für Infrastruktur–Services.

Aspekte

Zusätzlich weisen die Elemente jeweils einen von drei *Aspekten* auf [Gro13, Kapitel 2.2]:

- Ein *active structure element* kann ein Verhalten (*behavior*) ausüben.
- Ein *behavior element* ist ein Verhalten, das von *active structure elements* ausgeübt werden kann.
- Ein *passive structure element* ist ein Objekt, auf welches Verhalten (*behavior*) ausgeübt werden kann.

Die drei Aspekte sind an die natürliche Sprache angelehnt, deren Sätze aus Subjekt (active structure element), Verb (behavior) und Objekt (passive structure element) bestehen.

Profile

Jeder Elementtyp in ArchiMate kann ein sogenanntes *Profil* aufweisen [Gro13, Kapitel 9.1]. Profile sind eine einfache Möglichkeit, die ArchiMate-Elemente generisch um typisierte Attribute zu erweitern. Somit kann beispielsweise ein Business-Service-Element die beiden Attribute Fixkosten und variable Kosten (beide vom Typ „Währung“) aufweisen.

In dem Tool Archi sind Profile in Form von nicht typisierten Schlüssel-Wert-Paaren, den sogenannten „Properties“, realisiert. Der Schlüssel ist dabei jeweils der Attributname, und der Wert ist der Wert des Attributes als Freitext. Jedes Objekt im Modell kann eine individuelle Anzahl und Zusammensetzung von Properties besitzen. Archi bietet keine Möglichkeit, bestimmte Properties für einen Elementtyp zu forcieren. Es liegt in der Verantwortung des Architekten, die geforderten Informationen zuverlässig bereitzustellen.

Die Properties von Archi sind also nicht gänzlich konform mit der Profil-Definition, welche eine Menge typisierter Attribute pro Element-Typ vorgeben. Mit Bedacht genutzt können sie aber ArchiMate-konform wie Profile verwendet werden. Die ArchitektInnen müssen dafür auf die korrekten Properties pro Element-Typ achten.

Viewpoints und Views

Views [Gro13, Kapitel 8] sind definierte Ausschnitte aus dem Modell. Sie sind dafür da, aus dem Gesamtmodell nur die für bestimmte Stakeholder beziehungsweise für einen bestimmten Zweck interessanten Aspekte darzustellen. Eine View wird durch ihren *Viewpoint* bestimmt. Metaphorisch gesagt ist eine View das, was man sieht und ein Viewpoint das, woher man blickt. Zur Unterstützung der/des ArchitektIn sind in dem Spezifikations-Dokument einige Viewpoints definiert und klassifiziert (siehe [Gro13, Kapitel 8.4])

Internes und externes Verhalten

Ein weiteres Konzept von ArchiMate sind interne und externe Verhaltens-Elemente des Business Layers [Gro13, S. 19]. Interne Elemente sind sämtliche Elemente des modellierten Unternehmens und externe repräsentieren Umweltfaktoren, ausgehend beispielsweise von Kunden, Geschäftspartnern oder Regierungen. Externe Elemente stellen also Faktoren dar, auf die das Unternehmen zwar keinen direkten Einfluss hat, auf die es allerdings auf irgendeine Weise reagiert beziehungsweise reagieren muss. Externe Elemente sind für diese Arbeit nicht relevant, weil es keine entsprechende Anforderung gibt. Da nur internes Verhalten modelliert wird, wird diese Unterscheidung nicht weiter getätigt.

Erweiterungen

Jenseits der beiden Konzepte Ebene und Aspekt existieren die optionalen Erweiterungen *Motivation* zur Modellierung von Anforderungen und Zielen, und *Implementation and Migration* zur Modellierung von Migrationsplänen und Implementierungsprojekten.

Diese beiden Zusätze werden für die Fallstudie nicht benötigt, könnten aber eventuell später in einer weiteren Ausbaustufe des EAM im Partnerunternehmen eingeführt werden.

Im Folgenden werden die für die Fallstudie verwendeten ArchiMate-Elemente und deren Bedeutung erklärt.

Business Layer

Das erste Element des Business Layers, welches hier vorgestellt wird, heißt *Business Process* (Abbildung 3.7, vgl. [Gro13, Kapitel 3.3.1]). Ein Business Process ist ein Verhaltens-Element, ausgeführt von einer *Business Role* zum Produzieren von Produkten und Dienstleistungen. Ein Business Process kann eine Gruppierung von mehreren, aufeinander folgenden Business Processes sein. Der Name eines Business Processes sollte ein Verb in der Gegenwartsform beinhalten, wie zum Beispiel „Beschwerde bearbeiten“.

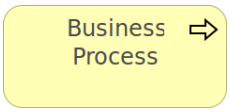


Abbildung 3.7: Abbildung des ArchiMate-Elements *Business Process*. Quelle: Archi-Screenshot

Ein *Business Actor* (Abbildung 3.8, vgl. [Gro13, Kapitel 3.2.1]) ist eine Verhalten ausübende Entität der Organisation. Das Element repräsentiert somit einen aktiven Aspekt („active structure element“).

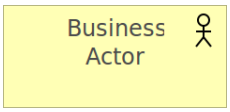


Abbildung 3.8: Abbildung des ArchiMate-Elements *Business Actor*. Quelle: Archi-Screenshot

Eine *Business Role* (Abbildung 3.9, vgl. [Gro13, Kapitel 3.2.2]), ebenfalls ein aktives Element, repräsentiert die Verantwortung dafür, Verhalten auszuüben. Eine Business Role kann einem Business Actor zugewiesen werden.

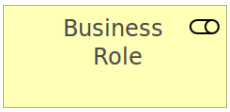


Abbildung 3.9: Abbildung des ArchiMate-Elements *Business Role*. Quelle: Archi-Screenshot

Einem Business Process sind Business Roles zugewiesen, diesen wiederum Business Actors.

Ein *Business Service* (Abbildung 3.10, vgl. [Gro13, Kapitel 3.3.5]) ist ein Verhaltens-Element, welches seiner Umgebung ein sinnvolle Funktionalität anbietet. Der Name so eines Elements sollte das Wort „Service“ oder, im Englischen, ein Verb endend mit „-ing“ enthalten.

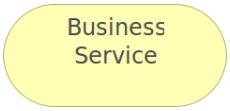


Abbildung 3.10: Abbildung des ArchiMate-Elements *Business Service*. Quelle: Archi-Screenshot

Application Layer

Ein *Application Component* (Abbildung 3.11, vgl. [Gro13, Kapitel 4.2.1]) ist ein aktives Element des Application Layers. Es repräsentiert eine in sich geschlossene Funktionseinheit, also etwa eine Software, ein Software-Modul oder ein ganzes Informationssystem. Allerdings stellt es nur den strukturellen Teil der Funktionseinheit dar, sein Verhalten wird ausschließlich über Verknüpfungen mit Verhaltens-Elementen modelliert. Der Name des Elements sollte ein Nomen sein.

Eine *Application Function* (Abbildung 3.12, vgl. [Gro13, Kapitel 4.3.1]) ist ein Verhaltens-Element, das automatisiertes Verhalten darstellt, welches von einem Application Component ausgeführt wird. Application Functions beschreiben das Verhalten von Application Components, ohne deren Implementierung zu berücksichtigen. Der Name des Elements sollte im Englischen ein Verb sein, das mit „-ing“ endet, wie zum Beispiel „billing“. Eine entsprechende Verbform existiert im

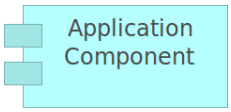


Abbildung 3.11: Abbildung des ArchiMate-Elements *Application Component*. Quelle: Archi-Screenshot

Abbildung 3.12: Abbildung des ArchiMate-Elements *Application Function*. Quelle: Archi-Screenshot

Deutschen nicht, weshalb in diesem Fall einfach ein Verb in aktiver Form wie etwa „in Rechnung stellen“ verwendet werden könnte.

Ein *Application Service* (Abbildung 3.13, vgl. [Gro13, Kapitel 4.3.3]) ist ein Verhaltens-Element, welches relevante Funktionalität eines Application Components an das Umfeld anbietet. Der Name so eines Elements sollte das Wort „Service“ oder, im Englischen, ein Verb endend mit „-ing“ enthalten.

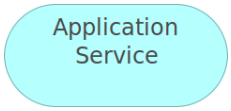


Abbildung 3.13: Abbildung des ArchiMate-Elements *Application Service*. Quelle: Archi-Screenshot

Das *Data Object* (Abbildung 3.14, vgl. [Gro13, Kapitel 4.4.1]) ist ein passives Element, auf welches automatisiertes Verhalten angewendet werden kann. Dieses Element kann automatisiert produziert oder benutzt werden und es sollte eine abgeschlossene, unabhängige Informationsmenge darstellen. Die repräsentierte Information sollte geschäftsrelevant sein. Der Name eines Data Objects sollte ein Nomen sein.

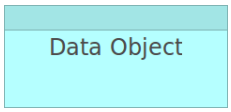


Abbildung 3.14: Abbildung des ArchiMate-Elements *Data Object*. Quelle: Archi-Screenshot

Technology Layer

Ein *Node* (Abbildung 3.15, vgl. [Gro13, Kapitel 5.2.1]) ist ein aktives Element und repräsentiert eine rechnerische Ressource, auf der Artifakte gespeichert oder zur Ausführung gebracht werden können. Üblicherweise werden Nodes verwendet, um Applikations-, Datenbank-Server oder Client Workstations zu modellieren. In der Regel besteht ein Node aus einem Hardware-Gerät und System-Software. Diese Bestandteile können explizit modelliert oder implizit gelassen werden. Der Name eines Nodes sollte ein Nomen sein.

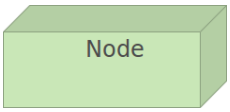


Abbildung 3.15: Abbildung des ArchiMate-Elements *Node*. Quelle: Archi-Screenshot

Ein *Infrastructure Service* (Abbildung 3.16, vgl. [Gro13, Kapitel 5.3.2]) ist ein Verhaltens-Element und repräsentiert für das Umfeld relevante Funktionalitäten von Nodes. Infrastructure Services werden von einem oder mehreren Nodes realisiert, und von Application Components oder anderen Nodes benutzt. Sie können Artifakte benötigen, benutzen und produzieren. Ihr Name sollte das Wort „Service“ beinhalten oder, im Englischen, ein Verb sein, das auf „-ing“ endet.

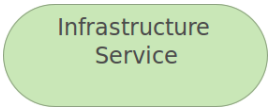


Abbildung 3.16: Abbildung des ArchiMate-Elements *Infrastructure Service*. Quelle: Archi-Screenshot

Ein *Artifact* (Abbildung 3.17, vgl. [Gro13, Kapitel 5.4.1]) ist ein passives Element und stellt eine physikalische Repräsentation an Datenobjekten in Form einer Datei dar. Der Name eines Artifacts sollte dem Dateinamen entsprechen.

Abbildung 3.17: Abbildung des ArchiMate-Elements *Artifact*. Quelle: Archi-Screenshot

Relationships

Relationships repräsentieren die Beziehungen der Elemente miteinander. Es gibt drei Typen von Beziehungen [Gro13, Kapitel 7]:

- *Structural Relationships* (strukturelle Beziehungen) stellen den strukturellen Zusammenhang von Konzepten gleichen oder unterschiedlichen Typs.
- *Dynamic Relationships* (dynamische Beziehungen) stellen Abhängigkeiten zwischen Verhaltenskonzepten dar, wie zum Beispiel eine zeitliche Abfolge.
- *Other Relationships* (andere Beziehungen) umfassen alle Beziehungen, die nicht in die ersten beiden Kategorien passen.

Eine weitere Besonderheit der ArchiMate-Relationships ist es, dass sie auf allen drei Layern, also Business, Application und Infrastructure mit der gleichen semantischen Bedeutung vorkommen.

Die *Composition Relationship* (Abbildung 3.18, vgl. [Gro13, Kapitel 7.1.1]) ist eine strukturelle Beziehung und stellt dar, dass ein Objekt aus einem oder mehreren anderen Objekten besteht. In dem Beispiel der Abbildung besteht Application Component 3 aus Application Component 1 und Application Component 2, erkennbar an der Position der gefüllten Raute.

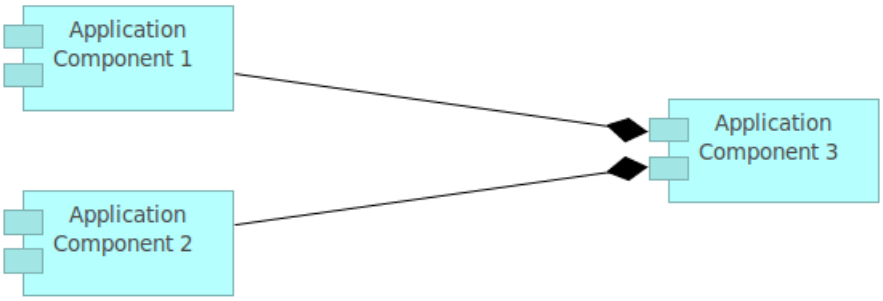


Abbildung 3.18: Abbildung des ArchiMate-Elements *Composition Relationship* als Verbindung dreier Application Components. Quelle: Archi-Screenshot

Die *Assignment Relationship* (Abbildung 3.19, vgl. [Gro13, Kapitel 7.1.3]) ist eine strukturelle Beziehung und verbindet ein „Active Structure Element“ (aktives Element) mit einem „Behavior Element“ (Verhalten), das von ihm ausgeführt wird. In dem Beispiel in der Abbildung führt Application Component 1 das Verhalten Application Function 1 aus.

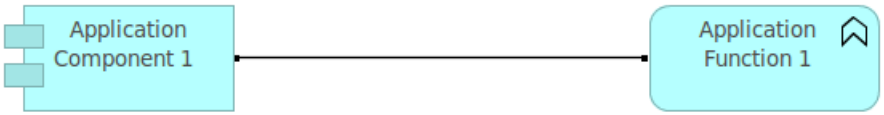


Abbildung 3.19: Abbildung des ArchiMate-Elements *Assignment Relationship* als Verbindung eines Application Components mit einer Application Function. Quelle: Archi-Screenshot

Die *Realization Relationship* (Abbildung 3.20, vgl. [Gro13, Kapitel 7.1.4]) ist eine strukturelle Beziehung und verbindet eine logische Einheit „(Was?)“ mit einer konkreteren Einheit „(Wie?)“, durch die sie realisiert wird. Application Component 1 realisiert das logische Element Application Service 1.

