Ruprecht-Karls-Universität Heidelberg
Institut für Informatik
Lehrstuhl Software Engineering
Erstbetreuerin: Prof. Dr. Barbara Paech
Zweitbetreuer: Christian Kücherer

Master-Thesis

Analyse und Optimierung der Softwarearchitektur für einen CIO-Navigator als Entscheidungsunterstützungssystem für das Informationsmanagement eines Krankenhauses

Name: Tim Bittersohl
Studiengang: Angewandte Informatik
Abgabedatum: 15. Dezember 2016

In Erinnerung an meinen Bruder
Felix Bittersohl

Zusammenfassung

Das DFG geförderte Projekt *Semantisches Netz des Informationsmanagements im Krankenhaus* (SNIK) erstellt unter anderem eine Ontologie der Informationsmanagement-Domäne in Krankenhäusern. Eines der Ergebnisse von SNIK ist der CIO-Navigator. Dieser erlaubt, insbesondere dem Chief Information Officer als LeiterIn des Informationsmanagements, die bedarfsgerechte Darstellung von Informationen beispielsweise zum Change- und Projektmanagement. Auf deren Grundlage werden dort Entscheidungen, beispielsweise zur Projektpriorisierung, getroffen. Der aktuelle Stand der Forschung zeigt, dass die Umsetzung des CIO-Navigators als webbasiertes System als gute Entscheidung zu bewerten ist. Um eine Anforderungs- und Architekturanalyse des CIO-Navigators durchzuführen, wird zunächst ein geeignetes Architekturbewertungsverfahren ausgewählt. Dazu werden die Verfahren *architecture tradeoff analysis method* (ATAM) und *software architecture analysis method* (SAAM) gegenübergestellt und die Auswahl von SAAM begründet. In SAAM wird zur Vorbereitung eine Dokumentation der bestehenden Softwarearchitektur, basierend auf dem Quellcode und Interviews mit den Entwicklern, erstellt. Bei der Analyse der vorliegenden Softwarearchitektur werden hier Vereinfachungen vorgeschlagen sowie Auffälligkeiten aufgedeckt. Dabei zeigt sich, dass die Werkzeuge *CodeMap*, *Codemetrik* sowie *CodeCity* wertvolle Einblicke in die Struktur der Software geben. Im Anschluss werden die bestehenden Anforderungen zusammengefasst und im Bereich der nichtfunktionalen Anforderungen weiter verfeinert. Daraufhin wird die Softwarearchitektur in Bezug auf die Unterstützung der Anforderungen evaluiert. Dabei zeigt sich, dass die zentrale Komponente für den Datenzugriff zu viele Funktionalitäten beherbergt. Auf Grundlage der Ergebnisse kann die Struktur des Quellcodes des CIO-Navigators in Zukunft weiter verbessert werden. Die vorgeschlagenen Verbesserungen werden nach Abschluss des SAAM-Verfahrens im Detail diskutiert. Dabei wird beim CIO-Navigator der Einsatz des Frameworks *ASP.NET MVC* empfohlen. Ein Anliegen dieser Arbeit ist außerdem, eine fundierte Grundlage für Softwareprojekte der gleichen oder ähnlicher Ausrichtung bereitzustellen.

Summary

The DFG-funded Project *Semantic Network of Information Management in the Hospital* (SNIK), among other things, creates an ontology of the information management domain in hospitals. One of the results of SNIK is the CIO-Navigator. This enables the chief information officer in particular, as the manager of the information management, to display the required information, for example regarding change and project management. On the basis of this decisions are made, for example on the project prioritization. The current state of research shows that the implementation of the CIO-Navigator as a web-based system is a good decision. In order to carry out a requirement and architecture analysis of the CIO-Navigator, a suitable architecture evaluation method is first selected. For this purpose the *architecture tradeoff analysis method* (ATAM) and the *software architecture analysis method* (SAAM) are compared and the selection of SAAM is justified. In SAAM a documentation of the existing software architecture, based on the source code and interviews with the developers, will be prepared. In the analysis of the present software architecture, simplifications are proposed as well as conspicuous abnormalities are revealed here. It turns out that the tools *CodeMap*, *Codemetrik* as well as *CodeCity* give valuable insights into the structure of the software. Subsequently, the existing requirements are summarized and further refined in the area of non-functional requirements. The software architecture is then evaluated with regard to the support of the requirements. This shows that the central component for data access has too many functionalities. Based on the results, the structure of the source code of the CIO-Navigator can be further improved in the future. The proposed improvements will be discussed in detail after completion of the SAAM procedure. In this discussion, the use of the *ASP.NET MVC* framework is recommended for the CIO-Navigator. Another concern of this work is to provide a sound foundation for software projects of the same or similar orientation.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.

Zuerst gebührt mein Dank Frau Prof. Dr. Barbara Paech, die meine Masterarbeit betreut hat. Besonderer Dank gilt meinem Betreuer Christian Kücherer für das Bereitstellen dieses interessanten Themas. Er hatte immer eine offene Tür für mich und betreute mich geduldig. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken. Dank gilt auch meinen Eltern Andrea und Jörg Bittersohl für ihre Unterstützung.

Weiter möchte ich mich auch bei meinen Kommilitonen Anatoli Zeiser und Dominik Sterk und allen weiteren KorrekturleserInnen und FreundInnen für ihre Hilfe und ihr Interesse bedanken.

Inhaltsverzeichnis

1 Einleitung

EntscheidungsträgerInnen sind in ihrem Umfeld mit zunehmender Komplexität in Bezug auf ihre Aufgaben konfrontiert. Starke Konkurrenz, schnelle Entscheidungsgeschwindigkeit, eine Überfülle an Informationen und über das ganze Unternehmen verteilte Daten sind einige der Ursachen.
Zudem sind Entscheidungen für EntscheidungsträgerInnen aufwändiger zu treffen. Dies liegt an der fortschreitenden Technologie, der zunehmenden Anzahl und Komplexität von Optionen und der steigenden Geschwindigkeit von Veränderungen. Insbesondere gilt dies für die Nutzung von Diensten im Bereich der Informationstechnologie [24].

Interaktive computerbasierte Entscheidungsunterstützungssysteme können die Entscheidungsfindung unterstützen. Sie schlagen oftmals Lösungen vor, welche nachweislich besser sind als die von Menschen ohne eine solche Unterstützung ausgewählten [11]. Dabei helfen sie, kognitive Benachteiligungen von Menschen beim Integrieren vielfältiger Informationsquellen auszugleichen. Außerdem stellen sie einen einfachen Zugang zu relevantem Wissen zur Verfügung und unterstützen zudem den Prozess der Strukturierung und Optimierung von Entscheidungen [11].

Entscheidungsunterstützungssysteme werden typischerweise für strategische und taktische Entscheidungen vom höheren Management genutzt. Entscheidungen sollten häufig auftreten und die Konsequenzen der Entscheidungen sollten deutlich sein, damit sich die Zeit für das Durchdenken und Modellieren der Entscheidungsfindung auf lange Sicht tatsächlich lohnt [11].

In dieser Arbeit wird eine entscheidungsunterstützende Software, die für die BereichsleiterIn der IT (Chief Information Officer) in einem ausgewählten Krankenhaus entwickelt wird, untersucht. Diese Software nennt sich kurz CIO-Navigator (CION). Wie in anderen Unternehmen auch, hat das Informationsmanagement im Krankenhaus eine strategische Bedeutung und ist zu einem entscheidenden Erfolgs- und Wettbewerbsfaktor geworden [16]. Sich aus dem Einsatz von Entscheidungsunterstützungssystemen ergebende Vorteile können beispielsweise die Steigerung der Entscheidungsqualität [13], gesteigerte Produktivität, Effizienz und Effektivität [11] sowie Zeitersparnis, Kostenreduktion, verbesserte Kommunikation und Angestelltenzufriedenheit [13] sein. Entscheidungsunterstützungssysteme werden inzwischen ausgiebig bei Geschäfts- und Managementtätigkeiten eingesetzt. Geschäftsunterstützende Dashboards und andere Software erlaubt EntscheidungsträgerInnen unter anderem schneller Entscheidungen zu treffen, Trends zu entdecken und Geschäftsressourcen besser einzusetzen.

Um die bestmögliche Qualität der unterstützenden Software sicher zu stellen, soll der aktuelle Stand der Forschung in Bezug auf Entscheidungsunterstützungssysteme bestimmt und angewendet werden. Auf dieser Grundlagen soll eine optimale Architektur von CION und damit

allgemein von Entscheidungsunterstützungssystemen kleineren Umfangs herausgearbeitet werden. Dies ist von besonderer Bedeutung, da sich Informationssysteme im Gesundheitsbereich immer noch in einer Phase schneller Entwicklungen befinden [19]. Dabei sind viele Fragestellungen bei der Architektur, der Funktionalität und im Management immer noch unbeantwortet [19].

Um die EntscheidungsträgerInnen bestmöglich zu unterstützen, ist neben den genannten Faktoren ein solider Aufbau der Software von herausragender Bedeutung, der sich in der Softwarearchitektur eines Informationssystems niederschlägt. Die Softwarearchitektur liefert damit die Grundlagen für eine optimale Funktionalität, Qualität, Anpassbarkeit und damit verbunden für eine lange Lebensdauer des umgesetzten Entscheidungsunterstützungssystems.

1.1 Geschichte von Entscheidungsunterstützungssystemen

Bei Entscheidungsunterstützungssystemen kann beginnend mit ihren ersten Anwendungen bereits auf eine lange Geschichte zurückgeblickt werden. Um wesentliche Komponenten von Entscheidungsunterstützungssystemen und deren Möglichkeiten auszumachen, folgt eine kurze Zusammenfassung der geschichtlichen Entwicklung dieser Systeme.

Forscher aus verschiedenen Disziplinen haben Entscheidungsunterstützungssysteme seit mehr als 45 Jahren untersucht. Entscheidungsunterstützungssysteme gehen auf computerunterstützte Modelle zur Entscheidungsfindung und Planung aus der Zeit vor den 1960er Jahren zurück [35]. Erstmals wurde damals demonstriert, dass EntscheidungsträgerInnen von einem computerbasierten Entscheidungssystem profitieren können [7].

In den 1970er Jahren wurden erste abstraktere Theorien für Entscheidungsunterstützungssysteme entwickelt und damit eine theoretische Grundlage für solche Systeme geschaffen [10]. Es war außerdem eine Zeit konzeptueller und technologischer Weiterentwicklung für Entscheidungsunterstützungssysteme [7]. Der Schwerpunkt lag dabei auf der individuellen Unterstützung von EntscheidungsträgerInnen [10].

Dieser Schwerpunkt wurde mit Einführung des PCs in den 1980er Jahren weiter ausgebaut [7]. Systeme zur Finanzierungsplanung und Gruppen-Entscheidungsunterstützungssysteme kamen in den frühen 1980er Jahren auf [10]. Gruppen-Entscheidungsunterstützungssysteme unterstützen seither kommunikationsorientierte Aufgaben bei der Entscheidungsfindung in Teams [7]. Mitte der 1980er Jahre wurden Entscheidungsunterstützungssysteme vorgeschlagen und umgesetzt, welche Wissenssysteme und Entscheidungsunterstützungssysteme kombinieren [10]. In den späten 1980er Jahren verbreiteten sich Führungsinformationssysteme in Unternehmen und erweiterten so die Vielfalt von Entscheidungsunterstützungsmöglichkeiten [7].

In den frühen 1990er Jahren führten manche Unternehmen durch die Nutzung von Werkzeugen zur ad-hoc Analyse von geschäftlichen Daten eine neue Art von datenbasierter Entscheidungsunterstützung ein [7].
Datenbasierte Entscheidungsunterstützungssysteme ermöglichten es EntscheidungsträgerInnen erstmals, große Datenbestände zu analysieren und auf dieser Grundlage Entscheidungen

zu treffen [4]. Die ersten für Analysezwecke optimierten zentralen Datenbanken von Unternehmen wurden fertig gestellt [7].
Während vorher bei Informationssystemen technische Probleme im Mittelpunkt standen, war nun außerdem ein Wechsel zu organisatorischen und sozialen Fragestellungen zu beobachten [19]. Darunter fallen auch Aspekte des Änderungsmanagements und des strategischen Informationsmanagements [19]. Besonders bei Informationssystemen im Gesundheitsbereich wurden diese Fragestellungen sehr bedeutsam [19].

Ab Mitte der 1990er Jahre wirkte sich die zunehmende Nutzung des World Wide Web auch auf Entscheidungsunterstützungssysteme aus. Webbasierte Entscheidungsunterstützungssysteme rückten in den Mittelpunkt [10].

In den späten 1990er Jahren entstanden Firmen-Intranets, welche den Informationsaustausch und die Wissensverwaltung unterstützten und die Bereitstellung von schnellen und sicheren Software-Services über das gesamte Unternehmen ermöglichten. Außerdem beinhalteten nun einige der verwendeten Werkzeuge zur Entscheidungsunterstützung Berichtsfunktionalitäten, Optimierungs- und Simulationsmodelle, aus großen Datenmengen gewonnene Informationen und Daten-Visualisierungen [7].

Die Weiterentwicklung der Entscheidungsunterstützungssysteme wird heute stark durch vorhandene Werkzeuge bestimmt [35]. Es ist außerdem zu beobachten, dass verschiedene im Bereich des Krankenhauses angesiedelte Entscheidungsunterstützungssysteme zur Bewältigung von Aufgaben zunehmend miteinander kombiniert werden [35].

Eine Übersicht über die soeben beschriebenen und weiteren historischen Entwicklungen im Rahmen von Entscheidungsunterstützungssystemen ist in Abbildung 1.1 gezeigt.