Abbildung 3.20: Abbildung des ArchiMate-Elements *Realization Relationship* als Verbindung eines Application Components mit einem Application Service. Quelle: Archi-Screenshot

Die *Used By Relationship* (Abbildung 3.21, vgl. [Gro13, Kapitel 7.1.5]) ist eine strukturelle Beziehung zur Darstellung der Nutzung eines Services beispielsweise durch Prozesse oder Funktionen. Im dargestellten Beispiel wird das Application Service 1 von Application Component 1 benutzt.

Die *Access Relationship* (Abbildung 3.22, vgl. [Gro13, Kapitel 7.1.6]) ist eine strukturelle Beziehung, die den Zugriff von Verhaltenselementen auf Business- und Data Objects darstellt. Auf die Objekte kann auf unterschiedliche Arten zugegriffen werden, wobei die Art des Zugriffs wird durch die Positionen der Pfeilspitzen symbolisiert wird. Folgende Zugriffsarten sind definiert (In Klammer stehen der Name des Data Objects, das in der Abbildung die jeweilige Zugriffsart aufweist):

1. „Write“: Objekt wird hinzugefügt, verändert oder gelöscht (Data Object 1)

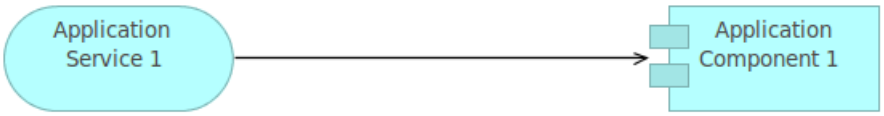


Abbildung 3.21: Abbildung des ArchiMate-Elements *Used By Relationship* als Verbindung eines Application Services mit einem Application Component. Quelle: Archi-Screenshot

2. „Read“: Daten des Objekts werden gelesen (Data Object 2)

3. „Read/Write“: Repräsentiert Lese- und Schreibzugriff auf das Datum. (Data Object 3)

4. „Access“: Das Datum ist auf irgendeine Weise mit dem Verhaltenselement assoziiert. Zum Beispiel stellt es die Information dar, die bei einem bestimmten Event generiert wird. (Data Object 4)

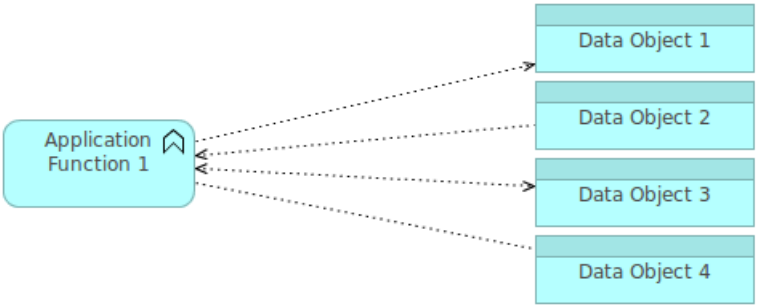


Abbildung 3.22: Abbildung des ArchiMate-Elements *Access Relationship* in vier verschiedenen Ausführungen als Verbindung einer Application Function mit je einem Data Object. Quelle: Archi-Screenshot

Die *Association Relationship* (Abbildung 3.23, vgl. [Gro13, Kapitel 7.1.7]) ist eine strukturelle Beziehung, und sie repräsentiert alle Beziehungen, die nicht durch eine spezifischere ArchiMate-Beziehung dargestellt werden. Die Abbildung 3.23 repräsentiert diese Zuordnung: Application Component 1 hat eine assoziative Beziehung zu Data Object 1.

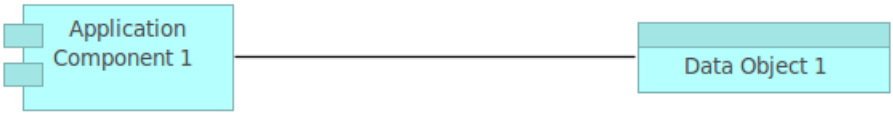


Abbildung 3.23: Abbildung des ArchiMate-Elements *Association Relationship* als Verbindung zwischen einem Application Component und einem Data Object. Quelle: Archi-Screenshot

Die *Group Relationship* (Abbildung 3.24, vgl. [Gro13, Kapitel 7.3.1]) zählt zu den „anderen Beziehungen“, da sie weder den strukturellen noch den dynamischen Beziehungen zuordenbar ist. Die Gruppenbeziehung umfasst Elemente, die aufgrund irgendeiner Charakteristik zusammengehören. Sie formen dabei kein übergeordnetes Objekt, wie es beispielsweise bei der Composition Relationship der Fall ist. Es wird lediglich graphisch dargestellt, dass mehrere Elemente etwas gemeinsam haben. Ein Element kann mehreren Gruppen zugeordnet sein.

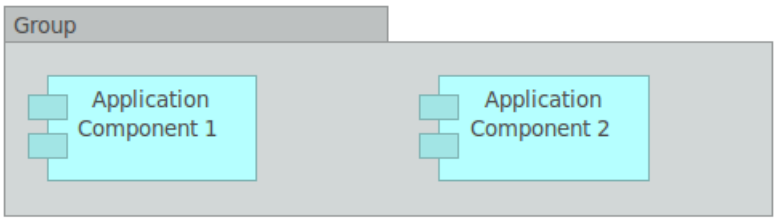


Abbildung 3.24: Abbildung des ArchiMate-Elements *Group Relationship* als Gruppierung zweier Application Components. Quelle: Archi-Screenshot

Dies sind alle Elemente, die für das Modell der Fallstudie verwendet werden. Im folgenden Abschnitt wird mit Hilfe dieser Elemente ein ArchiMate-Schema aus den Aussagen von Tabelle 3.2 entwickelt.

3.3.2 Das Schema in ArchiMate

Im Folgenden werden die identifizierten Modell-Aussagen aus Tabelle 3.2 mit ArchiMate-Elementen abgebildet. Im Anschluss werden die auf diese Weise definierten Elemente und Beziehungen zu einem Schema des Modells zusammengefasst.

Die erste Aussage lautet:
1. Das Modell repräsentiert die Applikation mit Namen X.
Applikationen werden in ArchiMate durch ein *Application Component* repräsentiert, wie im vorigen Kapitel beschrieben wurde. Der Name ist eine Standardeigenschaft von Application Components bei ArchiMate. Die erste Aussage ist in Abbildung 3.25 dargestellt.

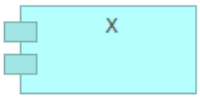


Abbildung 3.25: ArchiMate-Darstellung der Aussage „Das Modell beinhaltet die Applikation mit Namen X.“

2. Die Applikation X hat die Synonyme Y und Z. (optional)
Synonyme sind weitere Eigenschaften von Applikationen. Da sie keine Standard-Eigenschaft in ArchiMate sind, werden sie in das *Profil von Application Components* übernommen, wie in Abbildung 3.26 dargestellt ist.

Name	Value
Synonyme	Y, Z

Abbildung 3.26: Aufnahme der Aussage „Die Applikation X hat die Synonyme Y und Z. (optional)“ in das Profil von Application Components.

3. Die Applikation X ist eingebettet in die (Host-)Applikation Y.
Die Beziehung einer Einbettung wird über eine *Composition Relationship* zwischen den Application Components X und Y dargestellt (siehe Abbildung 3.27).

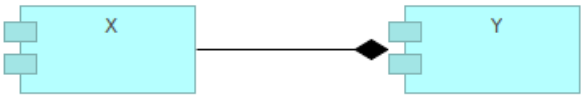


Abbildung 3.27: ArchiMate-Darstellung der Aussage „Die Applikation X ist eingebettet in die (Host-)Applikation Y.“

4. Die Applikation X läuft auf den Servern Y und Z.
Ein Server wird als *Node* dargestellt, welcher ein *Infrastructure Service* realisiert (*Realization Relationship*). Das Service beschreibt die Art des Servers, also zum Beispiel, ob es sich um einen Datenbank-Server, Web-Server oder eine andere Art von Server handelt. Ein Server kann mehrere Services anbieten. Dieses Service ist aus Mangel an erlaubten spezifischeren Beziehungsarten via *Association Relationship* mit dem Application Component verbunden. Used By Relationships zwischen Application Components und Services würden sich anbieten, werden allerdings zur Darstellung von Schnittstellen verwendet, wie weiter unten zu lesen ist, daher ist hier eine andere Beziehungsart zu verwenden, was nur noch die Association Relationship übrig lässt. Abbildung 3.28 zeigt die ArchiMate-Darstellung der vierten Aussage.

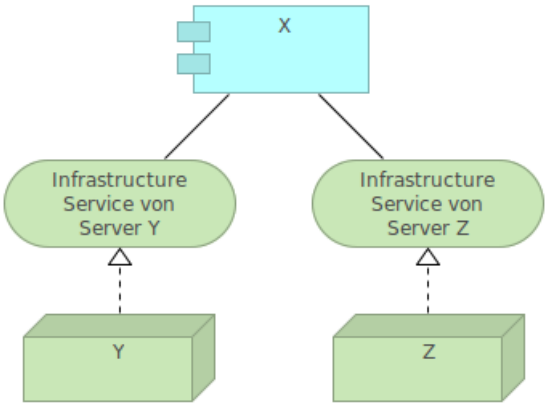


Abbildung 3.28: ArchiMate-Darstellung der Aussage „Die Applikation X läuft auf den Servern Y und Z.“

5. Die Applikation X hat die Standardisierungsklasse Y.
Die Standardisierungsklasse ist eine weitere Eigenschaft von Applikationen und wird in das *Profil* von Application Components übernommen. Details zu diesen Klassen sind Tabelle 3.1 zu entnehmen. Das um diese Aussage erweiterte Profil ist Abbildung 3.29 zu entnehmen.

Name	Value
Synonyme	
Standardisierungsklasse	Y

Abbildung 3.29: Aufnahme der Aussage „Die Applikation X hat die Standardisierungsklasse Y.“ in das Profil von Application Components.

6. Die Applikation X hat Person/Abteilung/Team Y als technische Betreuung.
Für diese Aussage werden drei unterschiedliche Modellierungsweisen in Erwägung gezogen:

1. Die technische Betreuung ist eine Eigenschaft, die in das *Profil von Application Components* übernommen wird.

2. Die Verantwortlichkeit einer technischen Betreuung eines Application Components wird als *Business Role* modelliert, welche von einem *Business Actor* ausgeführt und auf das *Application Component* angewendet wird.

3. Die technische Betreuung wird als *Business Service* modelliert, welches von einem *Business Actor* ausgeführt und mit einem oder mehreren *Application Components* verbunden ist. Dem ausführenden Business Actor ist eine generische Verantwortlichkeit zur technischen Betreuung einer Applikation in Form einer *Business Role* zugewiesen.

Die erste Methode würde darstellen, dass es eine Eigenschaft der Applikation ist, wer sie betreut. Das ist aber nicht der Fall. Eine Applikation kann eine neue technische Betreuung zugewiesen bekommen und sie selbst bleibt dabei völlig unverändert. Die Veränderung geschieht im *Business Layer* und sollte auch dort modelliert werden. Diese Methode hätte auch noch einen praktischen Nachteil. Wenn man das Modell danach durchsucht, welche Applikationen eine bestimmte Person technisch betreut, kann man auf unterschiedliche Schreibweisen ihres Namens in den Profilen treffen, da dort nur ein freier Text steht.

Die Erkenntnisse aus diesen Überlegungen sind erstens, dass die Betreuung im Business Layer modelliert werden sollte und zweitens, dass die betreuende Entität ein eigenes Element und kein Profil-Eintrag sein sollte. Die beiden weiteren Modellierungsweisen berücksichtigen diese Erkenntnisse.

Die zweite Methode stellt die Verantwortung der technischen Betreuung als Business Role dar, welche einem Akteur sowie einer Applikation zugewiesen wird. Die technische Betreuung jeder einzelnen Applikation ist auf diese Weise eine eigene Business Role.

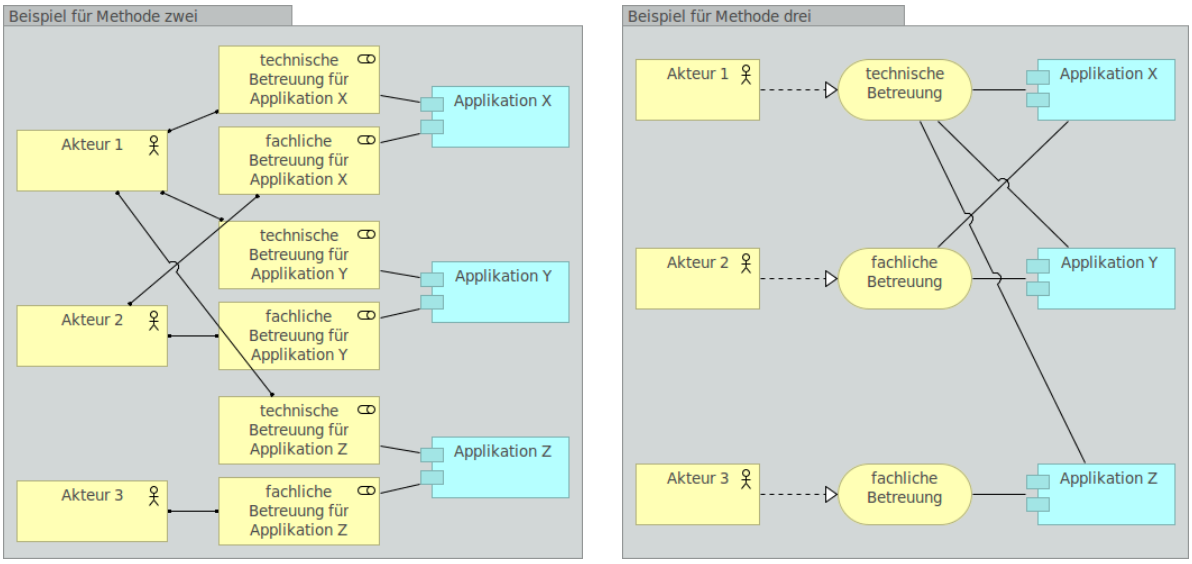
Die dritte Methode modelliert die technische Betreuung als Business Service. Ein Akteur bietet dieses Service maximal einmal an, in Gebrauch nehmen können es mehrere Applikationen.