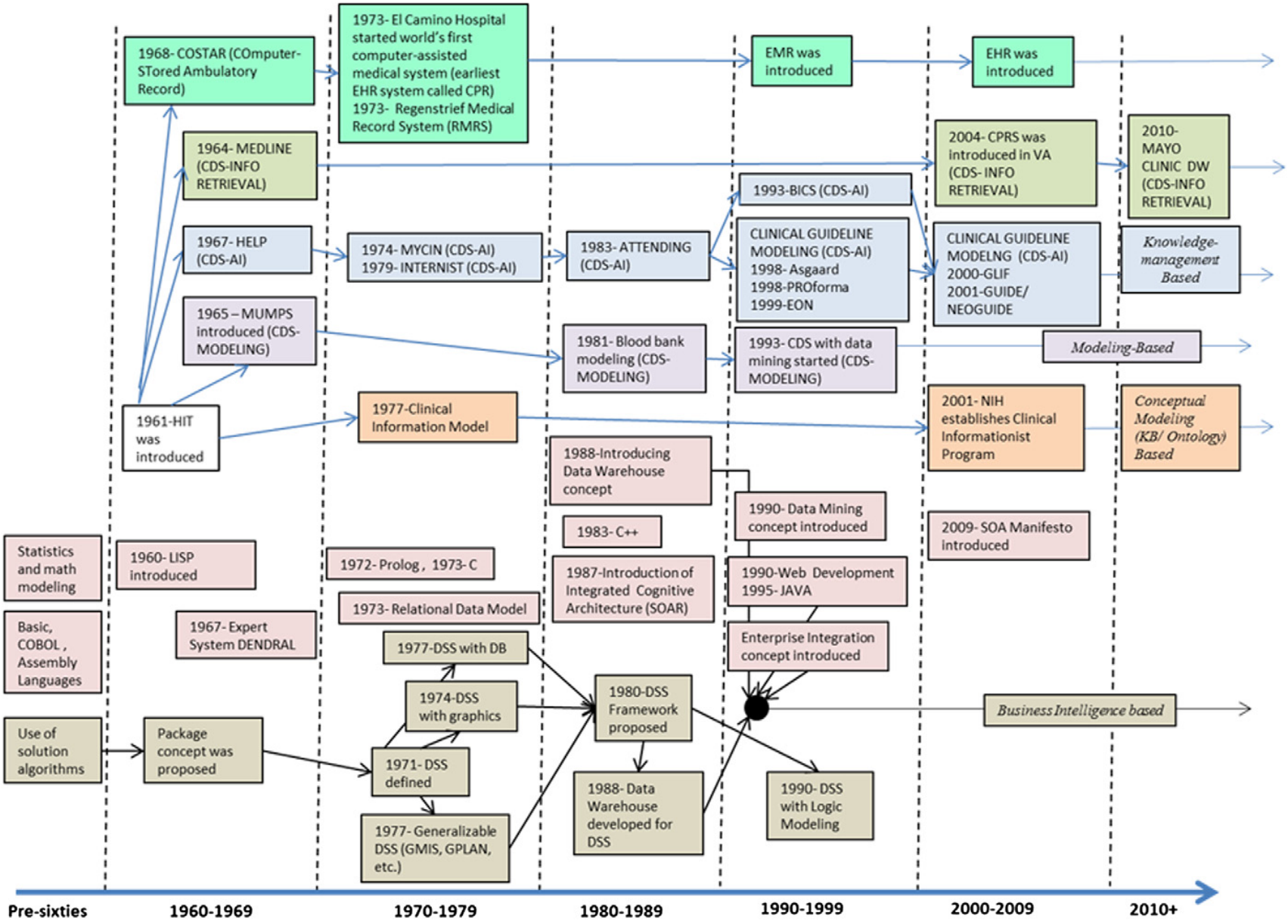


Abbildung 1.1: Übersicht über die Geschichte von Entscheidungsunterstützungssystemen (Quelle: [35])

1.2 Projektbezogenheit von Entscheidungsunterstützungssystemen

Um eine gute Softwarearchitektur aufzubauen ist es hilfreich, die Architektur bereits etablierter und erprobter Systeme zu analysieren und sich an diesen zu orientieren.

Vor allem in den letzten 20 Jahren wurden eine Vielzahl an Konzepten und Systemen zur Entscheidungsunterstützung entwickelt. In dieser langen Zeitspanne konnten sich jedoch keine von diesen als allgemeine Standards durchsetzen. Daher wird angenommen, dass die Entwicklung von entscheidungsunterstützenden Systemen in großem Ausmaß vom speziellen Projekt abhängig ist [40].

Damit können bereits veröffentlichte Architekturen im Bereich der Entscheidungsunterstützungssysteme nur in Teilaspekten als Vorlage oder Anregung dienen. Um jedoch einen Überblick zu gewinnen und mögliche Ausprägungen aufzuzeigen, werden solche Architekturen in dieser Arbeit dennoch untersucht.

1.3 Ziele der Arbeit

Die vorliegende Arbeit verfolgt mehrere Ziele.
Ziel 1: Der bestehende Prototyp des CIO-Navigators zur entscheidungsspezifischen Darstellung von Informationen des Informationsmanagement wird einer detaillierten Architekturanalyse mit SAAM unterzogen.
Voraussetzung dafür ist **Ziel 2**: Die Verfeinerung und Vervollständigung der vorliegenden, architekturrelevanten Anforderungen.
Da der CIO-Navigator Daten aus unterschiedlichen Quellen aggregiert, kann der Einsatz eines Integrationsservers sinnvoll sein.
Dies wird in **Ziel 3**: Die Spezifikation der Datenintegration mit dem Integrationsserver *Teiid* untersucht.
Ziel 4: Identifikation von verbesserungsfähigen Schwachstellen des CIO-Navigators auf Basis der Architekturanalyse.
Ziel 5: Exemplarische Durchführung der Verbesserung der Architektur. Hier auch durchgängige Verwendung von *ASP.NET MVC* als Model-View-Controller-Entwurfsmuster.

1.4 Aufbau der Arbeit

In Kapitel 2 werden Grundlagen zu Entscheidungsunterstützungssystemen und Softwarearchitekturen sowie Anforderungen an diese aufgeführt. Weiterhin wird dort das SNIK-Projekt vorgestellt und die wesentlichen Konzepte und Ideen des CIO-Navigators erläutert. In Kapitel 3 wird die Systematik der durchgeführten Literaturrecherche erläutert. Der daraus hervorgehende Stand der Forschung schließt sich dem an. Darauf wird in Kapitel 4 die Systemarchitektur des CIO-Navigators dargelegt. In Kapitel 5 wird zunächst das Architekturbewertungsverfahren ausgewählt und beschrieben. Im Anschluss folgt die Durchführung der Architekturbewertung.

Im darauf folgenden Kapitel 6 wird eine Ausführung zu Verbesserungsvorschlägen zur Softwarearchitektur des CIO-Navigators gegeben. Dort befindet sich zudem eine Bewertung zu den Ergebnissen dieser Arbeit. Schlussfolgerungen aus dieser Arbeit werden in Kapitel 7 gezogen. Hier wird abschließend auch ein Ausblick auf zukünftige Arbeiten und weiteren Forschungsbedarf gegeben. Im Anhang A befinden sich das Glossar sowie Materialien zur Literaturrecherche und Dokumentation des CIO-Navigators, die im Hauptteil der Arbeit keinen Platz gefunden haben.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der Einordnung und Abgrenzung von Entscheidungsunterstützungssystemen vorgestellt. Entscheidungsunterstützungssysteme unterscheiden sich von Systemen zur automatischen Entscheidungsfindung, die nicht im Fokus dieser Arbeit stehen.

Die vorgestellten Klassifikationen ermöglichen die Einordnung des entwickelten Prototyps in die verschiedenen Arten von Entscheidungsunterstützungssystemen. Des Weiteren schließen sich Definitionen und Ausprägungen zu Softwarearchitekturen und Grundlagen zu Anforderungen und Performanz an. Am Ende wird eine Übersicht über das SNIK-Projekt und die grundlegenden Ideen des CIO-Navigators gegeben.

2.1 Klassifikationen und Einteilungen zur Entscheidungsunterstützung

In diesem Abschnitt werden Klassifikationen und Einteilungen im Rahmen der Entscheidungsunterstützung aufgeführt.

2.1.1 Klassifikation von Entscheidungsproblemen

Entscheidungsprobleme können unterschiedlich ausfallen. Das Spektrum reicht hierbei von Einzelentscheidungsfindungen bis zu sich wiederholenden Routine-Entscheidungsfindungen. Ein Entscheidung besteht aus den drei Komponenten Daten, Prozess und Evaluierung.
Folgende drei Entscheidungsprobleme können nach Gorry und Morton unterschieden werden [15]:

- strukturiert
  Die drei Komponenten einer Entscheidung sind festgelegt. Eine Standard-Lösungsmethode existiert prinzipiell. Eine komplette Automatisierung ist oft machbar.
  Beispiel: Welche StudentInnen erfüllen bestimmte verpflichtende Zulassungsvoraussetzungen?

- teilstrukturiert
  Einige Elemente oder Phasen im Prozess der Entscheidungsfindung haben sich wiederholende Bestandteile, welche stark abhängig vom Urteilsvermögen des Entscheiders sind.
  Beispiel: Welche StudentInnen sollen anhand ihrer Bewerbung für einen Studiengang zugelassen werden?

- unstrukturiert
  Jede Entscheidungsfindung ist einmalig. Es existieren keine Standard-Lösungsmethoden. Eine Automatisierung ist nicht möglich.
  Beispiel: Soll der Unternehmensstandort in einer Stadt zugunsten einer neuen Lagerhalle in einer anderen Stadt geschlossen werden?

Entscheidungsunterstützungssysteme eignen sich am besten für sich wiederholende Aspekte von teilstrukturierten Problemen. Bei solchen Problemen ist die Beteiligung von Menschen für gewöhnlich nötig. Durch die teilweise vorhandene Strukturierung ist es einem Entscheidungsunterstützungssystem jedoch möglich, bei der Entscheidungsfindung zu helfen.

2.1.2 Unterstützungsdimensionen von betrieblichen Informationssystemen

Betriebliche Informationssysteme sind Softwaresysteme, welche die betrieblichen Abläufe unterstützen. Verschiedenen Arten dieser Systeme lassen sich von einander unabhängigen Unterstützungsdimensionen zuordnen.
Es werden folgende fünf prinzipielle Unterstützungsdimensionen betrieblicher Informationssysteme nach Haas unterschieden [18]:

- Verarbeitungsunterstützung
  Berechnen, Auswerten und Verdichten von Daten.
- Dokumentationsunterstützung
  Unterstützung der Erfassung, Speicherung und des Auffindens von Daten, Dokumenten und Wissen jeglicher Art.
- Organisationsunterstützung
  Unterstützung aller Aspekte der betrieblichen Organisation wie Terminmanagement, Ressourcenbelegungsplanung, Workflow-Management.
- Kommunikationsunterstützung
  Unterstützung jeglicher Art von Informationsaustausch zwischen menschlichen oder elektronischen Kommunikationspartnern innerhalb eines Unternehmens oder mit externen Partnern.
- Entscheidungsunterstützung
  Unterstützung menschlicher Entscheidungen durch intelligente wissensbasierte Anwendungssystemfunktionen.

Entscheidungsunterstützungssysteme sind Teil der Kategorie Entscheidungsunterstützung, können jedoch auch andere Unterstützungsdimensionen mit abdecken. So unterstützt beispielsweise ein Gruppen-Entscheidungsunterstützungssystem auch die Kommunikation oder ein dokumentenbasiertes System die Dokumentation.

Entscheidungsunterstützende Funktionen in Anwendungssystemen sind in der Lage, kontextsensitiv Wissen herauszusuchen und zu präsentieren [18]. Außerdem können sie dieses Wissen

gegebenenfalls einsetzen, indem sie BenutzerInnen Schlussfolgerungen oder Entscheidungsvorschläge unterbreiten bzw. Handlungen anmahnen [18].
Tabelle 2.1 zeigt, wie typischerweise die verschiedenen Systemtypen aus [17] (als Zeilen) im Rahmen dieser Unterstützungsdimensionen (als Spalten) abgedeckt werden:

Tabelle 2.1: Systemtypen und Unterstützungsdimensionen (nach [18])

Systeme	**Verarbeitungs-unterst.**	**Dokumentations-unterst.**	**Organisations-unterst.**	**Kommunikations-unterst.**	**Entscheidungs-unterst.**
Planung/ Entscheid.	+++			+	+++
Analyse	+++				
Bericht/ Kontrolle	+++	+		+	
Abrechnung	+++	++	+	+	
Operativ	++	+++	+++	+++	+++

Ein Plus-Zeichen (+) steht hierbei für eine gute Unterstützung, zwei Plus-Zeichen (++) für eine sehr gute Unterstützung und drei Plus-Zeichen (+++) für eine ausgezeichnete Unterstützung. Ein leeres Feld bedeutet keine Unterstützung.

Die Tabelle 2.1 zeigt, dass die Entscheidungsunterstützung wesentlich für Planungs- und Entscheidungssysteme sowie operative Systeme ist. Bei diesen beiden Typen von betrieblichen Informationssystemen kommen Entscheidungsunterstützungssysteme daher vorwiegend zum Einsatz.

2.1.3 Übliche Einteilungen von Entscheidungsunterstützungssystemen

Es gibt verschiedene Einteilungen für Kategorien von Entscheidungsunterstützungssystemen. Eine häufig verwendete Einteilung ist die folgende von Power und Kaparthi [32]:

- Kommunikationsbasiert
  Es werden Netzwerk- und Kommunikationstechnologien benutzt, um die Zusammenarbeit bei der Entscheidungsfindung zu vereinfachen. Kommunikationsbasierte Entscheidungsunterstützungssysteme verbinden mehrere EntscheidungsträgerInnen, welche örtlich oder zeitlich von einander getrennt sein können. Sie machen EntscheidungsträgerInnen außerdem entfernt gelegene Informationen oder Werkzeuge zugänglich [7].

- Datenbasiert
  Der Schwerpunkt liegt hier auf der Nutzung von Daten innerhalb eines Unternehmens oder auch außerhalb. Datenbasierte Entscheidungsunterstützungssysteme helfen EntscheidungsträgerInnen große Mengen von Daten unter Verwendung von Datenbankabfragen und Techniken zur ad-hoc Analyse von Daten zu organisieren, abzurufen und zu analysieren [7].

- Dokumentenbasiert
  Es werden Speicher- und Verarbeitungstechnologien für die Erstellung und Analyse von Dokumenten benutzt. Dabei können auch unstrukturierte Informationen, welche in einer Vielzahl von elektronischen Formaten vorkommen können, verarbeitet werden. Suchfunktionen sind dabei ein wichtiges Instrument.

- Wissensbasiert
  Wissen über Problemlösung, gespeichert als Fakten, Regeln, Prozeduren oder Ähnliches, wird bereitgestellt. Den EntscheidungsträgerInnen werden daraus gewonnene Handlungsempfehlungen präsentiert.

- Modellbasiert
  Die von BenutzerInnen bereitgestellten Daten und Parameter werden genutzt, um mit Hilfe eines komplexen Systems EntscheidungsträgerInnen bei einer Analyse von Situationen zu unterstützen. Diese sind nicht notwendigerweise datenintensiv. Modellbasierte Entscheidungsunterstützungssysteme verwenden formale Repräsentationen von Entscheidungsmodellen, welche entscheidungsbeeinflussende Faktoren abbilden sollen. Sie können Werkzeuge für Analyse, Optimierung, stochastischer Modellierung, Simulation und logische Modellierung nutzen [7].

Die zu entwerfende Architektur lässt sich anhand der bereits erhobenen Anforderungen als Mischform von datenbasiert und dokumentenbasiert kategorisieren.

2.2 Definitionen zur Softwarearchitektur

In diesem Abschnitt werden Definitionen im Umfeld der Architektur von Software gegeben.

2.2.1 Softwarearchitektur

Eine Softwarearchitektur beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb eines Softwaresystems. Sie ist vor allem von der Systemarchitektur zu unterscheiden, welche ausschließlich die Struktur von Systemen und ihrer Komponenten beschreibt. Die Betrachtung der Softwarearchitektur ist nicht nur abhängig vom Abstraktionsgrad, sondern auch von der Perspektive und Granularität [36].