Zur Illustration dieser beiden Methoden wird ein Beispiel herangezogen: Akteur 1 betreut die Applikationen X, Y und Z technisch. Akteur 2 betreut Applikationen X und Y fachlich. Akteur 3 betreut die Applikation Z fachlich. Diese Gegebenheiten sind nach den Methoden zwei und drei modelliert in Abbildung 3.30 dargestellt.

Sowohl das Herausstreichen der Verantwortlichkeit der Betreuung in Form einer Rolle (Methode zwei, Abbildung 3.30 (a)) als auch der Betreuungs-Leistung in Form eines Services (Methode drei, Abbildung 3.30 (b)) sind geeignet. Das Element Business Role wird weiter unten im Zusammenhang mit „Tätigkeiten“ eingeführt. Wie bereits erklärt wurde, sind so wenig verschiedene Elemente wie möglich zu verwenden, um das Modell übersichtlich zu halten. Aus diesem Grund wird in diesem Modell die zweite Methode (Abbildung 3.30 (a)) gewählt. Aussage sechs wird also gemäß Abbildung 3.31 dargestellt.

7. Die Applikation X hat Person/Abteilung/Team Z als fachliche Betreuung.
Diese Aussage wird analog zur vorherigen modelliert (siehe Abbildung 3.32).

8a. Das Datenobjekt D hat Applikation X als Masterapplikation.
8b. Das Datenobjekt D hat Datei X als Masterdatei.
Liegt das Datenobjekt in Form einer Datei vor, so *realisiert* die Datei das Datenobjekt. Es liegt



(a) Methode zwei (b) Methode drei

Abbildung 3.30: Modellierung eines Applikations-Betreuungs-Beispiels mit zwei unterschiedlichen Methoden. Quelle: Archi-Screenshot

Abbildung 3.31: ArchiMate-Darstellung der Aussage „Die Applikation X hat Person/Abteilung/Team Z als technische Betreuung.“

Abbildung 3.32: ArchiMate-Darstellung der Aussage „Die Applikation X hat Person/Abteilung/Team Z als fachliche Betreuung.“

(a) „Das Datenobjekt D hat Datei X als Masterdatei.“ (b) „Das Datenobjekt D hat Applikation X als Masterapplikation.“

Abbildung 3.33: ArchiMate-Darstellungen der zwei Aussagen über die Zugehörigkeit von Datenobjekten.

also eine *Realisation Relationship* vor, wie in Abbildung 3.33 (a) zu sehen ist. Gehört das Datenob-

jekt zu einer Masterapplikation, so wird diese Zusammengehörigkeit als *Association Relationship* modelliert, wie auf der rechten Seite der Abbildung zu sehen ist.

9. Über die Schnittstelle S wird das Datenobjekt D von Applikation X zu Applikation Y übertragen. Schnittstellen können sehr unterschiedlich realisiert sein. Im Zuge der Datenaufnahme von Kapitel 3.1.3.2 werden unter anderem folgende Schnittstellen identifiziert:

1. Web Services, die von einer Applikation angeboten werden,
2. direkter SQL-Zugriff auf die Datenbank einer anderen Applikation,
3. standardisierte Schnittstellen, wie zum Beispiel LDAP (Lightweight Directory Access Protocol [Zeine]),
4. proprietäre Schnittstellen, verwendet von proprietärer Software, über deren Interna nichts bekannt ist und
5. direkter Lese-/Schreib-Zugriff auf die Datei, die ein Datenobjekt realisiert.

Um zu sehen, wie diese unterschiedlichen Schnittstellen abgebildet werden können, wird von einem generischen Fall ausgegangen: Applikation X ist die Masterapplikation des Datenobjekts D und Applikation Y greift darauf zu. Dieser Fall wird in Abbildung 3.34 dargestellt. Die assoziative

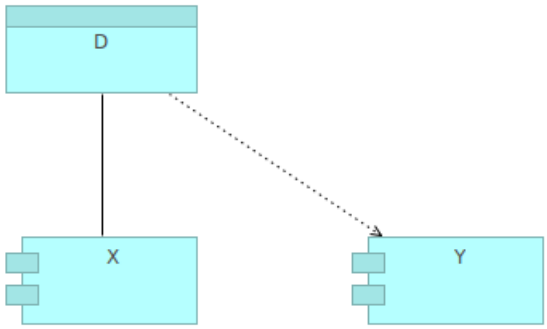


Abbildung 3.34: ArchiMate-Darstellung davon, dass das Datenobjekt D von Applikation X zu Applikation Y übertragen wird.

Beziehung von Applikation X und dem Datenobjekt D stellt dar, dass X die Masterapplikation von D ist und die Access-Relationship von Applikation Y zu D stellt den lesenden Zugriff von Y auf D dar. Bei dieser Darstellung ist somit nicht bekannt, auf welche Weise die Weitergabe der Daten erfolgt. Die Tatsache, dass die Applikation X auf irgendeine Weise Daten bereitstellt, kann über ein *Application Service* dargestellt werden. Der Name und die Beschreibung des Services können alle weiteren notwendigen Informationen bereitstellen. Wenn zum Beispiel das Application Service den Namen „RESTful Web Service http://intranet/resources/employee“ hat, ist es klar unterscheidbar von einem Application Service „SQL Read Access to database CRM“, ohne das Modell semantisch zu überladen. Auf diese Weise können die ersten vier Fälle semantisch gleich abgebildet werden, siehe Abbildung 3.35. Die Applikation X realisiert das Application Service S (*Realization Relation*), welches von Applikation Y verwendet wird (*Used By Relation*). Dabei wird das Datenobjekt D, welches X als Masterapplikation hat, sowohl von S als auch von Y gelesen. (*Access Relation*).

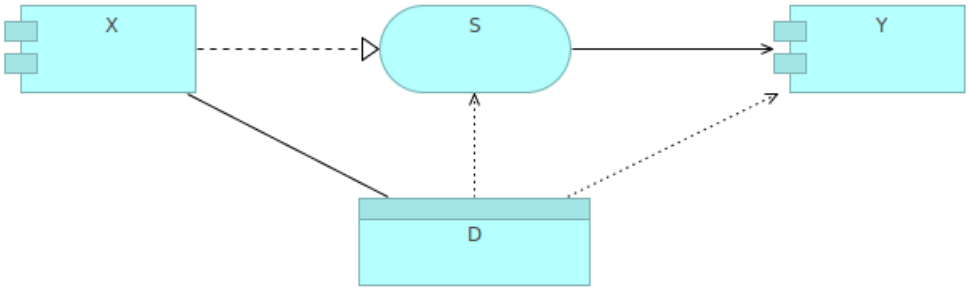


Abbildung 3.35: ArchiMate-Darstellung der Aussage „Über die Schnittstelle S wird das Datenobjekt D von Applikation X zu Applikation Y übertragen.“

Einzig der letzte Fall ist eine Ausnahme, weil keine Applikation sondern eine Datei die Daten bereitstellt. Lese-/Schreib-Zugriff auf die Datei wird analog zu Abbildung 3.35 als eine entsprechende Access-Relationship zu dem Datenobjekt dargestellt. Dies ist in Abbildung 3.36 zu sehen.

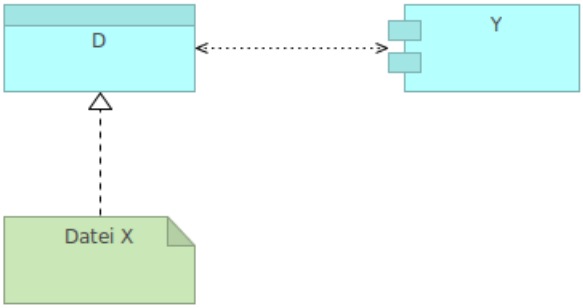


Abbildung 3.36: ArchiMate-Darstellung eines Lese- und Schreibzugriffs auf eine Datei.

10. Die Schnittstelle S wird von Trigger T gestartet. T ist entweder vom Typ „User“ oder vom Typ „Schedule“, geht von Applikation Z aus und hat die Bedingung Y.
Ein Trigger geht immer einher mit einer Funktionalität einer Applikation. Ein mögliches Beispiel dafür ist, wenn die Applikation Y einen Datensatz öffnet, welcher auf Informationen in dem Datenobjekt D der Applikation X verweist und somit auf dieses ebenfalls zugreift. Die Funktionalität „Öffnen“ der Applikation Y ist in diesem Beispiel der Trigger für die Schnittstelle S. In ArchiMate werden die Trigger daher als *Application Functions* dargestellt. Die nähere Beschreibung des Triggers (Typ, triggernde Applikation und Bedingung) kann im *Profil von Application Functions* festgehalten werden, wie in Abbildung 3.37 zu sehen ist. Die Beziehung des Triggers zu den anderen Elementen ist in Abbildung 3.38 dargestellt. Die Access Relation des Datenobjekts zeigt in der Abbildung nicht mehr zum Application Component selbst, sondern zur Application Function. Wenn eine Applikationen mehrere Schnittstellen hat, erhöht das die Übersichtlichkeit, über welche Schnittstelle welche Datenobjekte übertragen werden.

11. Die Niederlassung X verwendet die Applikation Y. Diese Aussage sollte eigentlich über die Geschäftsebene abgebildet werden. Geschäftstätigkeiten werden in bestimmten Niederlassungen unter Verwendung von bestimmten Applikationen ausgeführt. Über diese Beziehung (Niederlassung – Geschäftstätigkeit – Applikation) sollte die Aussage elf hergeleitet werden. Allerdings ist es im Fall des Partnerunternehmens so, dass nur in den größten Niederlassungen Geschäftsprozesse definiert sind und die Stakeholder dieser Arbeit nicht in der Position sind, dies zu ändern. Aus die-

Name	Value
Trigger	Y
Triggernde Applikation	Z
Trigger-Typ	entweder User oder Schedule

Abbildung 3.37: ArchiMate-Darstellung der Aussage „Die Schnittstelle S wird von Trigger T gestartet. T ist entweder vom Typ ‚User' oder vom Typ ‚Schedule', geht von Applikation Z aus und hat die Bedingung Y." in Form von Profil-Einträgen bei Application Functions.

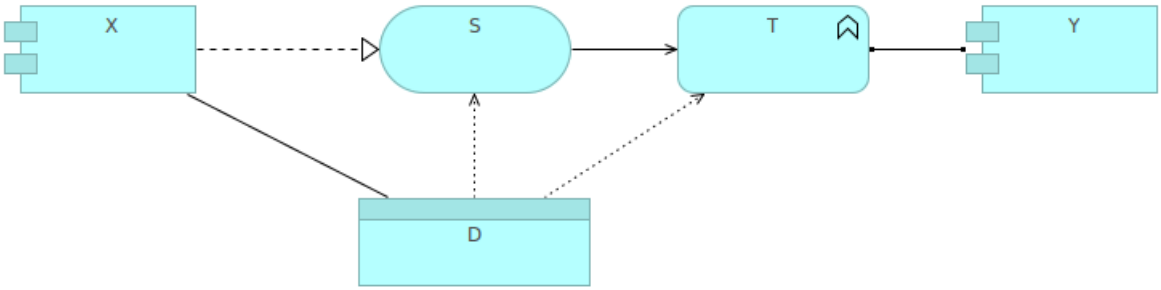


Abbildung 3.38: ArchiMate-Darstellungen einer vollständigen Schnittstelle inklusive Trigger.

sem Grund wird die Niederlassung einfach in das *Profil der Application Components* eingetragen (siehe Abbildung 3.39.

Name	Value
Synonyme	
Standardisierungsklasse	
Niederlassung	X

Abbildung 3.39: Aufnahme der Aussage „Die Niederlassung X verwendet die Applikation Y." in das Profil von Application Component Y.

Wenn das EAM im Unternehmen angenommen wird und weitere Stakeholder einbezogen werden bei der Etablierung und Weiterentwicklung, kann dieses Thema erneut diskutiert werden. Sobald die Geschäftsprozesse der Niederlassungen ebenfalls definiert und erfasst sind, sollte statt dieser Modellierung die ursprünglich angedachte Modellierungsweise verwendet werden, welche besagt, dass eine Geschäftstätigkeit in einer Niederlassung durchgeführt wird.

12. Die Tätigkeit T ist Teil des Geschäftsprozesses P.
Ein Geschäftsprozess gruppiert Tätigkeiten und Tätigkeiten werden von Rollen durchgeführt (vgl. Aussage 14 weiter unten).

Das ArchiMate-Element *Business Process* wird sowohl zum Darstellen von Geschäftsprozessen als auch von Tätigkeiten verwendet. Die Unterscheidung von Geschäftsprozess und Tätigkeit ist letzten Endes keine eindeutige. Eine Tätigkeit kann immer weiter in eine Abfolge noch detaillierter beschriebener Tätigkeiten aufgeteilt werden. Dann wäre die ursprüngliche Tätigkeit ein Geschäftsprozess, der die neuen, detaillierten Tätigkeiten zusammenfasst. Ein Beispiel dafür wäre die Tätigkeit *Rechnung bezahlen*. Diese Tätigkeit könnte detaillierter beschrieben werden mit *Rechnung annehmen*, *Rechnung archivieren* und *Geld bezahlen*.

Wenn Tätigkeit und Geschäftsprozess aus der Aussage zwölf durch das gleiche Element dargestellt werden, geht keine Information verloren. Denn Tätigkeiten werden stets von einer Rolle mithilfe einer Applikation ausgeführt (siehe Aussage 14) und sie werden zu Geschäftsprozessen (im Sinne von Aussage zwölf) zusammengefasst.

Die Beziehung „ist Teil von“ entspricht einer *Composition Relation*.

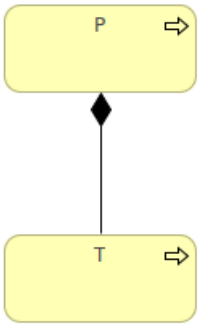


Abbildung 3.40: ArchiMate-Darstellung der Aussage „Die Tätigkeit T ist Teil des Geschäftsprozesses P.“

13. Der Geschäftsprozess P ist untergliedert in Sub-Geschäftsprozesse SP1 und SP2.
Diese Aussage wird analog zu Aussage drei über eine *Composition Relation* dargestellt und ist in Abbildung 3.41 zu sehen.

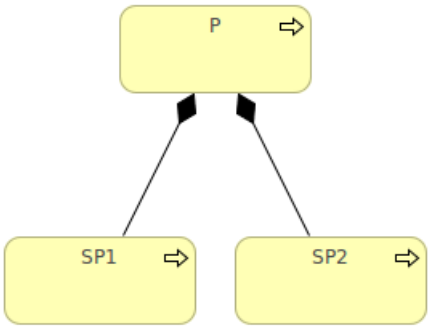


Abbildung 3.41: ArchiMate-Darstellung der Aussage „Der Geschäftsprozess P kann in Sub-Geschäftsprozesse SP1 und SP2 eingeteilt werden.“

14. Die Tätigkeit T wird ausgeführt von den Rollen A, B und C.
Tätigkeiten entsprechen *Business Processes* und diese werden von *Business Roles* ausgeführt. (vgl. voriges Kapitel). Diese Aussage kann daher wie in Abbildung 3.42 dargestellt modelliert werden: Die Tätigkeit T ist den Rollen D–F über eine *Assignment Relation* zugewiesen. Diese Rollen repräsentieren jeweils eine Verantwortung dafür, die Tätigkeit auszuführen.

15. Die Tätigkeit T wird ausgeführt unter Verwendung der Applikationen X, Y und Z.
Die Tatsache, dass eine Applikation für eine Tätigkeit verwendet wird, wird über eine *Used By Relation* dargestellt, wie in Abbildung 3.43 zu sehen ist.

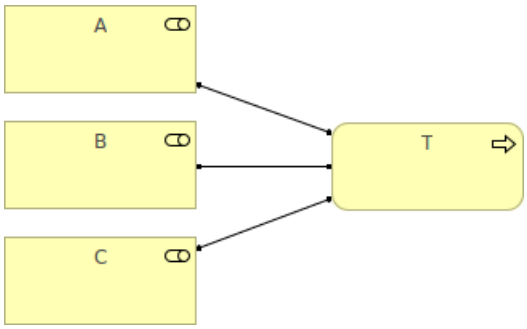


Abbildung 3.42: ArchiMate-Darstellung der Aussage „Die Tätigkeit T wird ausgeführt von den Rollen A, B und C.“

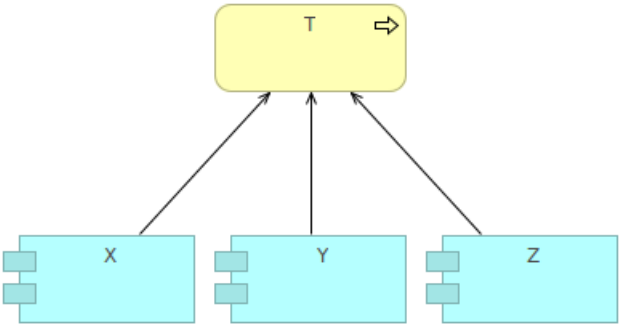


Abbildung 3.43: ArchiMate-Darstellung der Aussage „Die Tätigkeit T wird ausgeführt unter Verwendung der Applikationen X, Y und Z.“

Das Schema

Nun sind alle verwendeten Beziehungen und Elemente, sowie deren Bedeutung bekannt. In Abbildung 3.44 sind sie zusammengefasst. Zum vollständigen Schema des Modells zählen außerdem noch das Profil von Application Components (siehe Abbildung 3.39) und das Profil von Application Functions (siehe Abbildung 3.37).

Die in Kapitel 3.1.3.2 erhobenen Daten wurden entsprechend dem hier präsentierten Schema in ArchiMate umgesetzt. Das so erstellte Modell ist im Besitz des Partnerunternehmens. Im folgenden Kapitel wird ein kleiner Teil davon exemplarisch betrachtet und erklärt.

3.3.3 Die Projektmanagement-Sicht als beispielhafter Auszug aus dem Modell

Im Folgenden wird ein Auszug aus dem erstellten Modell rund um den Projektmanagement-Geschäftsprozess betrachtet. Zu beachten ist dabei, dass das Modell im Gegensatz zu dieser Arbeit auf Englisch verfasst ist. Dieses Kapitel betrachtet nur einen sehr eingegrenzten Bereich des Modells, um den LeserInnen dieser Arbeit einen Einblick in das Ergebnis zu verschaffen.

Business Layer

Wie bereits erwähnt existiert bereits eine Dokumentation der Geschäftsprozesse. Diese Dokumentation wurde herangezogen, um den Business Layer für das EAM abzubilden. Zusätzlich wird die Applikations-Betreuung ebenfalls im Business Layer dargestellt.

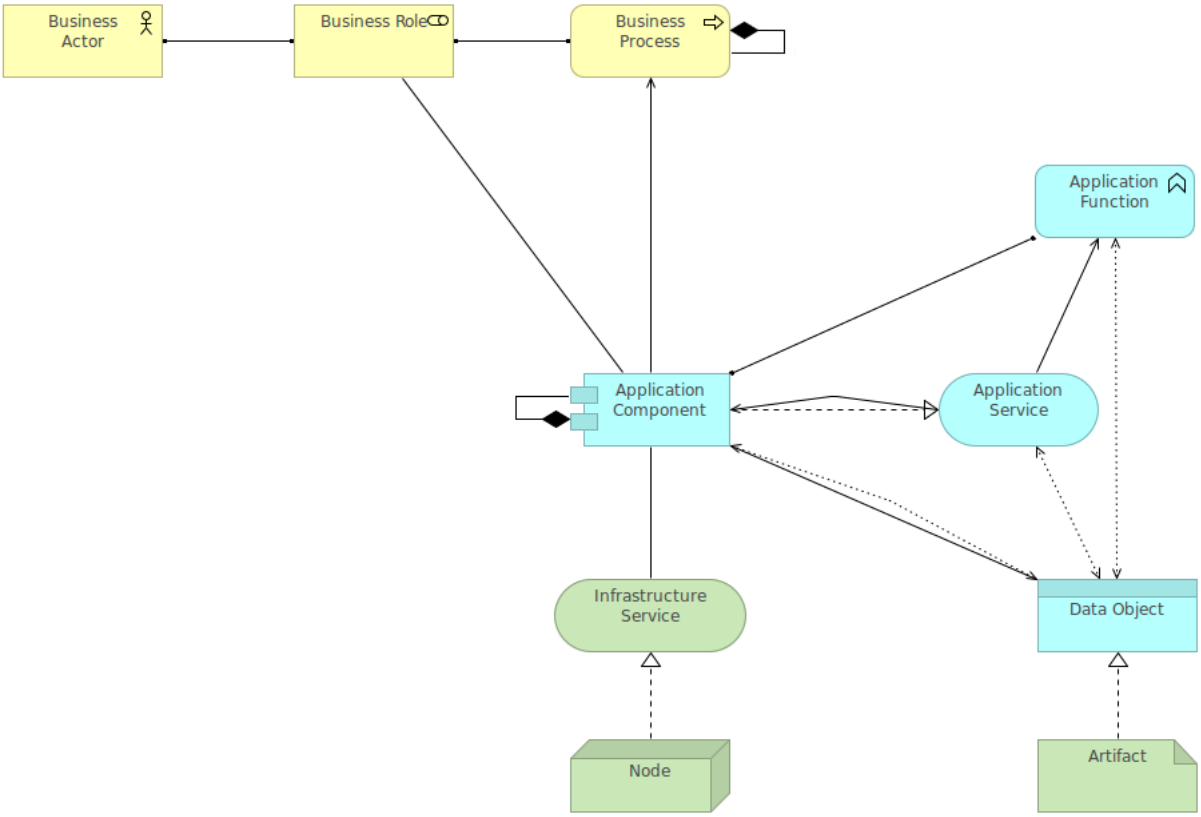


Abbildung 3.44: Das Schema des Modells abgebildet in ArchiMate.

Abbildung 3.45 stellt die Applikations-Betreuung dar. Die Namen der entsprechenden Personen wurden dafür abgeändert. Max Mustermann ist somit für die technische Betreuung der Applikation *PM Tool* verantwortlich („Technical Maintenance for PM Tool“) und Martin Mustermann für die fachliche Betreuung („Contentual Maintenance for PM Tool“). Für die Applikation *PM Tool USA Web* übernimmt Martina Mustermann die technische Betreuung und Markus Mustermann die fachliche.

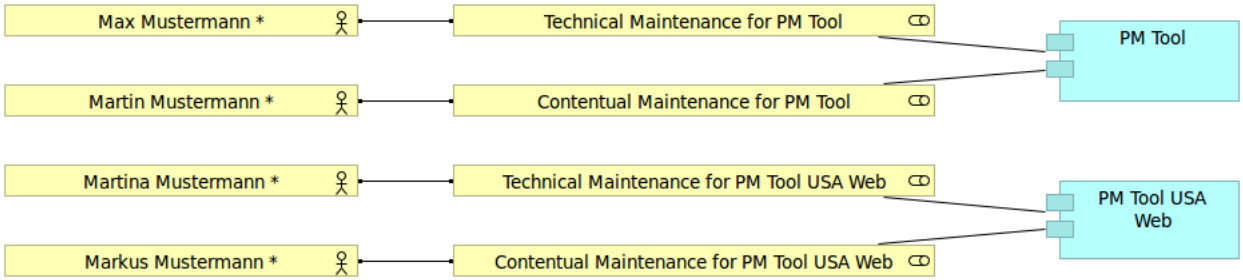


Abbildung 3.45: ArchiMate-Darstellung der Applikations-Betreuung (mit geänderten Personen-Namen).

Der Projektmanagement-Prozess hat vier Subprozesse definiert, von dem jeder in einer eigenen View präsentiert wird. Abbildung 3.46 ist die erste View auf den Geschäftsprozess und seine Subprozesse.

Man sieht die hierarchische Verknüpfung der Geschäftsprozesse. Der Geschäftsprozess *Project Management* ist unterteilt in *Project Classification*, *Order Handling*, *Setup Phase* und *Project Implementation*. Setup Phase wiederum ist weiter unterteilt in die Subprozesse *Project Definition* und *Project Planning*.

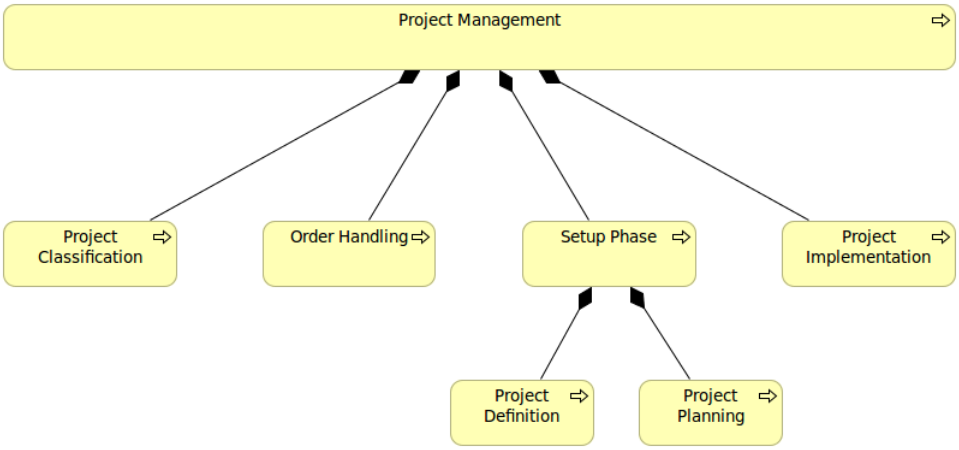


Abbildung 3.46: Übersicht über den Geschäftsprozess Project Management und seine Subprozesse.

Die nächste View (Abbildung 3.47) zeigt Details des Subprozesses Project Classification. Die einzige Tätigkeit, die diesem Prozess zugeordnet ist, lautet „Assessing projects according to their type and risk class“. Ausgeführt wird diese Tätigkeit von der Rolle *Sales Responsible*. Die Person hinter dieser Rolle ist nicht bekannt, weil in der Quell-Dokumentation nur Rollen und keine Personen definiert sind. Zum Ausführen der Tätigkeit wird außerdem die Applikation *CRM* verwendet, was durch die Used By Relation dargestellt ist.

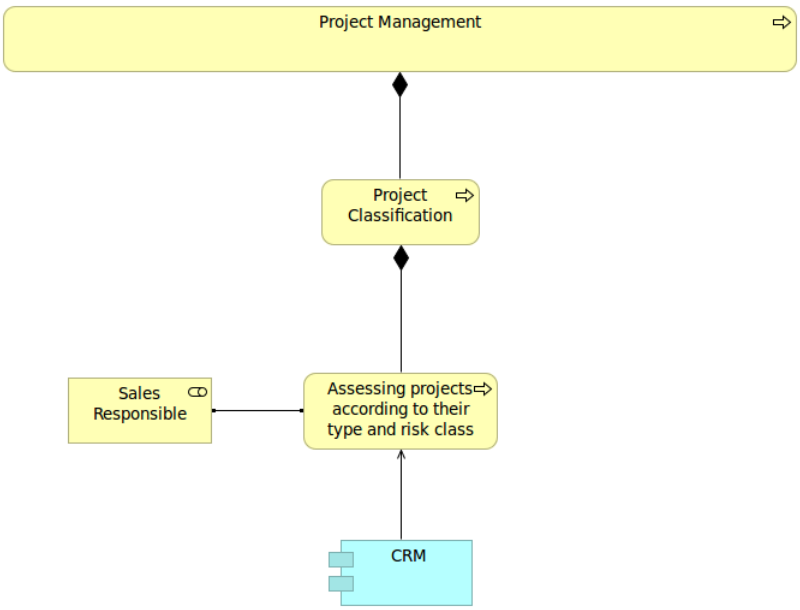


Abbildung 3.47: Detailsicht auf den Subprozess Project Classification und die ihm zugeordneten Tätigkeiten.

Abbildung 3.48 zeigt Details des Subprozesses Order Handling. Das Schema ist analog zum vorherigen Subprozess: Die Tätigkeit „Set up a PSP-Element“ wird durchgeführt von der Rolle *Order Responsible* unter Verwendung der Applikation *PM Tool*. „Set up a Payment Milestone in SAP“ wird von *Treasury* ausgeführt, unter Verwendung von *SAP SD*, einem Modul von *SAP* (Composition Relation). „Document incoming order in CRM“ wird ebenfalls von der Rolle *Order Responsible* unter Verwendung der Applikation *CRM* durchgeführt.