Üblicherweise wird erwartet, dass eine Beschreibung der Softwarearchitektur über unterschiedlichen Ebenen der Granularität hinweg folgende Elemente enthält [36]:

- Grundlegende Konzepte, unter deren Berücksichtigung die Anwendungskomponenten spezifiziert werden müssen
- Integrations-Infrastrukturen
- Architekturstrategien, welche verwendet werden um konkret Qualitätskriterien abzudecken

Die Architekturentwicklung ist der Vorgang der Umsetzung der Architekturbeschreibung und damit der Erstellung der konkreten Softwarearchitektur. Für eine erfolgreiche Architekturentwicklung ist in der Praxis nicht allein die technischen Konstruktion wichtig, sondern auch das Management dieses Prozesses, sowie die Kommunikation zwischen den verschiedenen Stakeholdern [34].

2.2.2 Standards für die Architektur von Software

Für die Beschreibung und Dokumentation von Softwarearchitekturen nehmen Standards eine wichtige Rolle ein. Standards sind wesentlich für alle Architekturbausteine, welche Austauschbarkeit und einfach gestaltete Kommunikation zwischen Systemen bereitstellen sollen [34].

Als eine Grundlage für eine Softwarearchitekturbeschreibung eignet sich der Standard „Recommended Practice for Architectural Description of Software Intensive Systems“ (in der aktuellen Version ISO/IEC/IEEE 42010:2011 [1]) [34].

2.2.3 Prinzip: Graduated Flexibility

Das Prinzip der *Graduated Flexibility* gibt folgende Richtlinie vor. Die Struktur sowie detaillierte Daten sollen so dargestellt und implementiert werden, dass über die gesamte Lebenszeit einer Anwendung die am häufigsten anfallenden Änderungen am leichtesten durchzuführen sind, während weniger häufig benötigte Änderungen schwieriger umzusetzen sein können [14].

Dieses Prinzip ist auch auf die Systementwicklung und Softwareentwicklung anwendbar. Dort führt es unter anderem zu besserer Wartbarkeit durch im Durchschnitt geringere Änderungsaufwände.

2.2.4 Prinzip: Separation of Concerns

Separation of Concerns ist ein Prinzip für das Design von Systemen, bei dem dieses in verschiedene Bausteine aufgeteilt wird. Bausteine, die an der gleichen Aufgabe beteiligt sind werden gruppiert und von denen abgegrenzt, die für andere Aufgaben zuständig sind. Damit wird das System so strukturiert, dass zusammengehörige Bausteine in enger oder direkter Beziehung zueinander stehen.

Vorteile durch das Prinzip ergeben sich bei Entwicklung und Wartung der Systeme. Werden verschiedene Zuständigkeiten voneinander getrennt, so lassen sich diese unabhängig von einander ändern. Sind dagegen unter Vernachlässigung des Prinzips mehrere Zuständigkeiten in einer Komponente zusammengefasst, sind von einer Änderung alle Nutzer der Komponente betroffen. Dies gilt auch, wenn diese zur geänderten Zuständigkeit überhaupt keinen Bezug besitzen.

Die Kapselung von Informationen bei der Verwendung von Schichten-Architekturen ist eine Umsetzungen dieses Prinzips.

2.3 Aufbau von bedeutsamen Softwarearchitekturen

Hier folgen Ausführungen zu wichtigen, bereits etablierten Softwarearchitekturen.

2.3.1 Drei-Schichten-Architektur

Die Drei-Schichten-Architektur ist eine weit verbreitete Architektur für Webanwendungen. Nach ihr wird das System in folgende drei Teile aufgeteilt [10]:

- Präsentations-Schicht
- Anwendungslogik-Schicht
- Datenpersistenz-Schicht

Die Präsentations-Schicht besteht aus der grafischen Benutzeroberfläche und den damit zusammenhängenden Komponenten [10]. In dieser Schicht werden die Daten, beispielsweise im Browser, angezeigt [10]. Die Anwendungslogik-Schicht setzt die fachspezifischen Geschäftsprozesse und -regeln um und verwaltet die Verarbeitungsdienste für das Modell [10]. Die Datenpersistenz-Schicht umfasst alle dauerhaften Datenspeicher- und Datenzugriffsdienste [10]. Zu diesen gehören beispielsweise Datenbanken, XML-Dokumente oder ganz allgemein alle Formen von Dateien [10]. Außerdem beinhalten sie auch Klassenbibliotheken, welche dazu verwendet werden Daten abzufragen wie z.B. JDBC (Java Database Connectivity) [10].

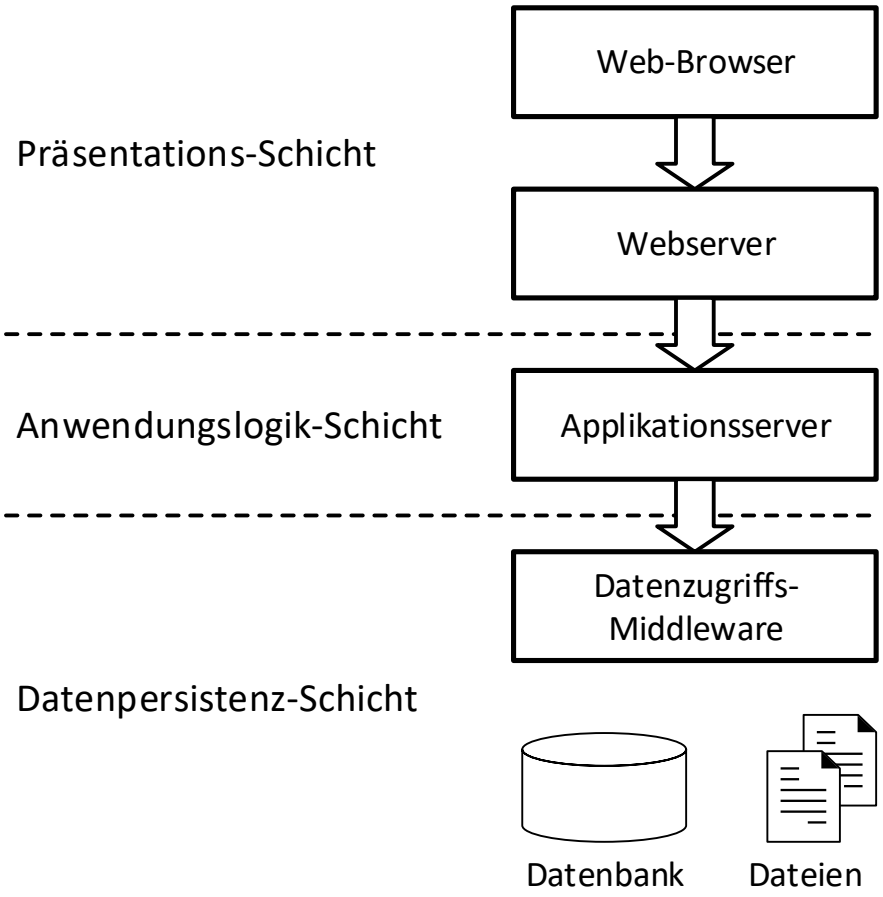


Abbildung 2.1: Drei-Schichten-Architektur (nach [10])

Abbildung 2.1 zeigt die Drei-Schichten-Architektur. Die Pfeile geben dabei die Aufrufrichtung an. Die Schichten greifen nur nach ihrer Anordnung und nichtzyklisch aufeinander zu, was zu einer besseren Nachvollziehbarkeit der Abhängigkeiten und zu einer einfacheren Modifizierbarkeit der betroffenen Elemente führt. Die Drei-Schichten-Architektur unterstützt damit unmittelbar das Prinzip *Separation of Concerns*.

2.3.2 Model-View-Controller-Entwurfsmuster

Das Model-View-Controller-Entwurfsmuster gewährleistet die Trennung von Benutzeroberfläche, Steuerung und Daten. Damit unterstützt es das Prinzip *Seperation of Concerns*. Die Architektur teilt die Zuständigkeiten des Systems in drei Einheiten auf [27]:

- Das **Modell** (engl. model)
- Die **Präsentation** (engl. view)
- Die **Steuerung** (engl. controller)

Die Präsentation dient der Darstellung von Informationen für BenutzerInnen. Sie repräsentiert das Design und den Design-Code der Darstellung [10]. Die Steuerung nimmt die Eingaben

der BenutzerInnen entgegen. Sie repräsentiert den Code für die Eingaben und die Navigation [10]. Das Modell beinhaltet und verarbeitet die Daten der Anwendung. Es repräsentiert die Geschäftslogik, den Datenbank-Code usw. [10]. Die Informationen der Präsentation werden den BenutzerInnen gezeigt und diese haben die Möglichkeit, Eingaben an die Steuerung zu senden. Die Steuerung speichert Eingaben im Modell und wählt die Art der Präsentation aus, welche den BenutzerInnen übermittelt werden soll. Die Daten des Modells werden in der Präsentation angezeigt. Konzeptionell ergeben sich die in Abbildung 2.2 zu sehenden Elemente und deren Zusammenhänge:

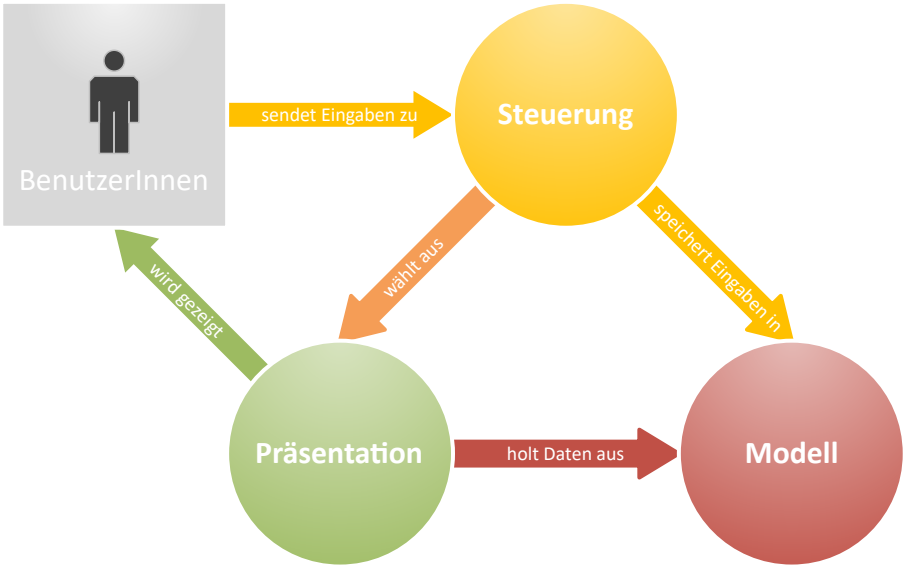


Abbildung 2.2: Model-View-Controller-Entwurfsmuster

Die Kreise stehen für die drei Einheiten des Model-View-Controller-Entwurfsmusters. Das Quadrat steht für BenutzerInnen des Systems. Die Pfeile stellen wichtige Beziehungen dar. Dabei ist die Art der Information durch ihre Farbe angezeigt, deren Bedeutung sich nach den Einfärbungen der drei Einheiten richtet. Die Farbe Orange steht zudem für einen der Steuerung zuzuordnenden Kontrollfluss.

Das Model-View-Controller-Entwurfsmuster kann in der Praxis unterschiedlich umgesetzt werden. Durch seine Trennung vereinfacht es die Wiederverwendung von Komponenten, reduziert die Komplexität bei der Entwicklung von webbasierten Anwendungen und erhöht die Wartbarkeit [27].

2.3.3 Grundlegende Architektur webbasierter Systeme

Abhängig davon wo die Berechnungen erfolgen, können Web-Technologien folgendermaßen klassifiziert werden [6]:

- Serverseitige Berechnung
- Clientseitige Berechnung
- Eine verteilte Berechnung, z.B. durch verteilt liegende Komponenten

Eine serverseitige Berechnung vereinfacht dabei einen plattformunabhängigen und universellen Zugriff auf entscheidungsunterstützende Anwendungen [7].

Die Web-Technologien stellen effektive Mittel zur Verfügung, um Funktionalitäten zur Entscheidungsunterstützung bereitzustellen und gemeinsam nutzbar zu machen [4].

2.3.4 Web-Services

Web-Services sind elektronische Dienste, welche über das World Wide Web zur Verfügung gestellt werden. Webbasierte Entscheidungsunterstützungssysteme können Web-Services in der Anwendungslogik-Schicht einsetzen [10].

Web-Services können zeitgleich und unabhängig voneinander entwickelt werden [10]. Web-Services bieten außerdem ein sprach- und umgebungsneutrales Programmiermodell, welches die Integration der Anwendungen innerhalb und außerhalb des Entscheidungsunterstützungssystems beschleunigt [10].

Eine Beschreibung einer möglichen Architektur zur Umsetzung von Web-Services findet sich im Anhang in Unterabschnitt A.2.1.

2.3.5 Serviceorientierte Architekturen

Serviceorientierten Architekturen nutzen das Konzept der Dienste, um Funktionalitäten modular anbieten zu können. Bei der Art der genutzten Dienste wird dabei zur Vereinfachung im Folgenden von den typischerweise verwendeten Web-Services ausgegangen.
Dadurch können Probleme gelöst werden, indem mehrere Modellierungs- oder Lösungsparadigmen aus verschiedenen Quellen als separate Web-Services kombiniert werden [9]. Die Ergebnisse können den NutzerInnen dann in zusammengefasster Form präsentiert werden [9].

Ein großer Vorteil der serviceorientierten Architekturen ist die einfachere Wiederverwendbarkeit und Modifikation von Komponenten durch die Aufteilung der Gesamtfunktion in einzelne Funktionalitäten, soweit sinnvoll [9]. Außerdem wird bei Bedarf das Anbieten dieser Funktionalitäten innerhalb eines Unternehmens oder zwischen verschiedenen Unternehmen erleichtert [9]. Zudem wird Plattformunabhängigkeit erreicht und die Integration von veralteten Systemen begünstigt [8].

Unter anderem kann dadurch die Pflege des Wissens erleichtert werden [30]. Es können Kosten reduziert werden und auch die Agilität kann erhöht werden [30].

Während immer noch Uneinigkeit über die genaue Definition einer Serviceorientierten Architektur besteht, existiert grundsätzlich ein Konsens über deren Schlüsseleigenschaften.

Diese Schlüsseleigenschaften sind folgende [21]:

- Nutzung geschäftsorientierter Web-Services
- Nachrichtenbasierte Interaktionen mit Implementierungen von Web-Services
- Kommunikation über ein Netzwerk
- Plattformneutralität
- Semantische Beschreibung der Web-Services und die Möglichkeit zum Auffinden von Web-Services
- Lose Kopplung von Systemkomponenten

Eine serviceorientierte Architektur setzt die Verwendung von Web-Services voraus [21]. Die Verwendung von Web-Services führt jedoch nicht notwendigerweise zu einer Serviceorientierten Architektur [21]. Es handelt sich nicht um eine serviceorientierte Architektur, wenn nicht alle verwendeten Web-Services den Schlüsseleigenschaften einer Serviceorientierten Architektur genügen [21].

2.4 Anforderungen an Softwaresysteme

Anforderungen sind zentrale Voraussetzungen für die Entwicklung bedarfsgerechter Softwaresysteme.

2.4.1 Definition von Anforderungen

Anforderungen bilden die Grundlage für den Entwurf von Softwaresystemen. Mit ihnen kann das Erreichen gesetzter Ziele überprüft werden. Um einzelne Anforderungen erfüllen zu können, müssen Entscheidungen über die Verwendung von Architekturen und Entwurfsmustern gefällt werden.

Anforderungen beschreiben die Merkmale eines Softwaresystems, die dieses haben muss, um den gewünschten Nutzen für AnwenderInnen zu erbringen. Architekturen lassen sich dabei nicht generell beurteilen. Sie können nur in Zusammenhang mit spezifizierten Anforderungen und damit mit den Bedürfnissen und Zielen der Stakeholder bewertet werden. Die Bewertung von Architekturen deckt also auf, ob diese die Erfüllung der Ziele der Stakeholder angemessen unterstützen und bildet eine Grundlage für bewusste Entscheidungen bezüglich des Aufbaus von Softwaresystemen.

Üblicherweise wird aus Anforderungen die Architektur spezifiziert. Diese wird dann implementiert und soweit notwendig, kontinuierlich angepasst. Dabei muss dauerhaft sichergestellt werden, dass die definierten Anforderungen vom Anwendungssystem erfüllt werden. Dies kann durch eine Architekturbewertung überprüft werden.

Architekturentscheidungen werden, obwohl sie von erhebliche Tragweite sind, in der Praxis häufig auf Basis vager Annahmen getroffen. Dies ist jedoch nicht zu empfehlen, was im Folgenden begründet ist. Außerdem beinhaltet die Spezifikation der Softwarearchitektur viele Designentscheidungen, welche in einer späteren Phase nur schwer geändert werden können [36]. Aus diesem Grund kommt einem frühen Treffen solcher Entscheidungen große Bedeutung zu.

2.4.2 Einteilung von Anforderungen

Prinzipiell werden zwei Arten von Anforderungen unterschieden:

- Funktionale Anforderungen, welche beschreiben was ein System leisten soll, also welche Funktionen es bereitstellen soll.
- Nichtfunktionale Anforderungen (Qualitätsanforderungen), welche beschreiben wie gut ein System etwas leisten soll, also in welcher Qualität es etwas bereitstellen soll.

Im Rahmen des Standards „Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE“ (in der aktuellen Version ISO/IEC 25000:2014 [20]) wurden unter anderem folgende Typen von nichtfunktionalen Anforderungen identifiziert: Performanz, Funktionalität, Usability, Portabilität und Sicherheit.

Nichtfunktionale Anforderungen werden häufig vernachlässigt. Sie sind den Stakeholdern oft unbekannt und sehr schwer zu quantifizieren. Auch müssen nichtfunktionale Anforderungen systematisch erhoben und dokumentiert werden. Architekturbewertungsverfahren bieten sich an, um zu prüfen, ob die Architektur die nichtfunktionale Anforderungen angemessen unterstützt.

2.4.3 Architekturbewertungsverfahren

Die Qualitätseigenschaften eines Anwendungssystems beeinflussen entscheidend dessen Softwarearchitektur und umgekehrt. Das Ziel von Architekturbewertungsverfahren ist eine Prüfung des Erfüllungsgrades der geforderten Qualitätseigenschaften durch eine Softwarearchitektur. Um Softwarearchitekturen bewerten zu können, müssen die Zusammenhänge zwischen den Entwurfsentscheidungen und den Qualitätseigenschaften systematisch analysiert werden.

Eine Bewertung setzt eine möglichst vollständige Definition der Qualitätseigenschaften voraus. Ist diese nicht gegeben, müssen Qualitätseigenschaften nacherfasst werden. Außerdem hängt die Zweckmäßigkeit einer Architektur von der vorausgesagten Nutzung und Weiterentwicklung ab. Wird ein System anders genutzt als erwartet, sind die Aussagen der Architekturbewertung nicht zuverlässig.

2.4.4 Ableitungen von Anforderungen und Softwarearchitekturen

Quellen für Anforderungen an diese können unter anderem Dokumente, mündlich mitgeteilte Informationen oder Ausschnitte vom Quellcode sein.

Ist bereits Quellcode vorhanden, so können die verwendete Softwarearchitektur und Entwurfsmuster aus diesem hergeleitet werden. Dies ist insbesondere notwendig, falls diese Architekturen nicht schon in ausreichendem Maße und ausreichender Qualität dokumentiert vorliegen.

Die Rückgewinnung von Architekturen aus bestehendem Quellcode fällt unter den Begriff Reverse-Engineering.

2.5 Vorstellung des SNIK-Projektes und des CIO-Navigators

Folgende Beschreibung stellt das SNIK-Projekt und die Idee des CIO-Navigators vor [2]:

Ziel des SNIK-Projektes ist ein semantisches Netz des integrierten Informationsmanagements im Krankenhaus (SNIK). Das semantische Netz soll unter anderem als Basis für die Entwicklung des Werkzeugs CIO-Navigator (CION) genutzt werden, das den Chief Information Officer im Rahmen des Informationsmanagements im Krankenhaus unterstützt. Zur Entscheidungsunterstützung soll der CIO-Navigator Informationen aus verschiedenen Systemen aggregieren und vernetzt in Form eines Dashboards darstellen.

Ein Prototyp des CIO-Navigators wird an einem Universitätsklinikum evaluiert.

3 Literaturrecherche

Die Literaturrecherche ist ein grundlegender Bestandteil dieser Arbeit und im folgenden detailliert beschrieben.

Zunächst werden aus der Forschungsfrage Suchbegriffe für die Literaturrecherche abgeleitet. Der daran anschließende Abschnitt beschreibt die Systematik und Durchführung der Literaturrecherche. Daraus hervorgehende Resultate, bei denen bestehende Ansätze eine wichtige Rolle einnehmen, werden anschließend präsentiert. Schließlich werden die Ergebnisse zusammengefasst und die Forschungsfrage wird auf deren Basis beantwortet.

3.1 Forschungsfrage

Das Hauptziel dieser Arbeit ist, eine Analyse der Architektur eines Entscheidungsunterstützungssystems durchzuführen, um Empfehlungen zur Optimierung dieser Architektur zu geben. Dafür müssen die Besonderheiten und typischen Komponenten von Entscheidungsunterstützungssystemen bekannt sein. Es ergibt sich daraus die folgende Forschungsfrage:

Welche Besonderheiten und welche typischen Komponenten von Entscheidungsunterstützungssystemen beeinflussen deren Softwarearchitektur?

3.2 Ableitung von Suchbegriffen für die Literaturrecherche

Für die Durchführung der Literaturrecherche wurden, wie in Abbildung 3.1 zu sehen, aus grundlegenden Begriffen (in fetter Schrift und deutsch) die für die Suche verwendeten Ausdrücke (in normaler Schrift und englisch) abgeleitet. Dabei sind die beiden oben dargestellten Begriffe aus der Forschungsfrage entnommen. Erste, vorab mit diesen beiden Begriffen durchgeführte Suchen führten zu sehr allgemeinen Ergebnissen. Daher wurden die in der unteren Hälfte dargestellten hellgrau eingefärbten Begriffe und abgeleiteten Ausdrücke zusätzlich noch mit aufgenommen, um die Treffermengen einzuschränken.

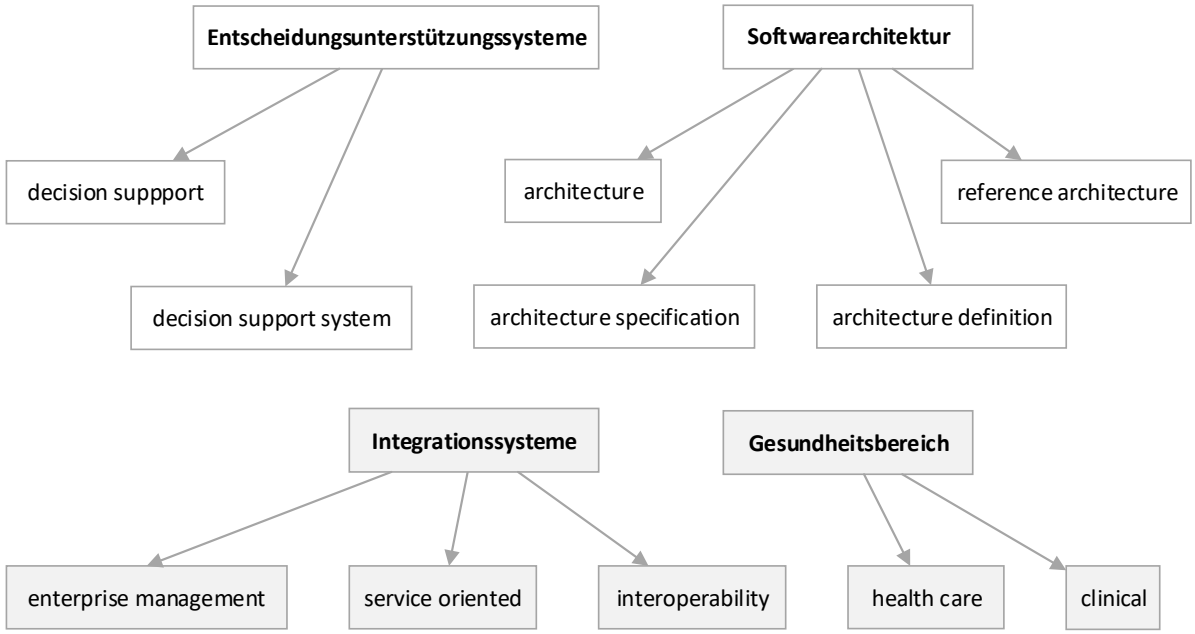


Abbildung 3.1: Abgeleitete Begriffe für die Literaturrecherche

3.3 Durchführung der Literaturrecherche

Dieser Abschnitt befasst sich mit der Systematik und dem Verlauf der Literaturrecherche. Zu jeder durchgeführten und dokumentierten Suche sind die relevanten Artikel angeben.

3.3.1 Einschränkende Kriterien für die Suche

Die Kriterien, welche die Auswahl bei der Suche einschränkten, sind in Tabelle 3.1 angegeben.

Tabelle 3.1: Einschränkende Kriterien für die Suche

ID	Kriterium
K1	Artikel muss vor Veröffentlichung einem Peer-Review unterzogen worden sein, um wissenschaftlichen Anforderungen zu genügen
K2	Artikel muss auf Deutsch oder Englisch geschrieben sein
K3	Artikel darf nicht älter als 10 Jahre sein (nicht vor 2006)
K4	Artikel muss zur Beantwortung der Forschungsfrage beitragen

3.3.2 Suchportale für wissenschaftliche Veröffentlichungen

Auf der Suche nach wissenschaftlichen Veröffentlichungen erwiesen sich die in Tabelle 3.2 aufgeführten Suchportale als die ergiebigsten Quellen und wurden daher für die Literaturrecherche verwendet:

Tabelle 3.2: Verwendete Suchportale für wissenschaftliche Veröffentlichungen

Komplette Bezeichnung	**Verwendete Kurzform**	**Web-Link des Suchportals**
Association for Computing Machinery	ACM	http://dl.acm.org
American Medical Informatics Association	AMIA	https://jamia.oxfordjournals.org
BioMed Central	BioMed	http://www.biomedcentral.com
Elsevier-Verlag	Elsevier	https://www.elsevier.com
Institute of Electrical and Electronics Engineers	IEEE	http://ieeexplore.ieee.org/Xplore/home.jsp
ScienceDirect-Portal	SD	http://www.sciencedirect.com
Springer-Verlag	Springer	http://link.springer.com

Der *Elsevier-Verlag* und der *Springer-Verlag* gehören zu den größten Verlagen für wissenschaftliche Inhalte weltweit und sind daher wichtige Quellen. Das *ScienceDirect*-Portal ist eine von Elsevier betriebene wissenschaftliche Online-Datenbank. Das *Institute of Electrical and Electronics Engineers (IEEE)* und das *Association for Computing Machinery (ACM)* sind die größten wissenschaftlichen Vereinigungen mit informationstechnologischer Ausrichtung. Bei zusätzlich medizinischer Ausrichtung sind die *American Medical Informatics Association* und die *BioMed Central* führende wissenschaftliche Organisationen.

3.3.3 Suchverlauf der Recherche

In Tabelle 3.3 sind die Ergebnisse der Recherche detailliert und geordnet nach dem zeitlichen Verlauf der Suche dargestellt. Auch das Umfeld und die Bedingungen der Suche werden dabei angegeben. Die Treffer wurden schrittweise analysiert. Dabei wurden Angaben zum Artikel oder Teile des Artikels untersucht, solange diese auf einen Beitrag des Artikels zur Beantwortung der Forschungsfrage hindeuteten. Sobald dieses nicht mehr der Fall war, wurde die Analyse für diesen Artikel abgebrochen. Dies geschah unter Verwendung der folgenden Reihenfolge: Titel, Schlüsselwörter, Zusammenfassung, Abbildungen, Einleitung, Schlussfolgerungen, kompletter Artikel. Die Auswahl der zitierten Veröffentlichungen erfolgte nach der so ausgeführten Durchsicht der Treffer der jeweiligen Suche.

Tabelle 3.3: Suchverlauf der Literaturrecherche

Suchdatum	Suchort	Exakte Suchanfrage	Suchbeschränkungen	Anzahl Treffer	Zitiert
22.03.2016	ACM	"query": keywords.author.keyword: (+software +architecture +reference +architecture) "filter": "publicationYear": "gte":2006 ,owners.owner=HOSTED	keywords. author.keyword publication year >= 2006	13	[36]
29.03.2016	ACM	"query": (+software +architecture +health +care) "filter": "publicationYear": "gte":2006 ,owners.owner=HOSTED	publication year >= 2006	117	[5], [33]
05.04.2016	ACM	"query": (+integration +system +architecture +enterprise +management +process) "filter": "publicationYear": "gte":2006 ,owners.owner=HOSTED	publication year >= 2006	115	[34]
15.04.2016	Springer	where the title contains: "clinical decision support" "service oriented"	where the title contains. publication year 2006-2016	19	[30]
18.04.2015	Springer	where the title contains: "decision support system" "interoperability"	where the title contains. publication year 2006-2016	41	[9]
22.04.2016	AMIA	Title: "decision support"	Title. Date of Publication: 2006-01-01 - 2017-01-01	83	[21]
13.05.2016	IEEE	((DSS OR "Decision Support" OR "Decision Support System") AND (architecture OR "architecture definition" OR "architecture specification"))	Year Range: 2006-2016	70	[4], [28], [31], [40]
28.05.2016	Elsevier	KEYWORDS("Decision Support Systems")	Publication Date After: 2006	52	[11]
28.05.2016	Springer	with the exact phrase "Medizinische Informationssysteme"	publication year 2006-2016	82	[18]
28.05.2016	ACM	"query": acmdlTitle:(+web-based +DSS) "filter": "publicationYear": "gte":2006, owners.owner=GUIDE	publication year >= 2006	45	[10]
28.05.2016	SD	pub-date > 2005 and TITLE(decision support) and TITLE(integrated architecture)	publication year > 2005	2	[35]
28.05.2016	BioMed	decision support system mvc		89	[27]
28.05.2016	SD	pub-date > 2005 and TITLE(web-based) and TITLE(decision support)	publication year > 2005	47	[7]

3.4 Ergebnisse der Literaturrecherche

Die Literaturrecherche führt zu den im Folgenden aufgeführten Ergebnissen bezüglich der Forschungsfrage mit Fokus auf Softwarearchitekturen und Entscheidungsunterstützungssystemen.