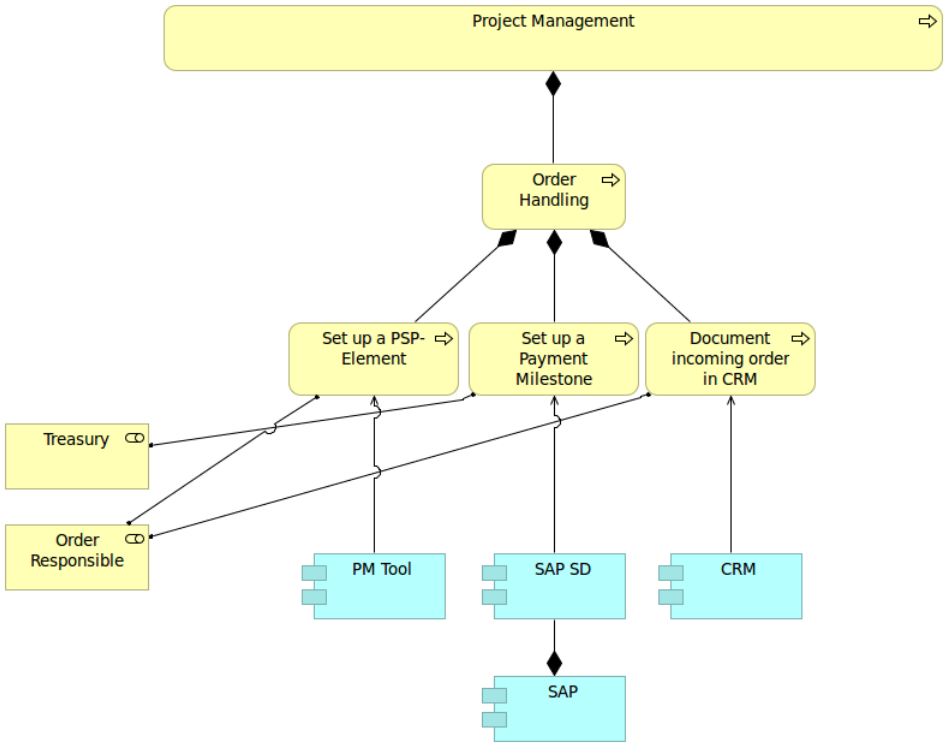


Abbildung 3.48: Detailsicht auf den Subprozess Order Handling und die ihm zugeordneten Tätigkeiten.

Die Abbildungen 3.49 bis 3.51 zeigen die übrigen Subprozesse, wobei Abbildung 3.51 sehr gut hervorhebt, warum es sinnvoll ist, jede Detail-Ansicht in einer eigenen View anzuzeigen. Die Menge an Elementen ist so groß, dass die View unübersichtlich ist. Eine unübersichtliche View erschwert klarerweise die Kommunikation über ihren Inhalt.

Im Business Layer sind die Aussagen sechs, sieben und zwölf bis 15 aus Tabelle 3.2 abgebildet.

Was im Business Layer nicht ersichtlich ist, sind kausale Zusammenhänge der Prozesse und Tätigkeiten, da es nicht Teil der Anforderungen war. Kausale Verknüpfungen wären in ArchiMate aber gut abbildbar und es wäre ein sinnvoller nächster Schritt, den Business Layer dahingehend auszubauen, mehr Informationen über die Geschäftsabläufe abzubilden.

Application Layer

Im Application Layer liegt der Hauptfokus dieses Modells. Als Einblick in diesen Layer wird die Applikation *PM Tool* gezeigt. Diese Applikation ist eine Eigenentwicklung des Unternehmens und wird für folgende Tätigkeiten des Project–Management–Geschäftsprozesses verwendet:

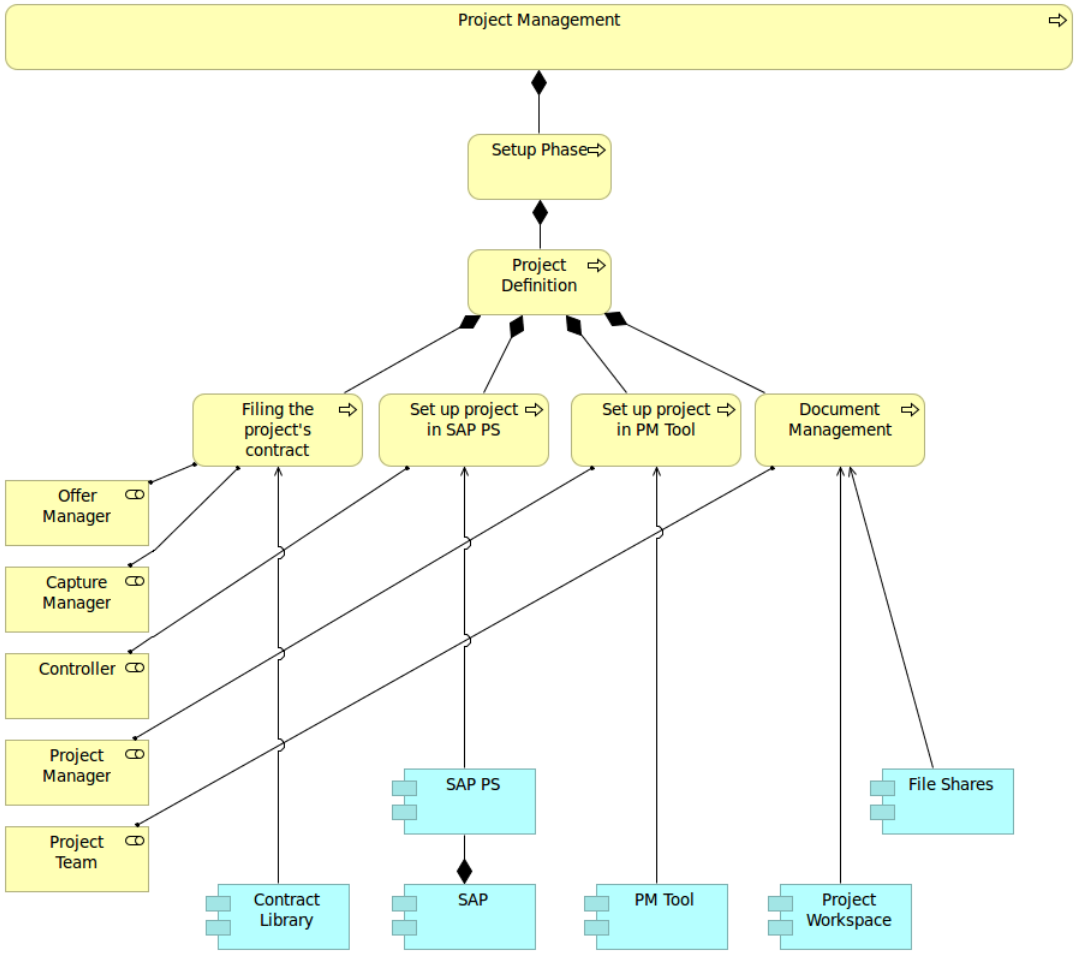


Abbildung 3.49: Detailsicht auf den Subprozess Project Definition und die ihm zugeordneten Tätigkeiten.

1. Set up a PSP–Element (siehe Abbildung 3.48)
2. Set up project in PM Tool (siehe Abbildung 3.49)
3. Budget creation (siehe Abbildung 3.50)
4. Creating and printing IPO-Document (siehe Abbildung 3.50)
5. Maintain Risks & Chances (siehe Abbildung 3.50)
6. Cost– and work–Tracking (siehe Abbildung 3.51)
7. Due Diligence (siehe Abbildung 3.51)
8. Release Quality Gates (siehe Abbildung 3.51)

In Abbildung 3.52 ist die Applikation PM Tool mit ihren Funktionen und Schnittstellen modelliert. Die beiden großen Application Components, links SAP und rechts PM Tool, deuten an, dass zwischen diesen beiden Systemen die meiste Interaktion stattfindet. Weitere Applikationen, die

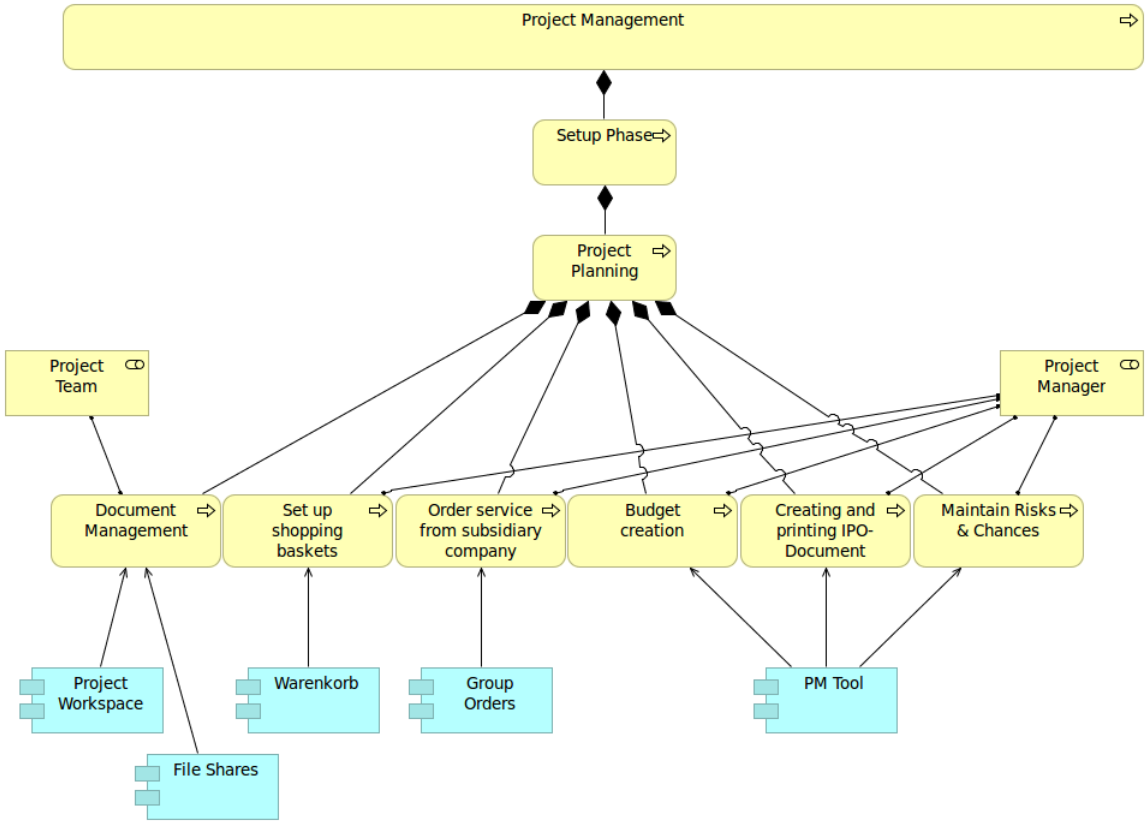


Abbildung 3.50: Detailsicht auf den Subprozess Project Planning und die ihm zugeordneten Tätigkeiten.

in Interaktion mit PM Tool stehen, sind Metadirectory, Lessons Learned, Project Workspace und PM Tool USA Web.

Das PM Tool hat die Standardisierungsklasse 5 (Eigenentwicklung). Es wird in den Niederlassungen Österreich und Deutschland verwendet. Ein vereinfachter Web Client, *PM Tool USA Web*, wird in der USA-Niederlassung verwendet. Diese Informationen sind den Profilen der jeweiligen Application Components zu entnehmen (hier nicht abgebildet).

Das PM Tool hat eine Funktionalität *Nightly Sync*, die die beiden Funktionen *Getting User Data* und *Updating Project Properties in Lessons Learned* startet. Diese Funktion wird nächtlich um 2 Uhr vom PM Tool getriggert. Diese Informationen sind im ebenfalls hier nicht abgebildeten Profil von Nightly Sync enthalten. Getting User Data liest das Datenobjekt *Person* (Access Relation, read). Dieses repräsentiert sämtliche IT-relevante Personen, welche von der Applikation *Identity Management* verwaltet werden. Zugriff auf das Datenobjekt erfolgt in diesem Fall über einen SQL-Lesezugriff (Used by Relation), den die Applikation Identity Management bereitstellt (Realization Relation). Updating Project Properties in Lessons Learned bezieht Informationen aus den Project Properties und benutzt das Web Service *LessonsLearnedService.svc*, welches das Datenobjekt *Learned Lesson* bearbeitet.

Zwei weitere Funktionen sind für die Synchronisation mit SAP verantwortlich. SAP bietet einige Web Services an, deren Namen in den Titeln der Application Services, die von SAP realisiert werden, zu finden sind. Die PM–Tool–Funktion *Sync with Forecast* benutzt die folgenden SAP–Services:

1. *Get Project Definition Web Service* wird verwendet um *Revenues*, also Erlöse, aus SAP zu übernehmen.
2. *Get Actuals Web Service* wird verwendet um *Actual Cost*, also Ist-Kosten, aus SAP zu übernehmen.
3. *Actuals Update Web Service* wird verwendet um *Project Forecast Cost* an SAP zu übermitteln.
4. *Project Definition Update Web Service* wird verwendet um *Project Properties* aus PM Tool zu übernehmen.

Die PM–Tool–Funktion *Sync without Forecast* benutzt die folgenden SAP–Services:

1. *Get Actuals Web Service* wird verwendet um *Actual Cost*, also Ist-Kosten, aus SAP zu übernehmen.
2. *Project Structure Update Web Service* wird verwendet um *Project Schedule and Structure* an SAP zu übermitteln.
3. *Project Definition Update Web Service* wird verwendet um *Project Properties* aus PM Tool zu übernehmen.

SAP ist in dieser Sicht nur soweit abgebildet, wie es für das PM Tool relevant ist. Weitere Informationen über die Applikation SAP ist in einer eigenen SAP-Sicht zu finden (hier nicht abgebildet).