3.4.1 Bedeutung der Benutzerschnittstelle

Über eine Benutzerschnittstelle (UI) werden Komponenten der Anwendung (wie Entscheidungsmodelle oder entscheidungsrelevante Daten) den BenutzerInnen verfügbar gemacht. Von Druzdzel und Flynn werden über die Benutzerschnittstelle folgende Aussagen gemacht [11]:

Die interne Architektur eines Entscheidungsunterstützungssystems ist sehr wichtig. Der kritischste Aspekt von Entscheidungsunterstützungssystemen ist jedoch bei weitem ihre Benutzerschnittstelle. Systeme mit unhandlichen und unverständlichen Benutzerschnittstellen oder welchen, die ungewöhnliche Fähigkeiten voraussetzen, sind in der Praxis wenig nützlich und akzeptiert.

Des Weiteren werden NutzerInnen von Entscheidungsunterstützungssystemen deren Empfehlungen nicht durchgängig befolgen, wenn diese für sie nicht nachvollzogen werden können. Selbst wenn das Modell eine sehr gute Annäherung an die Wirklichkeit darstellt und dessen Empfehlungen korrekt sind, lehnen BenutzerInnen ohne umfassendes Verständnis eventuell den Vorschlag des Systems aus falschen Gründen ab. Das kann die Nützlichkeit des Systems einschränken oder gar negieren. Eine gute Benutzerschnittstelle sollte daher auch das Modell, auf welchem die Entscheidungsfindung des Systems aufbaut, für die BenutzerInnen transparent machen.

Obwohl numerische Berechnungen bei der Entscheidungsfindung eine wichtige Rolle spielen, ist das Nachvollziehen der Problemstruktur durch NutzerInnen sogar noch wichtiger. Wenn das System und sein Modell komplex sind, ist es für die EntscheidungsträgerInnen oft Verständnis fördernd zu begreifen, auf welche Weise die Systemvariablen zusammenhängen.

Gestaltung der Präsentation von Daten

Die Veröffentlichung von Liu und Liu beschäftigt sich in [28] mit Entscheidungsunterstützungssystemen im Kontext von Enterprise-Resource-Planning-Systemen. Die Autoren zeigen, dass Daten verständlich, akkurat und in einem geeigneten Format gespeichert den NutzerInnen dargestellt werden müssen.

Für die Präsentation der Daten kann daraus geschlossen werden, dass auf Verständlichkeit und Genauigkeit der Daten großer Wert gelegt werden sollte.

3.4.2 Grundlegende Architektur von Entscheidungsunterstützungssystemen

In der Entwicklung von Entscheidungsunterstützungssystemen seit den 1980er Jahren sind nach einer Veröffentlichung [40] bereits mehr als 20 Methoden für die Implementierung von

Entscheidungsunterstützungssystemen zu unterscheiden. Dies liegt an dem Umstand, dass die Umsetzung von Entscheidungsunterstützungssystemen äußerst abhängig vom jeweiligen Projekt ist.

Bei zahlreichen Architekturen im Gesundheitsbereich sind unter anderem die Verfügbarkeit, die Datenqualität, die Datenmodellierung, die Qualität der Funktionalität sowie eine einfache Bedienbarkeit (besonders bei der Eingabe von Daten) von großer Bedeutung [19].

3.4.3 Webbasierte Entscheidungsunterstützungssysteme

Bhargava, Power und Sun machen folgende Feststellungen für webbasierte Entscheidungsunterstützungssysteme [7]:

Heute ist das World Wide Web die Plattform der Wahl, um Entscheidungsunterstützungssysteme umzusetzen. Die Web-Technologien haben Auswirkungen auf Design, Entwicklung, Implementierung und Deployment von Entscheidungsunterstützungssystemen.

Webbasierte Entscheidungsunterstützungssysteme können auf verschiedene Arten konzipiert werden [6]. So können beispielsweise die gesamten Entscheidungsmodelle, Algorithmen, Dokumente und Daten auf einem einzelnen Web-Server verfügbar sein, auf den über einen Web-Browser zugegriffen wird. Alternativ kann das Entscheidungsunterstützungssystem auch Daten aus verschiedenen Quellen kombinieren, um eine anwendungsspezifische Lösung zu unterstützen.

Fast alle Implementierungen nutzen Web-Browser zur Darstellung von Benutzerschnittstellen. Ein oder mehrere Server führen dann die Abfragen an Datenbanken und Modelle aus. Dieser Ansatz ist jedoch mit den Beschränkungen verbunden, die sich aus der Verwendung des Web-Browsers als Benutzerschnittstelle, der Zustandslosigkeit des HTTP-Protokolls, der Round-Trip-Netzwerkverzögerungen bei Interaktionen und der „Pull“-Natur in traditionellen webbasierten und client-server-basierten Berechnungen ergeben.

Um ein webbasiertes Entscheidungsunterstützungssystem bereit zu stellen, bietet es sich der einfachen Umsetzung wegen an, ein vorhandenes System ohne große Anpassungen den beteiligten Stakeholdern durch einen Web-Browser verfügbar zu machen [10]. Dies liefert jedoch in der Regel unbefriedigende Resultate, da das System nicht von Beginn an darauf ausgelegt ist [10].

3.4.4 Serviceorientierte Architekturen bei Entscheidungsunterstützungssystemen

Insbesondere bei modellbasierten Entscheidungsunterstützungssystemen werden serviceorientierte Architekturen eingesetzt [9]. Durch den damit verbundenen erhöhten Entwicklungsaufwand zur Umsetzung und Einhaltung der Architektur, stellen sich serviceorientierte Architekturen als ungeeignet für die Verwendung im CIO-Navigator heraus.

3.4.5 Performanzeigenschaften von Entscheidungsunterstützungssystemen

Es ist wichtig zu berücksichtigen, dass verschieden ausgerichtete Entscheidungsunterstützungssysteme auch verschiedene Performanzfaktoren besitzen [4]. Beispielsweise werden in einem Entscheidungsunterstützungssystem im Umfeld eines Krankenhauses die Performanzfaktoren meistens nicht die gleichen sein, wie in einem in der Industrie verwendeten Entscheidungsunterstützungssystem [4].

Die Komponenten, mit denen ein Entscheidungsunterstützungssystem realisiert ist, können unterschiedlich verteilt sein. Je nach Grad dieser Verteilung von nicht verteilt bis hochgradig verteilt kann die Latenz durch Netzwerkzugriffe für die Performanz ausschlaggebend sein. Bewiesenermaßen ist die Netzwerk-Latenz eine der bedeutendsten Faktoren für die Performanz von webbasierten und dadurch verteilten Entscheidungsunterstützungssystemen [4], [30].

3.4.6 Integration von Quelldaten

Entscheidungsunterstützungssysteme benötigen Daten, die eventuell aus unterschiedlichen Quellen integriert werden müssen [28]. Der Mangel an adäquaten Daten oder Probleme bei deren Integration können den Erfolg von Entscheidungsunterstützungssystemen behindern [28].

Die Integration von Quelldaten in einem Informationssystem kann aufgrund der möglichen Verteilung und heterogenen Struktur dieser Daten verhältnismäßig komplex sein. Konsistente Entscheidungen benötigen konsistente Informationen [29]. Daher müssen verteilte Daten eventuell in konsistente und in Zusammenhang mit der Entscheidungsfindung verständliche Daten umgewandelt werden [28]. Dies kann beispielsweise durch die Verwendung eines Integrationsservers umgesetzt werden.

3.4.7 Veröffentlichte Architekturen

Komponenten von Entscheidungsunterstützungssystemen nach Druzdzel und Flynn

Die im Folgenden genannten drei Bestandteile können in vielen Architekturen von Entscheidungsunterstützungssystemen in der einen oder anderen Ausgestaltung aufgefunden werden. Nach Druzdzel und Flynn [11] bestehen Entscheidungsunterstützungssysteme aus drei grundlegende Systemkomponenten:

- Datenbank-Management-System (DBMS)
  Die Datenmanagement-Komponente speichert Informationen, beispielsweise aus den operativen Datenquellen einer Organisation oder externen Quellen.
- Modell-Basis-Management-System (MBMS)
  Die Modellmanagement-Komponente befasst sich mit der Repräsentation von Ereignissen, Fakten oder Situationen unter Verwendung einer Auswahl aus vielen möglichen Modellen.
- Dialog-Generierungs- und Management-System (DGMS)
  Die Schnittstellenmanagement-Komponente erlaubt BenutzerInnen mit dem System zu interagieren.

Diese Struktur entspricht der schon eingeführten Drei-Schichten-Architektur, die hier eine konkrete Ausprägung erfährt.

Fundamentale Systemarchitektur nach Mohktar et al.

Eine Veröffentlichung von Mohktar et al. zu einem medizinischen Entscheidungsunterstützungssystem [31] präsentiert ein Software-Framework bestehend aus drei Komponenten:

- Datenzugangs-Schicht, welche das Entscheidungsunterstützungssystem mit einer Datenbank verbindet
- Geschäftslogik-Schicht, welche die Geschäftsaktivitäten umfasst, also die Vorgänge, welche durchgeführt werden müssen um Aufgaben unter Verwendung einer Wissensbasis auszuführen
- Frontend-Schicht, welche die Benutzerschnittstelle bereit stellt und damit die Kommunikation mit BenutzerInnen ermöglicht

Auch diese Struktur ist eine Ausprägung der schon eingeführten Drei-Schichten-Architektur, jedoch wird hier die nicht zur Geschäftslogik gehörende Anwendungslogik keiner der Schichten zugeordnet.

Alle drei Komponenten des Entscheidungsunterstützungssystems können auf einem Server laufen.

Die Datenzugangs-Schicht ermöglicht es dem System, Daten in der Datenbank zu erstellen, zu lesen, zu aktualisieren oder zu löschen. Diese Schicht verbindet die weiteren Schichten der Architektur mit der Datenbank. Die Geschäftslogik-Schicht besteht aus dem folgendem generischen Framework, welches in jedem beliebigen Entscheidungsunterstützungssystem eingesetzt werden kann:

Mit Hilfe einer Prozessmanagementkomponente innerhalb des Entscheidungsunterstützungssystems werden die Entscheidungsunterstützungsprozesse verwaltet. Diese bestehen hierbei aus einer Folge von Aufgaben, die in einer definierten Reihenfolge ausgeführt werden. Sie beinhalten Komponenten wie das Ziel, welches erreicht werden soll, Aktivitäten, die in einer bestimmten Reihenfolge ausgeführt werden sollen und Ausgaben, welche für die BenutzerInnen erstellt werden sollen. Diese Ausgaben werden BenutzerInnen in Form von Entscheidungsempfehlungen oder Alarmen vermittelt. Das System verfolgt einen regelbasierten Ansatz, um bei der Entscheidungsfindung im medizinischen Bereich zu helfen. Die Entwicklung der Regeln für die Entscheidungsunterstützung ist unabhängig vom Design des Entscheidungsunterstützungssystems gehalten.

In Abbildung 3.2 ist eine grafische Übersicht über die hier beschriebene Systemarchitektur gegeben:

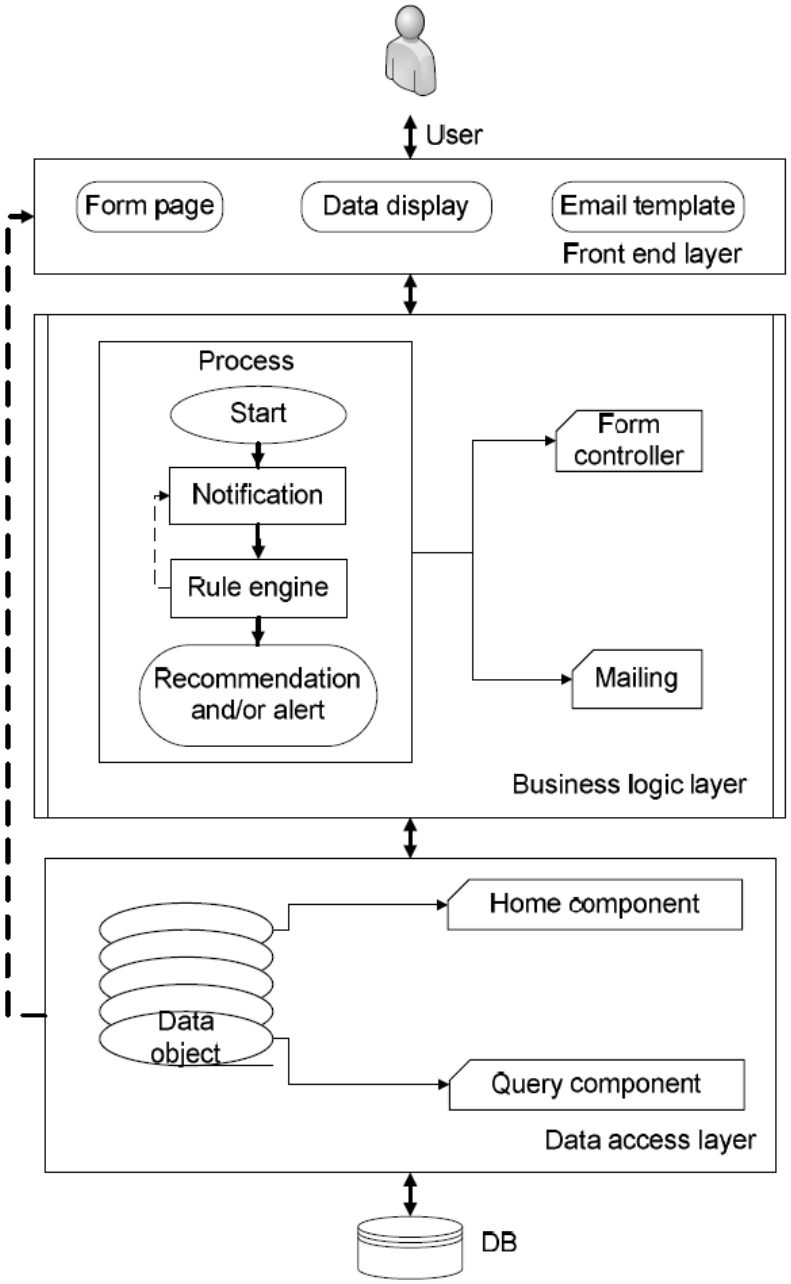


Abbildung 3.2: Architektur des Entscheidungsunterstützungssystems nach Mohktar et al. (Quelle: [31])

Eine Beschreibung der konkreten Technologien, welche in der Veröffentlichung [31] zur Umsetzung der hier dargelegten Architektur eingesetzt werden, findet sich im Anhang in Unterabschnitt A.2.2.

Fundamentale Systemarchitektur nach Boza et al.

Ein von Boza et al. veröffentlichtes Framework für Entscheidungsunterstützungssysteme [9] wird folgendermaßen beschrieben:

Mit dem Framework wird eine mögliche Architektur für Entscheidungsunterstützungssysteme gegeben. Diese sollen dabei den Entscheidungsfindungsprozess innerhalb von Unternehmen und zwischen Unternehmen optimieren. Das Framework ist webbasiert und nutzt eine service-orientierte Architektur (SOA) mit Web-Services. Der Einsatz sollte schon in den frühen Phasen des Designs eines Entscheidungsunterstützungssystems berücksichtigt werden.

Die Entscheidungsmodellierung (Decision Modeling) für den Entscheidungsfindungsprozess, die Datenmodellierung (Data Modeling) für die zugrunde liegenden Daten und die Modellanalyse und -untersuchung (Model Analysis and Investigation) in welcher die Verarbeitung stattfindet, sind wichtige logische Konstrukte. Diese spielen eine große Rolle sowohl bei der Interaktion von Informationssystemen und Entscheidungstechnologien als auch bei der rationalen Entscheidungsfindung. Die folgende Abbildung ordnet die Bestandteile des Entscheidungsunterstützungssystems diesen drei Kategorien zu.

In Abbildung 3.3 ist eine grafische Übersicht über diese Systemarchitektur gegeben:

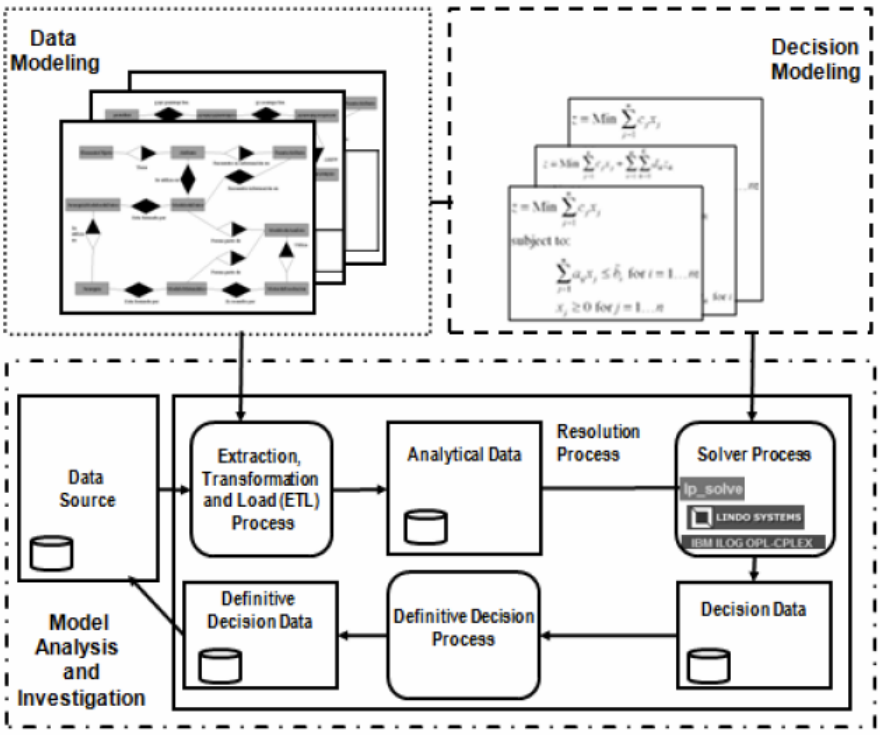


Abbildung 3.3: Unterscheidung von Entscheidungsmodellierung, Datenmodellierung, sowie Modellanalyse und Modelluntersuchung (Quelle: [9])

Integration von Quelldaten nach Lin et al.

Das von Lin et al. in [27] veröffentlichte Framework erreicht eine Integration von Quelldaten auf die im Folgenden erläuterte Weise.

Die in verschiedenen Systemen gespeicherten Daten werden in einer virtuellen, zentralen Datenbank integriert, indem Daten aus mehreren Datenbanken aggregiert werden. Diese Systemarchitektur verbirgt die tatsächliche Verteilung und Heterogenität der einzelnen Datenbanken.

Eine Beschreibung der konkreten Technologien, welche in der Veröffentlichung [27] zur Umsetzung der hier dargelegten Architektur eingesetzt werden, findet sich im Anhang in Unterabschnitt A.2.3.

Einordnung und Vergleich

Die Architekturen von Druzdzel und Flynn, sowie Mohktar et al. bauen auf der Drei-Schichten-Architektur auf. Mohktar et al., sowie Boza et al. stellen im Speziellen eine webbasierte Benutzeroberfläche zur Verfügung.

Mohktar et al., sowie Boza et al. verwenden für die Erstellung einer Entscheidungsempfehlung eine Entscheidungsmodellierung. Weitere, bei Entscheidungsunterstützungssystemen spezifische Komponenten finden sich für die Aufbereitung von Daten, auf deren Grundlage die Entscheidungen getroffen werden sowie bei der Koordination der Prozesse zum Erstellen einer Empfehlung.

3.5 Beantwortung der Forschungsfrage

Die in Abschnitt 3.1 beschriebene Forschungsfrage kann wie folgt beantwortet werden:

Die Ausgestaltung von Entscheidungsunterstützungssystemen ist hochgradig abhängig vom jeweiligen Projekt in dessen Umfeld diese umgesetzt werden sollen.

Das World Wide Web ist inzwischen die Plattform der Wahl um Entscheidungsunterstützungssysteme zu realisieren. Die Web-Technologien haben große Auswirkungen auf Design, Entwicklung, Implementierung und Deployment von Entscheidungsunterstützungssystemen. Üblicherweise wird ein Web-Browser genutzt, um die Benutzerschnittstelle bereitzustellen.

Das Entscheidungsunterstützungssystem nach Boza et al. basiert auf einer serviceorientierten Architektur mit dem Einsatz von Web-Services.

Webbasierte Entscheidungsunterstützungssysteme weisen immer eine Verteilung auf, wodurch Netzwerk-Latenzen zu einem kritischen Faktor werden.

Bei manchen Entscheidungsunterstützungssystemen, insbesondere bei datenbasierten bzw. dokumentenbasierten wie dem CIO-Navigator, müssen Daten in eine konsistente und im Rahmen der Entscheidung verständliche Form umgewandelt werden. Die Integration der Quelldaten kann dann sehr komplex werden.

Ein sehr wichtiges Element bei Entscheidungsunterstützungssystemen ist ihre Benutzerschnittstelle. Eine gute Benutzeroberfläche macht die Verwendung eines Entscheidungsunterstützungssystems für die NutzerInnen zugänglicher. NutzerInnen von Entscheidungsunterstützungssystemen werden außerdem deren Empfehlungen nicht durchgängig befolgen wollen, wenn diese für sie nicht nachvollziehbar sind. Insbesondere sollten die präsentierten Daten verständlich und akkurat sein. Aber auch die Qualität der Informationen und des zugrunde liegenden Softwaresystems sind wichtige Bestandteile eines Entscheidungsunterstützungssystems.

4 Systemarchitektur und Infrastruktur

In diesem Kapitel wird die Systemarchitektur des CIO-Navigators untersucht. Dazu gehört die Infrastruktur. Als Ausprägung einer Integrationsarchitektur soll der Integrationsserver *Teiid*[1] des *JBoss*-Projektes betrachtet werden.

4.1 Einbettung der Anwendung in das Umfeld

Die Analyse beginnt mit der Untersuchung der Systemarchitektur des CIO-Navigators (CION). Bei der Systemarchitektur wird anders als bei der Softwarearchitektur nicht ausschließlich der Aufbau der Softwarekomponenten eines Anwendungssystems betrachtet, sondern insbesondere das Zusammenspiel von Anwendungen mit ihrer Umgebung. Vor allem die Infrastruktur, welche benötigt wird damit die Anwendung und externe Komponenten laufen, wird dabei einbezogen.

Der CIO-Navigator ist eine Webanwendung. Als solche ist er wie im Folgenden beschrieben in sein Umfeld eingebettet. Die CIO-Navigator-Anwendung läuft innerhalb eines *Microsoft Internet Information Services (IIS)*-Servers. Dieser wiederum läuft auf einem *Microsoft Server 2012*-Betriebssystem. Dieses befindet sich aktuell in einer virtuelle Maschine auf einem Server der Universität Heidelberg als Entwicklungs- und Testumgebung, welcher dem Client über das Netzwerk zugänglich ist. Selbige virtuelle Maschine beherbergt auch die benötigte SharePoint-Installation. In dieser kann über einen Link der im *IIS*-Server liegende CIO-Navigator aufgerufen werden. Außerdem kann über diese SharePoint-Installation auf in SharePoint verfügbar gemachte Dokumente (wie Excel-Tabellen, PDF-Dateien oder Webseiten) zugegriffen werden.

Mit diesem Aufbau fügt sich der CIO-Navigator in die vorhandene SharePoint-Installation des Krankenhauses ein. Zudem kann der CIO-Navigator auf Ressourcen dieser SharePoint-Umgebung zugreifen, um benötigte Daten zu beschaffen.

Die Infrastruktur des CIO-Navigators ist mit einem *UML 2.5*[2]-konformen Komponentendiagramm in Abbildung 4.1 dargestellt.

[1] Webseite: http://teiid.jboss.org/
[2] Webseite: http://www.uml.org/

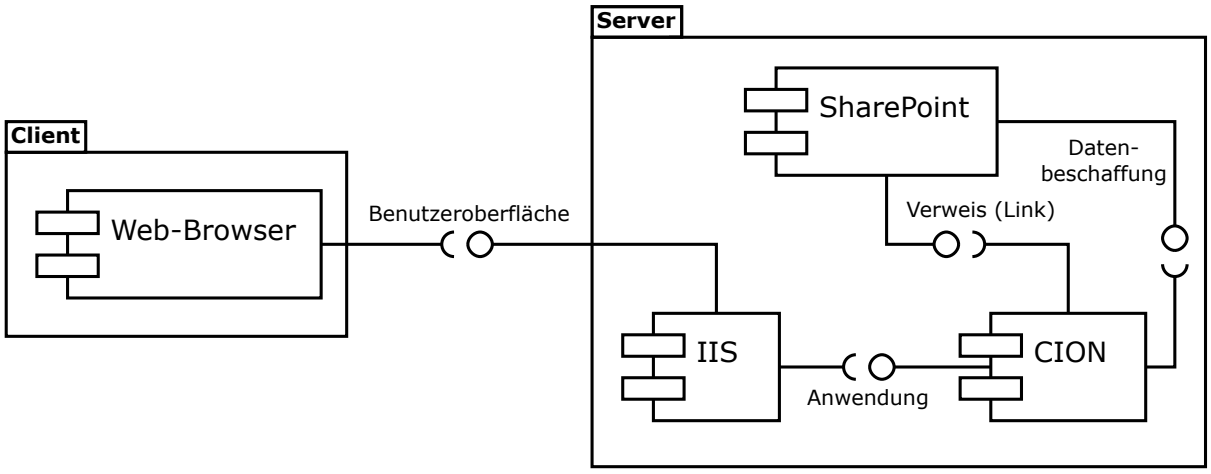


Abbildung 4.1: Schematische Darstellung der Infrastruktur des CIO-Navigators als UML-Komponentendiagramm

Beim Aufruf des CIO-Navigators finden folgende Abläufe statt. Die BenutzerIn ruft mit dem Web-Browser über den Link im SharePoint den CIO-Navigator auf. Dieser läd die benötigten Daten bei Bedarf über SharePoint und verarbeitet diese. Die BenutzerIn erhält im Web-Browser die Ergebnisse zurück. Abbildung 4.2 zeigt das zugehörige *UML 2.5*-konforme Sequenzdiagramm.

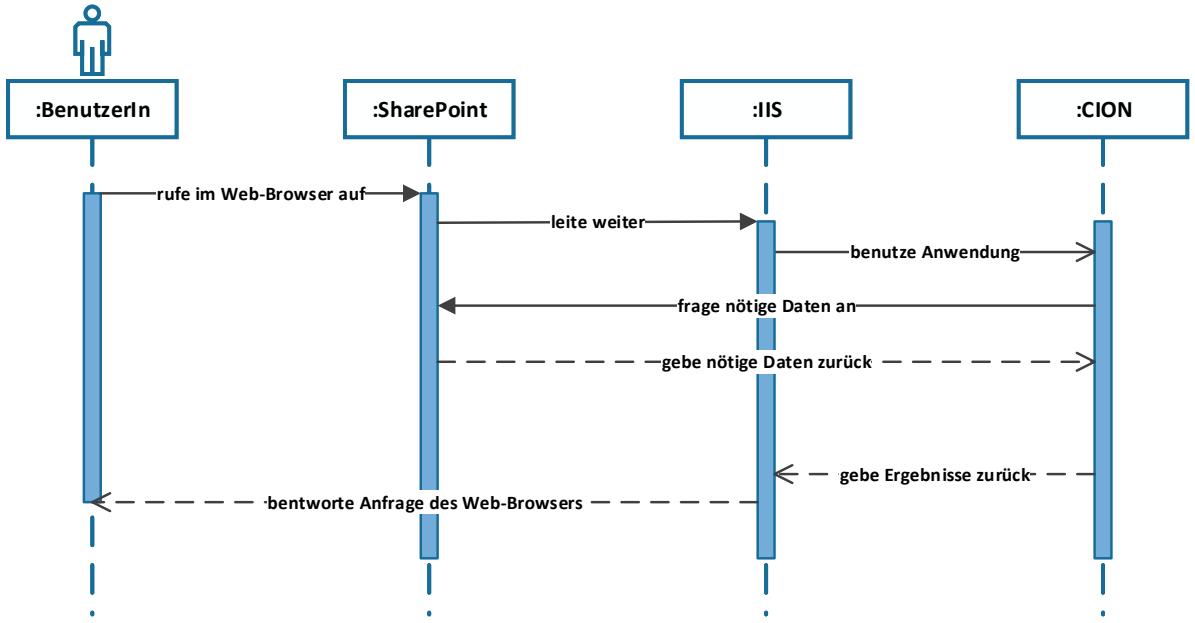


Abbildung 4.2: UML-Sequenzdiagramm für Anfragen an den CIO-Navigator

4.2 Eignung von Teiid für die Integration von Daten

Im Rahmen der Betrachtungen zur Integration von Daten für den CIO-Navigator wurde auch *Teiid* auf seine Eignung geprüft. *Teiid* ist ein System für Daten-Virtualisierung, mit dem es möglich ist auf Daten aus unterschiedlichen Quellen einheitlich zuzugreifen, ohne diese duplizieren zu

müssen. Es wurde für eine genauere Betrachtung ausgewählt, da es das am meisten verbreitete quelloffene System dieser Art ist. Es besitzt den in Abbildung 4.3 dargestellten Aufbau.