Der Web Client für die USA-Niederlassung, PM Tool USA Web, hat nur eine Schnittstelle nach außen, *Get Non-USA projects*, welche über einen SQL-Zugriff *Project Properties* einliest.

Wenn das PM Tool ein Projekt abspeichert (*Saving Project*) wird das Datenobjekt *Project Team* an die Applikation *Project Workspace* gesendet, welches seine Informationen über das Projekt Team entsprechend erneuert (und somit seine Zugriffsrechte anpasst).

Ein weiteres Datenobjekt, *Company Forecast*, mit PM Tool als Masterapplikation wird von keiner anderen Applikation verwendet.

Im Application Layer sind die Aussagen eins bis drei, fünf und acht bis elf abgebildet.

Technology Layer

Ein Auszug aus dem Technology Layer für die Applikationen *PM Tool* und *PM Tool Web* ist in Abbildung 3.53 zu sehen. Die Applikation PM Tool benutzt den Datenbank-Server *VIESQL01*[3], welcher wiederum auf den beiden gespiegelten Servern *VIESQL011* und *VIESQL012* basiert. Die Applikation PM Tool USA Web verwendet zusätzlich noch einen Web-Server *VIEIIS01*.

Im Technology Layer ist die Aussage vier abgebildet.

[3] Die Servernamen wurden verändert.

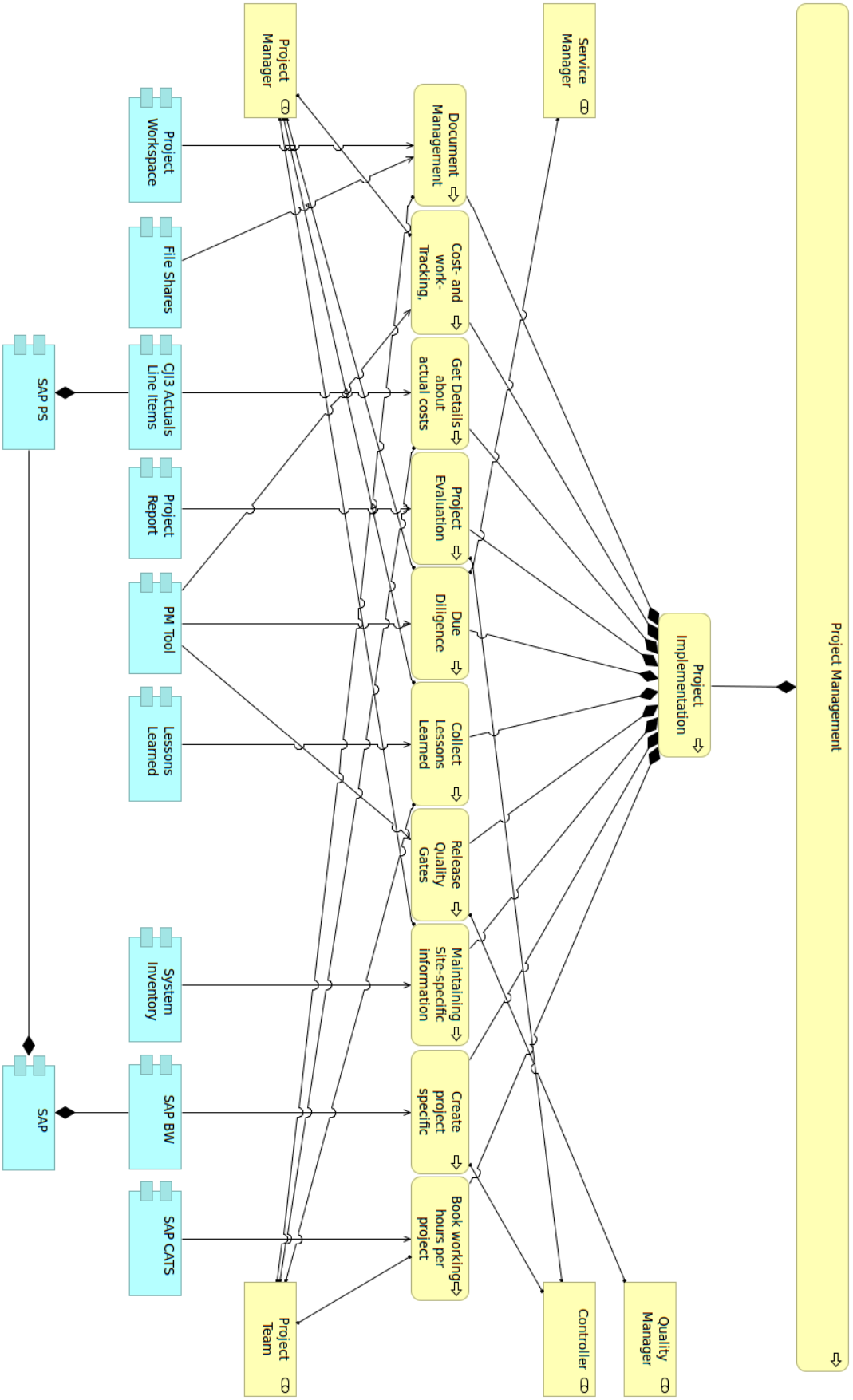


Abbildung 3.51: Detailsicht auf den Subprozess Project Implementation und die ihm zugeordneten Tätigkeiten.

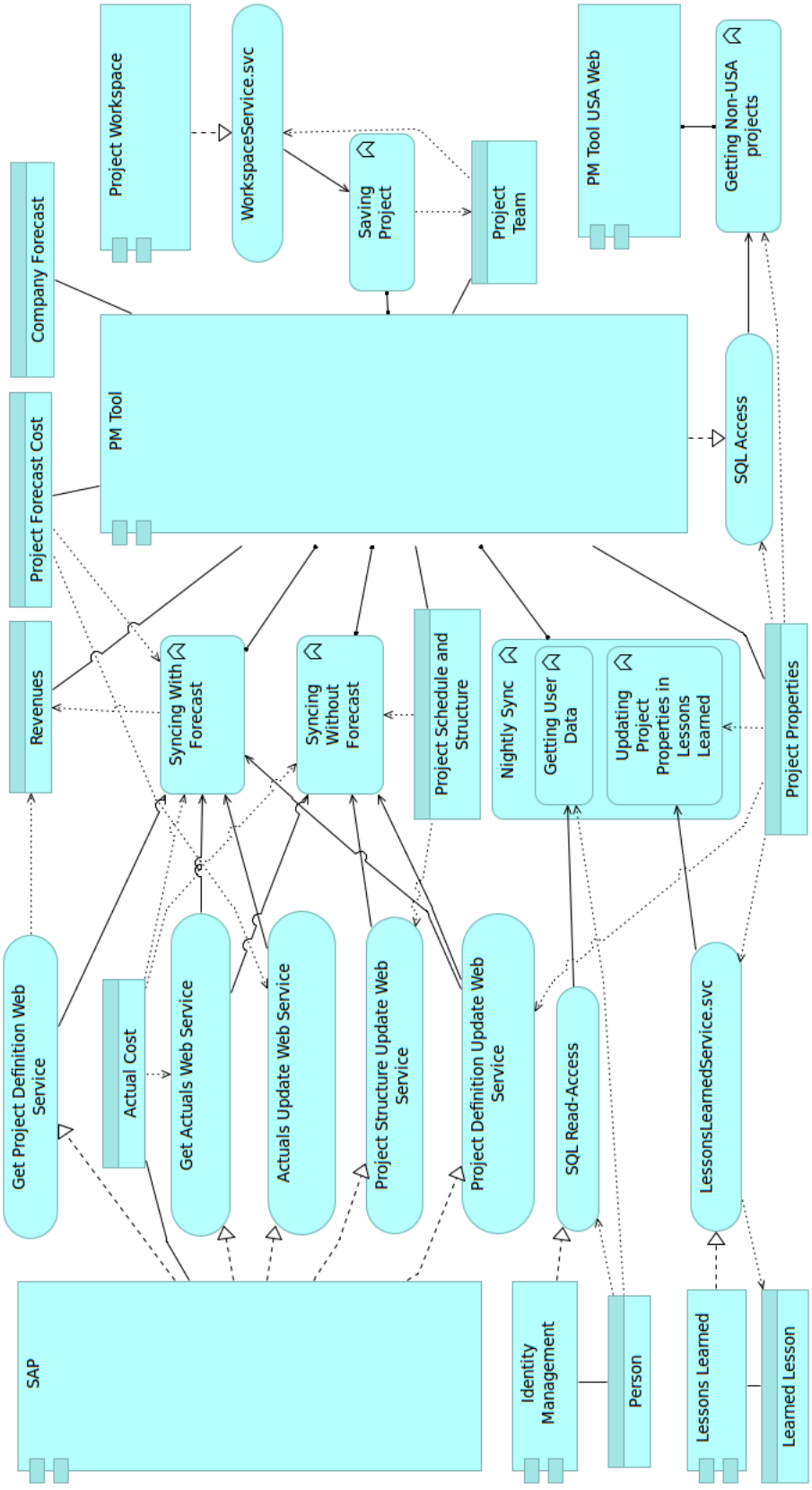


Abbildung 3.52: Die Applikation PM Tool mit ihren Funktionen und Schnittstellen.

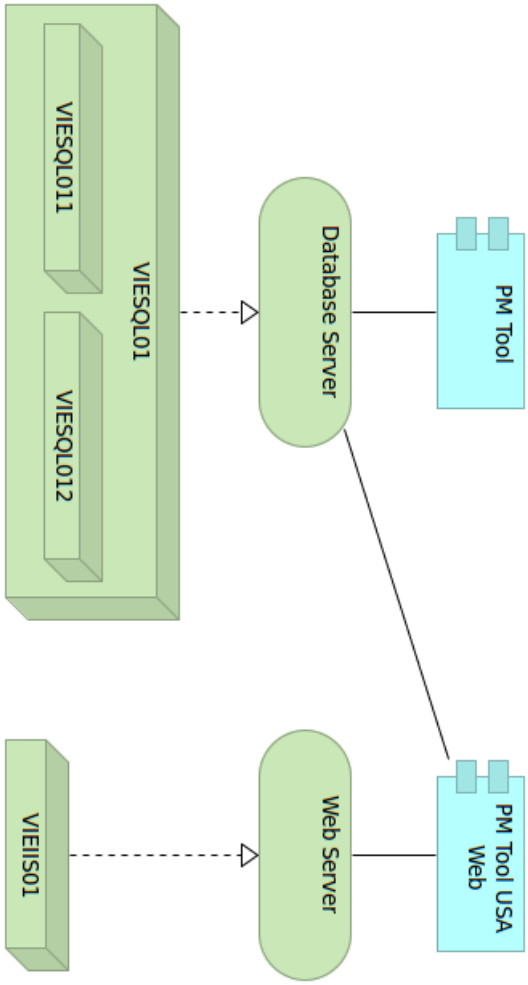


Abbildung 3.53: ArchiMate-Darstellung der Infrastruktur der Project Management Software.

3.4 Resultate und Erkenntnisse

In diesem Kapitel werden die erzielten Resultate, sowie die gewonnenen Erkenntnisse und Erfahrungen (oft auch „Lessons Learned“ genannt) festgehalten.

3.4.1 Resultate

Das Resultat dieser Fallstudie ist ein ArchiMate-Modell, welches die IT-Architektur entsprechend den von den Stakeholdern gewünschten Aussagen repräsentiert. Das Modell liegt dem Partnerunternehmen in digitaler Form vor und ein kleiner Auszug davon wurde in dieser Arbeit in Kapitel 3.3.3 betrachtet und erläutert.

Mitarbeiter des Partnerunternehmens, die sich in Zukunft neu für das Modell interessieren, ist es empfohlen, zum Einarbeiten in das Modell Kapitel 3.3 zu lesen. ArchiMate wird darin soweit erklärt, wie es zum Verständnis des Modells notwendig ist, da nicht alle Möglichkeiten von ArchiMate auch genutzt werden. Außerdem wird erläutert und beispielhaft erklärt, wie die jeweiligen Aussagen in ArchiMate abgebildet sind.

3.4.2 Erkenntnisse und Erfahrungen

In diesem Kapitel werden die im Zuge der Arbeit gewonnenen Erkenntnisse und Erfahrungen beschrieben.

3.4.2.1 Kommunikation

Es hat sich in den Gesprächen zur Erst-Erstellung des Modells als hilfreich herausgestellt, mit Stakeholdern zweimal in kurzem Abstand über das Modell zu sprechen. Beim ersten Gespräch wurden die Fakten in Tabellenform niedergeschrieben (siehe Kapitel 3.1.3.2). Nach dem Gespräch wurde das Modell in Archi in einer eigenen View um die neuen Fakten erweitert. Beim zweiten Gespräch wurde diese View als Gesprächsbasis verwendet.

Diese Visualisierung hatte zwei positive Effekte auf die Gesprächspartner. Einerseits wurde ihnen dadurch oft bewusst, dass sie beim ersten Gespräch etwas übersehen hatten oder etwas vom Interviewer falsch verstanden wurde. Andererseits erlebten sie ArchiMate anhand von etwas, dass ihnen vertraut ist. Das zweite Gespräch war also in der Regel zusätzlich eine Einführung in ArchiMate.

Diese Form der Kommunikation hat sich aus diesen beiden Gründen als sehr gut erwiesen. Dennoch bleibt ein menschliches Problem bestehen: Wenn jemand nicht zugeben möchte, dass er oder sie etwas nicht verstanden hat, bleiben Missverständnisse und Fehler unentdeckt. Das kann zum Beispiel aufgrund von Unsicherheit, Zurückhaltung oder Desinteresse passieren. Man kann versuchen, das durch Rückfragen und Empathie auszugleichen, aber es gibt für diesen menschlichen Faktor keine Musterlösung.

3.4.2.2 ArchiMate als Architektursprache

ArchiMate hat sich gut bewährt als Modellierungssprache für diese Fallstudie. Die semantische Nähe zu UML ist gut bei den TechnikerInnen angekommen und die Modellierungsmöglichkeiten, die ArchiMate bietet, waren mehr als ausreichend. Es wäre, wie bereits angesprochen, möglich, das ArchiMate-Modell auf TOGAF zu portieren. Dafür besteht bisher jedoch kein Bedarf.

3.4.2.3 Archi

Das verwendete Software-Tool Archi hat sich im Allgemeinen als einsteigerfreundlich und intuitiv erwiesen. Die Stakeholder fanden sich aufgrund der flachen Lernkurve schnell im Tool zurecht.