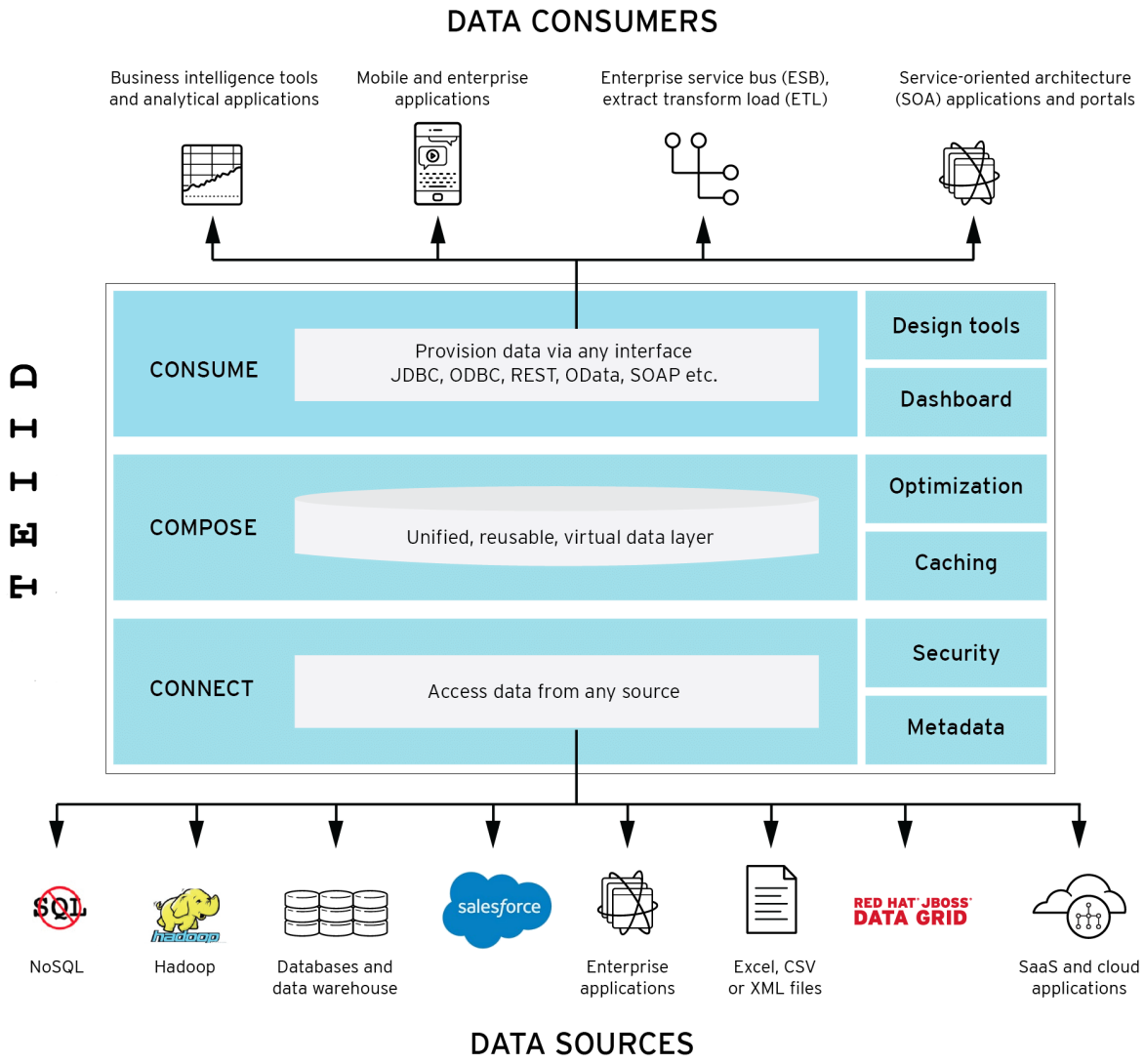


Abbildung 4.3: Aufbau von Teiid (Quelle: http://teiid.jboss.org/)

Die Abbildung zeigt, dass mit *Teiid* verschiedene Datenquellen integriert werden können (unterer Abschnitt der Darstellung), so auch Excel-Dateien oder CSV-Dateien. Auf die so zusammengestellten Daten kann dann mittels verschiedener Schnittstellen (oberer Abschnitt der Darstellung) zugegriffen werden, unter anderem über Datenbanktreiber oder Web-Technologien.

Die Verwendung von *Teiid* bietet somit Vorteile bei der flexiblen Integration von Daten aus unterschiedlichen Quellen. Nachteilig sind jedoch die mit der Einrichtung und der Aufrechterhaltung einer weiteren Infrastrukturkomponente verbundenen zusätzlichen Aufwände, sowie die zusätzlich in Anspruch genommenen Hardware-Ressourcen. Diese Begleitumstände führen dazu, dass sich der Einsatz von *Teiid* nur für die Integration von Daten komplexer und aufwändiger Vorhaben tatsächlich lohnt. Aufgrund dessen finden beim CIO-Navigator aufwand- und ressourcensparende Alternativen Verwendung.

5 Anforderungs- und Architekturanalyse mit Architekturbewertungsverfahren

An dieser Stelle wird zunächst ein geeignetes Architekturbewertungsverfahren ausgewählt. Es folgt die Durchführung dieses Verfahrens. In diesem Rahmen findet eine ausführliche Analyse der vorhandenen Softwarearchitektur, zunächst ohne Werkzeuge und danach mit *CodeMap* und *Codemetrik* aus *Microsoft Visual Studio Enterprise 2015*[1] und *CodeCity*[2], statt. Außerdem werden dabei unter anderem die Anforderungen und die zugehörigen Stakeholder ermittelt und eine Stakeholder-Map aufgestellt.

5.1 Auswahl des Architekturbewertungsverfahrens

Unter vielen möglichen Anforderungs- und Architekturanalyseverfahren (beispielsweise aus der Zusammenstellung in [12]) wurden die bewährten Architekturbewertungsverfahren *Software Architecture Analysis Method* (SAAM) [22] und *Architecture Tradeoff Analysis Method* (ATAM) [23] zur detaillierteren Prüfung ihrer Eignung ausgewählt. Beide Verfahren ermöglichen wichtige grundlegende Erkenntnisse und Einblicke, vor allem in die Qualitätseigenschaften von Architekturen.

5.1.1 Vergleich von SAAM und ATAM

SAAM wurde vom Software Engineering Institute (SEI) der Carnegie Mellon University entwickelt und analysiert Softwarearchitekturen in Bezug auf deren Qualitätseigenschaften. SAAM versucht die Vereinbarkeit der grundlegenden Architektur mit den geforderten Eigenschaften einer Anwendung zu überprüfen. Dabei hilft es außerdem Architekturrisiken abzuschätzen. Identifizierte Risiken rücken damit in den Fokus und können durch angemessene Maßnahmen verhindert oder abgeschwächt werden. Außerdem sollen bei Bedarf verschiedene Architekturansätze vergleichbar gemacht werden, indem diese durch das Verfahren bewertet und die Ergebnisse miteinander verglichen werden. Zusätzlich werden Anforderungen geklärt und miteinander abgeglichen. Die Dokumentation der Architektur wird dabei substanziell verbessert und ein intensiver Austausch zwischen den Stakeholdern ermöglicht.

[1] Webseite: https://www.visualstudio.com/vs/
[2] Webseite: https://wettel.github.io/codecity.html

ATAM wurde, aufbauend auf SAAM, ebenfalls vom SEI entwickelt. Die Zielsetzung der Methode ist dabei zu helfen, eine geeignete Architektur für ein Softwaresystem auszumachen indem sogenannte Risks, Sensitivity Points und Tradeoffs aufgedeckt werden. Das Klären von Anforderungen und der Austausch zwischen Stakeholdern nimmt bei dieser Methode einen noch höheren Stellenwert ein als beim SAAM-Verfahren.

Die Architekturbewertungsverfahren SAAM und ATAM haben vieles gemeinsam. Es handelt sich bei beiden verwandten Verfahren um szenariobasierte Architekturbewertungsverfahren (vergleiche Unterabschnitt 5.2.1, in dem Szenarien im Detail beschreiben werden) mit eher allgemeinem Fokus, welche einige unterschiedliche Ausrichtungen besitzen. Die Zielsetzung beider Verfahren ist nicht, einzelne Architekturen allgemein zu kritisieren oder zu empfehlen sondern herauszufinden, ob oder mit welchen Änderungen Architekturen die Bedürfnisse eines Systems unterstützen. Die Verfahren besitzen beide ein festgelegtes Vorgehensmodell, welches zu einer systematischen Architekturbewertung führt. Dabei nehmen das Erheben, Priorisieren und Bewerten von Szenarien eine zentrale Rolle ein. Jede Qualitätseigenschaft, die mit Szenarien erfasst werden kann, lässt sich betrachten. Außerdem werden die Wünsche der Stakeholder harmonisiert, was ein gemeinsames Verständnis für spätere Entscheidungen schafft. Tabelle 5.1 stellt wichtige Aspekte der Verfahren SAAM und ATAM gegenüber.

ATAM ist im Rahmen von größeren Softwareprojekten der vorherrschende Standard und ermöglicht Einblicke in Bestandteile der Architektur. Das ebenfalls bewährte SAAM liefert dagegen wertvolle Einsichten in Bezug auf die Gesamtstruktur von Architekturen.

Tabelle 5.1: Gegenüberstellung von SAAM und ATAM (nach [12])

Kriterium	**SAAM**	**ATAM**
Qualitätsmerkmale	Verschiedene (besonders Modifizierbarkeit)	Verschiedene
Berücksichtigung von Beziehungen zwischen Entwurfsmustern	Nein	Ja
Vorraussetzungen für die Anwendung	Keine besonderen Anforderungen; leicht anzuwendende Methode	Durch die Analyse und Befragungsphase hoher Ressourcenbedarf
Reifegrad / Validierung	Ausgereift; erstmals 1994 beschrieben; Anwendung in vielen Domänen	Sehr ausgereift; mittlerweile in 2. Version; Fallstudien und Trainings; validiert in vielen Projekten
Detaillierungsgrad der Prozessbeschreibung	Ausführlich; inklusive Fallstudie	Ausführlich; inklusive Fallstudie
Ziel der Methode	Tauglichkeit (Suitability), Risiko	Sensitivity, Tradeoff, Risiko
Projektphase	Früh	Früh
Bewertungsansatz	Szenarien	Utility-Baum, Szenarien
Beteiligung Stakeholder	Ja	Ja
Erfahrung und Kenntnisse des Evaluations-Teams	Leicht anzuwenden; kaum Erfahrung nötig	Hohe Anforderung an das Evaluations-Team wegen Aufstellen des Utility-Baums und Identifizierung der Architekturansätze

5.1.2 Entscheidung für SAAM

SAAM lässt sich im Vergleich zu ATAM bei kleineren Projekten einfacher handhaben. Bei SAAM steht außerdem die grundlegende Architektur im Vordergrund, im Gegensatz zu einzelnen Entwurfsmustern bei ATAM. Daher passt dieses Verfahren besser auf die Fragestellung dieser Arbeit. So liegt der Schwerpunkt bei SAAM auf der Prüfung, ob sich eine Architektur zur Erfüllung von Qualitätsanforderungen eignet und wo Überschneidungen bei der Abbildung von Szenarien auf die Architektur auftreten. Daher fällt hier die Wahl beim einzusetzenden Architekturbewertungsverfahren auf SAAM. Dieses wird im Folgenden durchgeführt.

Für die Durchführung des Verfahrens standen die am CIO-Navigator-Projekt beteiligten Stakeholder des Krankenhauses nur in eingeschränktem Umfang zur Verfügung. Das SAAM-Verfahren wird daher im Rahmen dieser Arbeit, wie in [22] ausdrücklich empfohlen, an entsprechenden Stellen angepasst.

5.2 Durchführung der Software Architecture Analysis Method (SAAM)

In diesem Abschnitt sind die Durchführung von SAAM und die dabei erzielten Ergebnisse beschrieben.

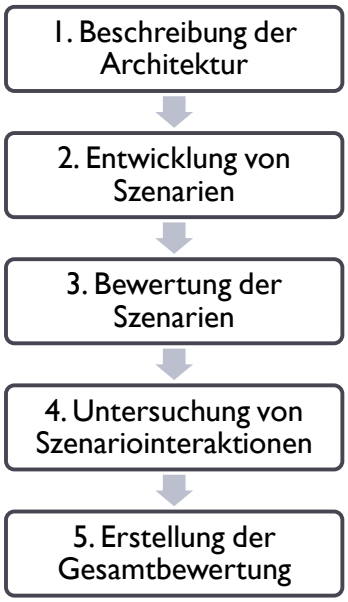
5.2.1 Szenarien bei SAAM

Um die Anforderungen an das untersuchte System zu konkretisieren werden bei SAAM Szenarien verwendet. Szenarien unterscheiden sich stark in der Breite und Ausrichtung. Hier werden Szenarien als eine kurze Beschreibung einer geplanten oder gewünschten Verwendung eines Systems genutzt. Diese umfassen bei SAAM üblicherweise lediglich einen Satz. Es sollen Szenarien für alle am Projekt beteiligten Stakeholder (beispielsweise Nutzer, Administrator oder System Designer) erstellt werden, um deren jeweilige Bedürfnisse einzubeziehen. Anschließend wird die Architektur daraufhin untersucht, wie gut oder einfach diese die erstellten Szenarien erfüllt.

Beispiel für ein Szenario im Bereich Wartbarkeit: *Hinzufügen eines neuen Feldes auf einer der Webseiten zum Anzeigen eines Wertes aus einer der zugrunde liegenden Excel-Tabellen.*

5.2.2 Schritte bei der Durchführung von SAAM

Die Durchführung von SAAM beinhaltet die in Abbildung 5.1 dargestellten Schritte:



Beschreibung der zu prüfenden Architektur mittels einer einfachen (nicht festgelegten) Notation

Entwicklung von textuellen Aufgabenszenarien, die das System jetzt oder zukünftig unterstützen muss

Voraussichtliches Ausmaß und Aufwendungen für durch Szenarien nötige Änderungen festhalten

Überschneidungen verschiedenartiger Szenarien bei Architekturkomponenten untersuchen

Gewichten der ausgewerteten Szenarien und Szenariointeraktionen nach relativer Wichtigkeit und Erstellung einer Einordnung (alle Stakeholder)

Abbildung 5.1: Schritte von SAAM

Der erste Schritt der Analyse stellt eine Beschreibung der zu prüfenden Architektur dar. Die dazu verwendete Notation ist dabei durch das Verfahren nicht festgelegt. Sie sollte jedoch nicht komplizierter ausfallen als nötig. Im zweiten Schritt werden die Szenarien entwickelt. Diese konkretisieren aktuelle oder geplante Anforderungen an das System in Form von Aufgaben an dieses. Diese Aufgaben sind üblicherweise mit einem Satz in Textform ausformuliert. Im dritten Schritt werden die zusammengestellten Szenarien bewertet. Dafür werden diese auf die beschriebene Architektur abgebildet. Ist ein Szenario mit dieser Architektur ohne Änderung durchführbar, so wird es als *direktes Szenario* bezeichnet. Ist es nur mit Änderungen an bestehenden Elementen oder Beziehungen oder durch Hinzufügen neuer Elemente oder Beziehungen realisierbar, so wird es als *indirektes Szenario* bezeichnet. Für die *indirekten Szenarien* werden die benötigten Aufwände, um diese zu ermöglichen, festgehalten. Die Merkmale nach denen alle Aufwände gemessen werden sind dabei nicht vorgegeben. Üblich sind die Anzahl der zu ändernden Komponenten und Verbindungen oder die Anzahl der zu ändernden Quellcodezeilen. Im vierten Schritt werden nun die Überschneidungen von *indirekten Szenarien* bei den Architekturelementen untersucht. Überschneiden sich verschiedenartige Szenarien bei den Elementen bei denen Änderungen nötig sind, so interagieren diese Szenarien. Dies wird als Szenariointeraktion bezeichnet. Da eine Architektur mit vielen Szenariointeraktionen auf eine schlechte Umsetzung des Prinzips *Separation of Concerns* hinweist, wird bei SAAM die Ausgestaltung der Architektur mit den wenigsten Szenariointeraktionen bevorzugt. Beim fünften Schritt schließlich werden die ausgewerteten Szenarien und Szenariointeraktionen nach ihrer verhältnismäßigen Bedeutung gewichtet und abschließend sortiert nach ihrer Wichtigkeit angeordnet. Dies ist ein subjektiver Prozess und wird unter Teilnahme möglichst aller Stakeholder durchgeführt.