Features wie etwa die Generierung von Reports sind allerdings bedeutend simpler gehalten als vom Autor dieser Arbeit gehofft. Außerdem gibt es keine Möglichkeit einer automatisierten Auswertung in irgendeiner Form.

Als das Modell immer größer wurde, wurde auch das Tool spürbar langsamer. Das erweckte bei einigen Stakeholdern den negativen Eindruck, dass das Tool nicht für professionelle Modellierung geeignet ist.

Der größte Negativpunkt ist jedoch die Untauglichkeit von Archi für Multi-User-Szenarien. Das kommt daher, dass ein Archi-Modell in eine einzige XML-Datei gespeichert wird. Wenn zwei verschiedene Personen gleichzeitig an dem Modell arbeiten, so überschreiben sie beim Speichern die Arbeit des jeweils anderen.

Im Zuge dieser Arbeit wurde versucht diesen Nachteil durch die Verwendung der Software Apache™ Subversion® zur Versionsverwaltung auszugleichen. Das abgespeicherte Modell wurde in einem Subversion-Repository abgelegt, wodurch Änderungen nachvollziehbar, zuordenbar und umkehrbar wurden. (Nähere Informationen zur Funktionsweise von Subversion sind in der offiziellen Dokumentation [sub11] zu finden.)

Leider war das Ergebnis nicht zufriedenstellend. Die Versionsverwaltungs-Software stieß schnell an ihre Grenzen, da das gesamte Modell in einer einzigen Datei gespeichert ist. Es kam regelmäßig zu Versionskonflikten, wenn verschiedene Personen Änderungen am Modell machten.

Es gäbe noch die Möglichkeit, dass MitarbeiterInnen die Datei bei Benutzung sperren („svn lock“), wodurch andere sie nicht mehr überschreiben könnten. Das löst jedoch ebenfalls nicht das Problem, wie in folgendem Beispiel dargelegt wird: Ein Mitarbeiter führt lokal Änderungen durch. Diese werden aber vom Server nicht übernommen, weil die Datei von einem anderen Mitarbeiter gesperrt ist. Wenn der erste Mitarbeiter abwartet, bis die Datei entsperrt ist, und danach seine Änderungen zum Server übergibt, kommt es ebenfalls zu Versionskonflikten.

Mit Hilfe eines Copy-Modify-Merge-Workflows können Personen, die Änderungen an der Datei vornehmen wollen, die Datei kopieren (copy), ändern (modify) und, wenn die Änderungen abgeschlossen sind, zusammenfassen (merge), wodurch sie sämtliche ihrer mit anderen in der Zwischenzeit vorgenommenen Änderungen in Konflikt stehende Änderungen von Subversion aufgezeigt bekommen und händisch beheben können. Das würde bedeuten, dass alle an dem Modell arbeitenden Personen bei ihren Änderungen XML-Text lesen müssten, um Konflikte zu beheben.

Letztenendes wurde keine zufriedenstellende technische Lösung für das Multi-User-Szenario gefunden. Das ist ein Umstand, der das Tool nach Meinung des Autors für professionelle Anwendung

in Unternehmen disqualifiziert. Laut Diskussionen im Entwickler-Forum wird an diesem Problem bereits gearbeitet [7]. Das ist allerdings ein komplexes Unterfangen, da ein ArchiMate-Modell vielschichtige Abhängigkeiten beinhaltet, mit seinen Elementen und deren Beziehungen, die wiederum in mehreren Views verwendet werden.

3.4.2.4 Erste Erfolge

Die erste praktische Anwendung des Modells geschah im Zuge einer Datenbank-Konsolidierung. Das Projektteam hatte den Auftrag, verschiedene alte Datenbank-Server durch einen neuen zu ersetzen. Durch das EAM konnten sofort alle betroffenen Applikationen und deren BetreuerInnen ausfindig gemacht werden. Ohne dem Modell wären mehrere Gespräche notwendig gewesen, um an diese Informationen zu kommen.

Der zweite Anwendungsfall geschah im Zuge eines Projekts zur Einführung eines Dokumenten-Management-Systems (DMS). Aufgabe ware es, für eine Präsentation für das Management die Auswirkungen der Einführung auf Applikationsebene darzustellen. Dem Management sollten damit die Vorteile und das Ausmaß der Änderungen aufgezeigt werden.

Für das Management ist eine High-Level-Sicht angebracht und daher wurden zuerst vom Projektteam alle Datenobjekte, die mit Dokumenten zu tun haben und somit in das DMS migriert werden in einer ArchiMate-View zur Anzeige gebracht. Diese View ist in Abbildung 3.54 zu sehen. Dieses Projekt hat eine Erweiterung des Schemas notwendig gemacht. Zur Darstellung der Aussage, dass Datenobjekte aus anderen Datenobjekten zusammengesetzt sein können, wird eine *Composition Relation* verwendet. In der Abbildung ist zum Beispiel das Datenobjekt *Hardware Documentation* zusammengesetzt aus *Metadata of Hardware Documentation* und *Hardware Documentation On File Shares*. Diese Aufteilung hat sich aus den Limitierungen der jeweiligen Applikationen ergeben. Die Hardware-Dokumentation liegen auf *File Shares*, welche nicht in der Lage sind, ausreichend Metadaten pro Dokument festzuhalten und durchsuchbar zu machen. Dazu wird die Applikation *HW Doc* verwendet. Genauso verhält es sich mit *Offer Documentation* – zusammengesetzt aus *Offer Documents On File Shares* und *Metadata for Offer Documents*.

Development Documentation wird in unterschiedlicher Weise festgehalten. Manche Teams legen ihre Dokumentation auf File Shares ab, andere im Project Workspace des dazugehörigen Projekts und manche in ihren Versionskontroll-Tools. Eine einheitliche Art der Dokumentation kann viele Vorteile bringen. Beispielsweise ermöglicht es, eine zentrale Suche einzurichten und es wäre immer klar, wo Dokumente zu finden sind, anstatt projekt- oder systemabhängig umdenken zu müssen.

Die Einführung des neuen DMS bringt daher klarerweise auch auf der Geschäftsebene Änderungen, wie zum Beispiel eine andere Ablage der Development Documentation. Zu diesem Zweck wird das EAM ebenfalls behilflich sein.

In der linken Spalte der Abbildung 3.54 sind die Applikationen zu sehen, die von der DMS-Einführung zumindest teilweise abgelöst werden. Die Applikation *HW Doc* wurde bereits angesprochen. Ihre Funktion der Metadatenspeicherung und -suche wird obsolet, da das DMS diese übernehmen soll. Funktionen sind in dieser View ebenfalls nicht dargestellt, da es zu detailliert werden würde für den Zweck der Präsentation.

Die Abbildung 3.55 zeigt den Zustand der Datenobjekte und deren Mastapplikationen nach Einführung des DMS. Sämtliche Daten sind in der neuen Applikation angesiedelt, anstatt auf

mehrere aufgestreut zu sein. Das DMS wird in *Files* genannte Bereiche unterteilt sein. Die *Development Documentation* wird beispielsweise in das *Product File* wandern, wie der Abbildung zu entnehmen ist.

Es ist klar ersichtlich, dass die Einführung des DMS die durch die Applikationen erzwungene oder historisch bedingte Aufteilung der Datenobjekte eliminieren wird. Zehn Applikationen werden zumindest teilweise abgelöst durch eine einzige. Aus Datensicht bringt die Umstellung also klare Vorteile.

Das Modell hat also mit geringem Aufwand den aktuellen Zustand der betroffenen Systeme auf eine Weise sichtbar gemacht, die abstrakt genug ist für das Management. Informationen wie etwa Datenflüsse zwischen den einzelnen Applikationen oder über die technische Infrastruktur dahinter können auf einfache Weise ausgeblendet werden, da sie für den Zweck dieser Präsentation nicht relevant waren.

Im geschäftlichen Alltag der IT-Abteilung dient das Modell immer wieder als Nachschlagewerk und als Referenz. Es hat sich wiederholt bewährt und wird höchstwahrscheinlich weiter ausgebaut werden.

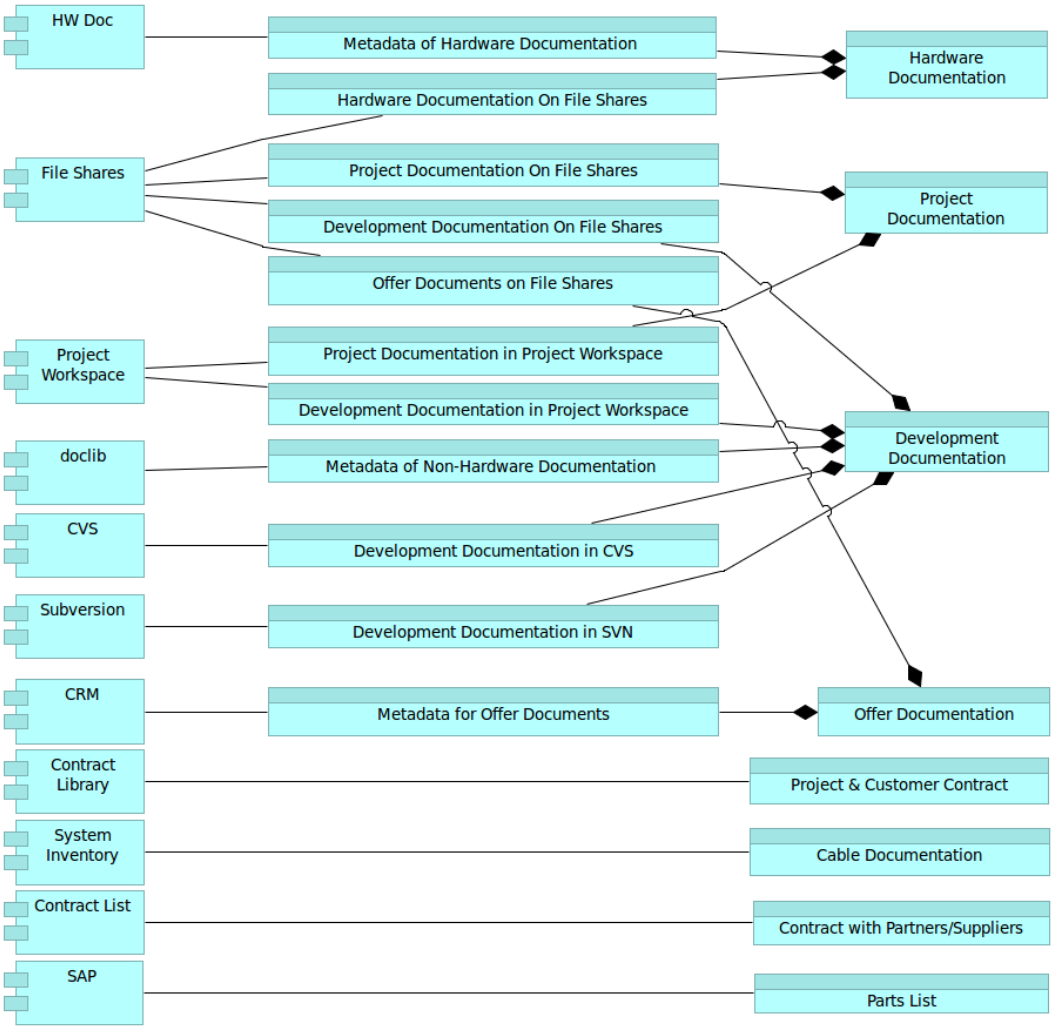


Abbildung 3.54: Auszug aus einer Präsentation: Zustand vor der Einführung eines neuen Dokumenten-Management-Systems.

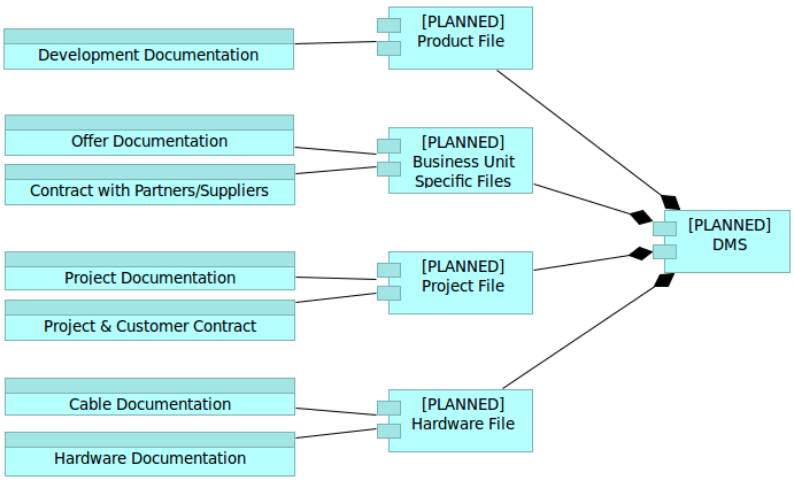


Abbildung 3.55: Auszug aus einer Präsentation: Zustand mit neuem Dokumenten-Management-System.

4 Diskussion und Zukunftsaussichten

In diesem Kapitel wird betrachtet, welche Aspekte des Modells noch verbesserungswürdig und ausbaufähig sind. Es werden dem Unternehmen einige Vorschläge gemacht, wie es das vorhandene Modell weiterentwickeln kann und wie es das Modell auf eine Weise in die Arbeitsabläufe integrieren kann, sodass es immer aktuell gehalten wird.

4.1 Nächste Schritte zur sinnvollen Weiterentwicklung des Modells

Der sinnvollste nächste Schritt ist der Ausbau des Business Layers. Bisher ist er nur in Form einer hierarchischen Struktur der Prozesse und Funktionen abgebildet. Temporale Abfolgen und weitere Konzepte wie Events, Produkte oder Services fehlen bisher vollständig, sind aber mit ArchiMate modellierbar. Es wurden bereits Gespräche gestartet, die bisherige Geschäftsprozess-Dokumentation durch ArchiMate abzulösen. Der große Nutzen gegenüber der jetzigen Situation ist die dadurch eliminierte Doppelgleisigkeit. Der Mehraufwand der doppelten Pflege entfällt und mit ihm verschwindet auch die Möglichkeit eventuell entstehender Abweichungen und Fehler.

Nach dem Business Layer kann der Data Layer weiter ausgebaut werden. Dieser ist aktuell nur sehr elementar erfasst (Was ist die Masterapplikation/Masterdatei des Datenobjekts und wohin wird es übertragen?). Wenn im Business Layer abgebildet ist, durch welche Tätigkeiten Information geschaffen wird, kann das Informationsobjekt mit dem Datenobjekt des Application Layers verknüpft werden. Dann kann eine Übersicht geschaffen werden, wo im Unternehmen Information erzeugt, gespeichert, konsumiert und wohin sie übertragen wird. Mit diesen Informationen können eventuell wieder Doppelgleisigkeiten durch Mehrfach-Erzeugung von Informationen entdeckt werden. Ein(e) Software-EntwicklerIn kann dadurch schnell sehen, über welche Applikationen und Schnittstellen man benötigte Informationen beziehen kann.

Eine weitere sinnvolle Maßnahme wäre die Evaluierung von Software-Alternativen zu dem Tool Archi. Im vorigen Kapitel wurde bereits auf die Schwächen von Archi eingegangen. Da ArchiMate ein herstellerunabhängiger Standard ist, gibt es unterschiedliche Tool-Anbieter. Ein entsprechendes Projekt zur Evaluierung von Archi-Alternativen wurde bereits gestartet.

Eine weitere wichtige Maßnahme wäre es, die Rolle eines/einer ArchitektIn einzuführen. Ein(e) ArchitektIn sollte die Weiterentwicklung des EAMs koordinieren und verantworten. Wenn das nicht gewünscht ist, kann alternativ ein Konsortium unterschiedlicher StakeholderInnen einberufen werden, welche die Weiterentwicklung lenken.

Die kleineren Niederlassungen sollten ebenfalls besser in dem Modell berücksichigt werden. Dazu wäre es notwendig, VertreterInnen zu bestimmen, welche die Geschäftsprozesse ihrer Niederlassung in das Modell eintragen. Dann könnte, wie in Kapitel 3.3.2 unter Aussage elf bereits angesprochen, die Abbildung der Niederlassungen überarbeitet werden.

4.2 Maßnahmen zur Integration des Modells in die Arbeitsabläufe

Das Modell muss ständig aktuell gehalten werden. Um das zu gewährleisten, müssen Personen damit beauftragt werden, Änderungen in der realen Welt im Modell ebenfalls nachzuziehen. Es folgt ein Vorschlag, wie das umgesetzt werden könnte.

Abbildung 4.1 zeigt eine Empfehlung zur Handhabung Architektur-ändernder Projekte.

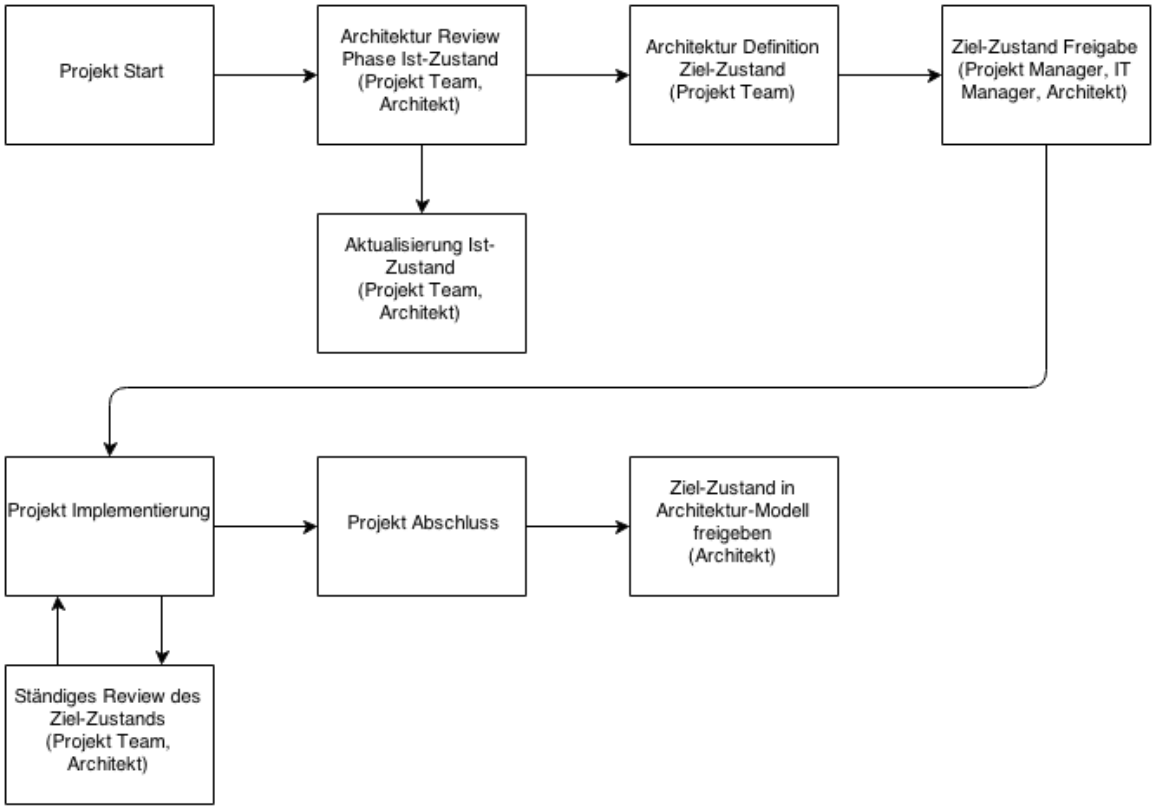


Abbildung 4.1: Empfehlung zur Handhabung Architektur-ändernder Projekte

Zum Projekt-Start wird der Ist-Zustand im Architektur-Bild vom Architekten gemeinsam mit dem Projekt-Team überprüft und eventuell aktualisiert. Das Projekt-Team definiert dann in ArchiMate den Ziel-Zustand, welcher im Anschluss von Projekt-Manager, IT-Manager und dem Architekten geprüft und freigegeben wird.

Während das Projekt umgesetzt wird, behält das Projekt-Team ständig den Ziel-Zustand im Auge und passt ihn an aktuelle Erkenntnisse und Entwicklungen an. Diese Anpassungen müssen in Übereinstimmung mit dem Architekten passieren. Das garantiert, dass die Ziele im Sinne der initialen Freigabe bleiben. Nachdem das Projekt abgeschlossen ist und die Architektur somit geändert wurde, übernimmt der Architekt den neuen Zustand in das Architektur-Modell.

Auf diese Weise wird nicht nur das Modell stets aktuell gehalten, sondern auch aktiv zur Betrachtung des aktuellen Zustands und zum Entwerfen und Darstellen des Ziel-Zustands verwendet. Darüberhinaus gibt es einen Qualitätssicherungs-Schritt, in dem das Management und der Architekt die Änderungen besprechen und eventuell widerrufen oder anpassen.

5 Zusammenfassung

Im Zuge dieser Fallstudie sollte ein Modell der IT-Architektur entstehen. Dazu wurden die Wünsche verschiedener Personen bezüglich des Modells aufgenommen (siehe Kapitel 3.1.2). Da diese Wünsche freisprachlich formuliert waren, wurden sie in 15 strukturierte Aussagen übersetzt (siehe Kapitel 3.1.3.1). Diese Aussagen sollen von dem gewünschten „Gesamtbild“ ablesbar sein und bilden somit die Basis für das weitere Vorgehen. Basierend auf diesen Aussagen wurde eine Tabelle definiert, welche im Anschluss zur Befragung involvierter Personen zur Erhebung der IT-Architektur verwendet wurde (siehe Kapitel 3.1.3.2).

Zur Realisierung des Modells wurde die Disziplin Enterprise Architecture Management (EAM) herangezogen und verschiedene EAM-Frameworks betrachtet, die zur Realisierung der Aufgabe in Frage kamen (siehe Kapitel 2). Aus den Soll-Aussagen wurde ein Werkzeug-unabhängiges Schema erstellt (siehe Kapitel 3.1) und aus den daraus gewonnenen Informationen wurde die Entscheidung für das Framework ArchiMate als Modellierungssprache getroffen (siehe Kapitel 3.2.1). Aus den Aussagen wurde ein ArchiMate-Schema erstellt und das Modell schließlich mittels ArchiMate realisiert (siehe Kapitel 3.3).

Die Datei mit dem ArchiMate-Modell wurde dem Partnerunternehmen gemeinsam mit Empfehlungen zur weiteren Vorgehensweise übergeben (siehe Kapitel 4).

Internet Referenzen

[1]

[2] *Archi*, abgerufen am 4. März 2014. http://www.archimatetool.com.

[3] *Archi|About*, abgerufen am 4. März 2014. http://www.archimatetool.com/about.

[4] *ArchiMate® 2 Tool Certification Register*, abgerufen am 4. März 2014. http://www.opengroup.org/certifications/archimate/ts-register.

[5] *BiZZdesign Architect Functionality*, abgerufen am 4. März 2014. http://www.bizzdesign.com/tools/bizzdesign-architect/bizzdesign-architect-functionality/.

[6] *Business Process Model and Notation (BPMN)*, abgerufen am 25. September 2014. http://www.omg.org/spec/BPMN/.

[7] *Git as a model repository*, abgerufen am 13. März 2014. https://groups.google.com/forum/#!topic/archi-dev/8sCoD6Ctj-c.

[8] *Jisc*, abgerufen am 4. März 2014. http://www.jisc.ac.uk.

[9] *Welcome to TOGAF® Version 9.1 Enterprise Edition*, abgerufen am 4. Jänner 2014. http://www.opengroup.org/togaf/.

Literatur

[AT09] H. Safari Asl and Y.F. Tang. *Document Management System design architecture for interdepartmental organization.* Master's thesis, Delft University of Technology, The Netherlands, 2009.

[BBL12] Stefan Bente, Uwe Bombosch, and Shailendra Langade. *Collaborative Enterprise Architecture – Enriching EA with Lean, Agile, and Enterprise 2.0 Practices.* Newnes, London, 2012.

[Ett05] Roland Ettema. *Adopting ArchiMate ? Expressing Architecture.* Master's thesis, Cibit Academy, The Netherlands, 2005.

[Gro13] The Open Group. *ArchiMate® 2.1 Specification.* The Open Group, December 2013.

[Han09] Inge Hanschke. *Strategisches Management der IT-Landschaft – ein praktischer Leitfaden für das Enterprise-architecture-Management.* Hanser, München, 2009.

[Han12] Inge Hanschke. *Enterprise Architecture Management – einfach und effektiv – Ein praktischer Leitfaden für die Einführung von EAM.* Hanser Fachbuchverlag, München, 2012.

[Har11] Van Haren. *TOGAF Version 9.1.* Van Haren Publishing, Zaltbommel, 10th new edition edition, 2011.

[HP13] Frank Harmsen and Henderik A. Proper. *Practice-Driven Research on Enterprise Transformation: 6th Working Conference, PRET 2013, Utrecht, The Netherlands, June 6, 2013, Proceedings.* Springer Publishing Company, Incorporated, 2013.

[Ins05] Institute For Enterprise Architecture Developments. *Trends in Enterprise Architecture 2005: How are Organizations Progressing?*, Dezember 2005. `http://www.ea-consulting.com/Reports/Enterprise%20Architecture%20Survey%202005%20IFEAD%20v10.pdf`.

[Kel07] Wolfgang Keller. *IT-Unternehmensarchitektur – Von der Geschäftsstrategie zur optimalen IT-Unterstützung.* Dpunkt.Verlag GmbH, Heidelberg, 1. Ausgabe, 2007.

[LA14] Nils Labusch and Stephan Aier. Information provision as a success factor in the architectural support of enterprise transformations. In *Business Informatics (CBI), 2014 IEEE 16th Conference on*, volume 2, Seiten 141-148, Juli 2014.

[Lan09] Marc Lankhorst. *Enterprise Architecture at Work – Modelling, Communication and Analysis.* Springer, Berlin, Heidelberg, 2009.

[LM11] Matthias Lange and Jan Mendling. An expert's perspective on enterprise architecture goals, framework adoption and benefit assessment. *In Proc. of the 6th Trends in Enterprise Architecture Research Workshop*, Helsinki, Finnland, 29. August 2011. `http://www.mendling.com//publications/11-TEAR.pdf`.

[LPW+09] Martin Op 't Land, Erik Proper, Maarten Waage, Jeroen Cloo, and Claudia Steghuis.

Enterprise Architecture – Creating Value by Informed Governance. Springer Science Business Media, Berlin Heidelberg, 2009.

[Mat11] Dirk Matthes. *Enterprise Architecture Frameworks Kompendium*. Springer DE, Berlin, 2011.

[Ope12] The Open Group. *An Introduction to ArchiMate 2.0*, Jänner 2012. Whitepaper von Andrew Josey (The Open Group) und Henry Franken (BiZZdesign), Document No.: W121.

[Rut11] Sören Ruttkowski. Comparison of IT Governance and Enterprise Architecture Management Frameworks with Emphasis on Business Relevant Key Performance Indicators. Master's thesis, Technische Universität München, Fakultät für Informatik, 2011.

[SMS+09] Dirk Stähler, Ingo Meier, Rolf Scheuch, Christian Schmülling, and Daniel Somssich. *Enterprise Architecture, BPM und SOA für Business-Analysten – Leitfaden für die Praxis ; [am Beispiel der Oracle BPA Suite 11g und der ARIS-Methode]*. Hanser Verlag, München, Wien, 2009.

[sub11] *Versionskontrolle mit Subversion*, 2011. `http://svnbook.red-bean.com/de/1.6/svn-book.pdf`.

[SZ92] J. F. Sowa and J. A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31(3):590–616, 1992. `http://www.zachman.com/images/ZI_PIcs/ibmsj1992.pdf`.

[U.Sst] U.S. Department of Defense. *DoDAF Architecture Framework Version 2.02*, 2010, August. `http://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF-DM2_v2-02_VDD.pdf`.

[Usm12] Roger Usmany. *Conceptual Modeling for Business Process Semantics*. Master's thesis, Radboud University Nijmegen, Faculty of Science, Institute of Computing and Information Sciences, The Netherlands, 2012.

[Zac87] John A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292, 1987.

[Zeine] K. Zeilenga. Rfc4510 lightweight directory access protocol (ldap): Technical specification road map. *The Internet Society*, 2006, June.