5.2.3 Aktivitäten und Abhängigkeiten von SAAM

Bei Durchführung einzelner Schritte von SAAM können neue Informationen gewonnen werden, welche für vorherige Schritte Relevanz besitzen. Um diese Informationen zu nutzen, lässt das Verfahren ein Zurückspringen zu vorherigen Schritten zu. In Abbildung 5.1 sind Aktivitäten und Abhängigkeiten der Vorgänge des Verfahrens dargestellt.

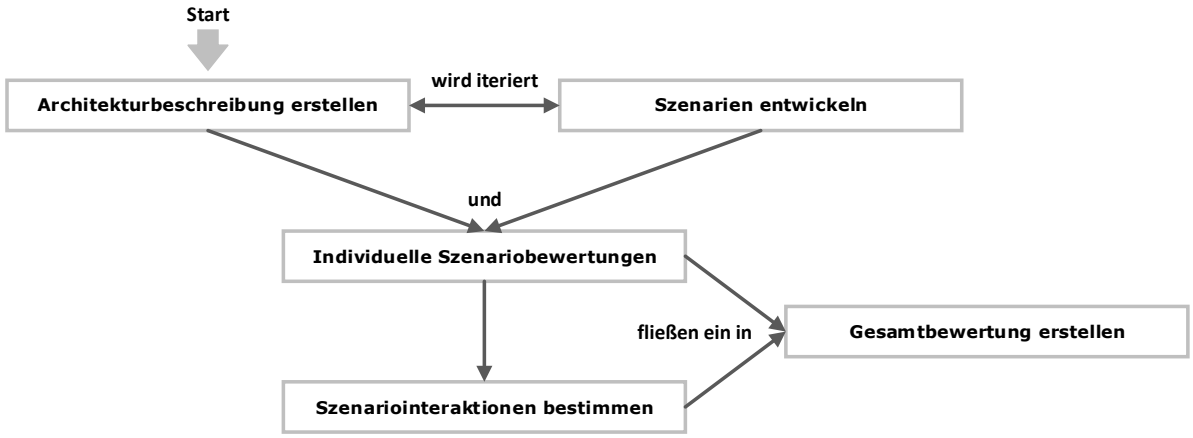


Abbildung 5.1: Aktivitäten und Abhängigkeiten in der szenariobasierten Analyse (nach [22])

Die Beschreibung der Architektur in Schritt 1 und die Entwicklung von Szenarien in Schritt 2 beeinflussen sich in der Regel gegenseitig. Diese können üblicherweise wechselseitig in mehreren Iterationen immer weiter verfeinert werden. Als Ergebnis der dabei stattfindenden Abbildung von Szenarien auf die Architektur lassen sich in Schritt 3 die einzelnen Szenariobewertungen erstellen. Anhand der Informationen aus den bewerteten Szenarien können außerdem in Schritt 4 die Szenariointeraktionen abgeleitet werden. Szenariobewertungen und Szenariointeraktionen fließen schließlich in Schritt 5 in die Gesamtbewertung ein.

5.3 SAAM Schritt 1 - Beschreibung der Architektur

Im ersten Schritt wird eine für das Architekturverfahren geeignete Repräsentation der Softwarearchitektur gesucht und für die Beschreibung der Architektur genutzt. Dafür finden zunächst verschiedene Analysen dieser Architektur statt.

5.3.1 Grundlegende Architektur

Als Vorbereitung der Architekturbeschreibung wird, basierend auf einer ersten Untersuchung des Quellcodes, der Stand des Prototyps vom 23.08.2016 betrachtet, um einen grundlegenden Eindruck des Anwendungssystems CIO-Navigator zu bekommen.

Aufbau der Komponentenarchitektur

Vom Ersteller des Prototyps ist angegeben, dass sich dessen grundlegender Aufbau an der Drei-Schichten-Architektur orientiert.

Als Ergebnis der Ableitungen der grundlegenden Architektur der Komponenten wurden die in Abbildung 5.2 dargestellten Elemente und deren wichtigste Beziehungen ausgemacht.

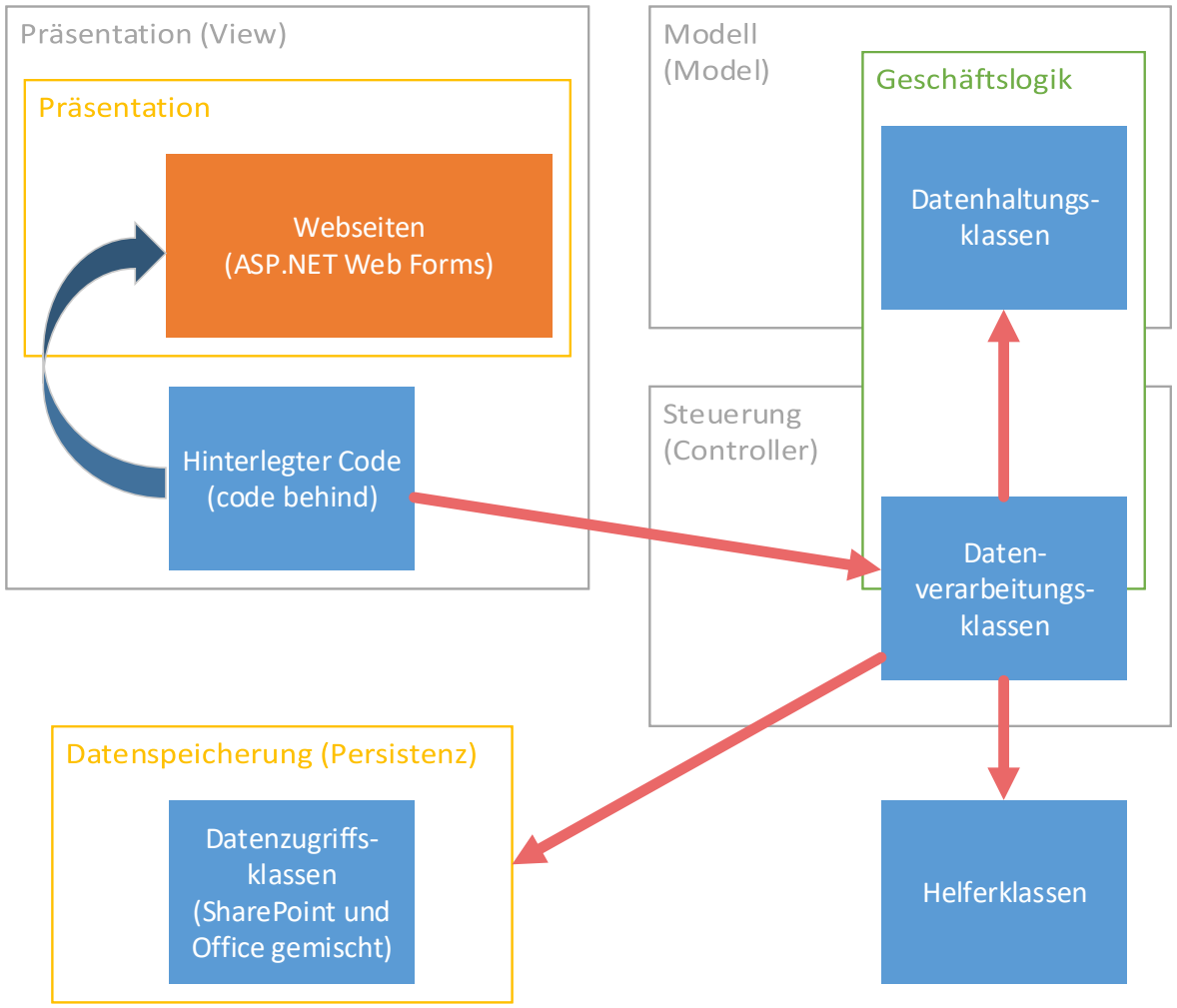


Abbildung 5.2: Ist-Zustand der Komponentenarchitektur des CIO-Navigators

Die hellblauen Rechtecke sind zusammengehörige Klassenelemente. Für die zur Darstellung der Webseite gehörenden Elemente steht das orangene Rechteck. Die gelben Behälterelemente bilden Zuordnungen zu den beiden Schichten Datenspeicherung und Präsentation der Drei-Schichten-Architektur ab. Alles was nicht von den gelben Behälterelementen umfasst wird, ist der Anwendungslogik-Schicht zuzuordnen. Die grauen Behälterelemente zeigen die vorhandenen Zuordnungen zu den drei Einheiten des Model-View-Controller-Entwurfsmusters. Grüne Behälterelemente beinhalten durch ihre Benennung bestimmte weitere Strukturen, beispielsweise die Geschäftslogik. Klassenelemente, welche sich nicht komplett in Behälterelementen befinden, sind diesen nur teilweise zuzurechnen. Die roten Pfeile stellen wichtige Zugriffsabhängigkeiten dar. Der blaue Pfeil zeigt lediglich die Beziehungen zum hinterlegten Quellcode (Code-Behind-Ansatz) an.

Aufbau des Model-View-Controller-Entwurfsmusters

Vom Ersteller des Prototyps wurde weiterhin mitgeteilt, dass das Model-View-Controller-Entwurfsmuster zum Einsatz kommt, um Daten von Datenbeschaffungslogik und Präsentation zu trennen.

Die Analyse der Verwendung des Model-View-Controller-Entwurfsmusters für den Prototypen zeigt, dass bei diesem weder das *ASP.NET MVC*-Framework noch ein anderes Framework, welches dieses Entwurfsmuster umsetzt, zum Einsatz kommt. Außerdem ist das Model-View-Controller-Entwurfsmuster, wie sich bei genauerer Untersuchung herausstellt, auch manuell nicht durchgängig umgesetzt. Es ergibt sich folgendes Bild.

Als die Grundelemente werden hier wie vorgesehen die Präsentation (View), die Steuerung (Contoller) und das Modell (Model) unterschieden. Die Präsentation dient, wie im Model-View-Controller-Entwurfsmuster vorgesehen, der Darstellung von Informationen für BenutzerInnen. Die Steuerung sollte die Eingaben der BenutzerInnen entgegennehmen, um sie an das Modell zur Verarbeitung und Abspeicherung weiterzugeben. Außerdem sollte sie die Form der Präsentation auswählen. Statt dessen wird die Möglichkeit des *ASP.NET Web Forms*-Frameworks Eingaben unter Verwendung der Präsentationslogik zu tätigen genutzt. Diese werden hier an die Steuerung gesendet, welche diese verarbeitet. Ebenfalls über die Steuerung holt sich die Präsentation beim Aufruf der zugehörigen Webseiten die benötigten Daten aus dem Modell. Abbildung 5.3 verdeutlicht diesen Aufbau.

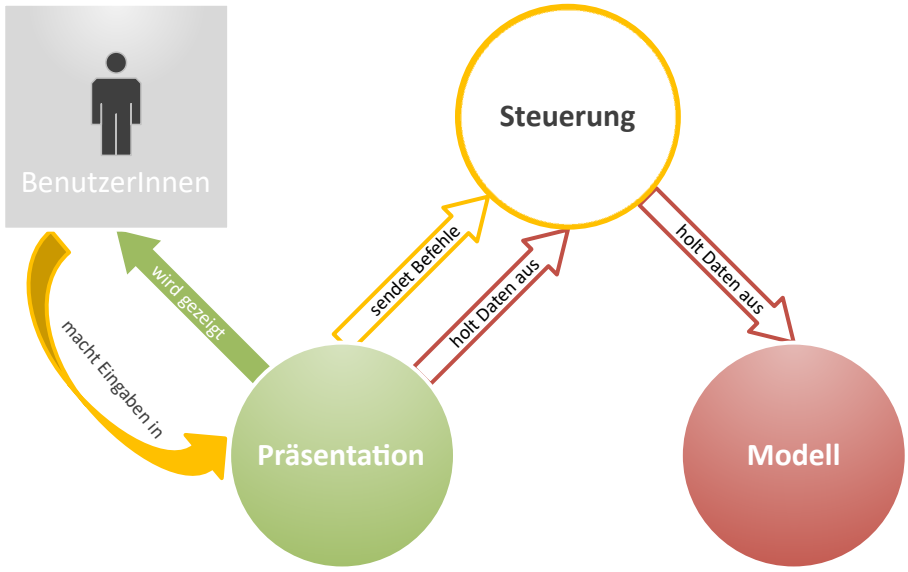


Abbildung 5.3: Model-View-Controller-Entwurfsmuster beim Prototypen

Die Bedeutung der Elemente und Beziehungen ist analog zu Abbildung 2.2. Unausgefüllte, lediglich umrandete Elemente und Beziehungen weisen auf nicht konform zur Architektur verwendete Strukturen hin. Die geschwungenen Pfeile zeigen im Framework vorhandene, sich jedoch von der Standard-Architektur unterscheidende Beziehungen an.

Bewertung der grundlegenden Architektur

Die aufgefundene Architektur befindet sich nicht vollständig im Einklang mit dem Model-View-Controller-Entwurfsmuster. Zunächst fällt auf, dass die eigentlichen Steuerungsklassen für die Steuerungseinheit nicht vorhanden sind. Statt dessen wurden die Klassen für die Datenverarbeitung dort platziert, obwohl diese anhand ihrer Funktionalitäten eigentlich der Modelleinheit zuzuordnen sind. Die ebenfalls dieser Einheit zugehörigen Klassen für die Persistenz sind hier keiner Einheit des Model-View-Controller-Entwurfsmusters zugeordnet. Die Klassen zur Befüllung der Webseiten mit Daten sind als hinterlegter Quellcode für die einzelnen Webseiten umgesetzt. Diese mit den Daten aus dem Modell zu versorgen, sollte eigentlich zum Teil Aufgabe der Klassen der Steuerungseinheit sein und nicht des hinterlegten Quellcodes für die Webseiten.

Auch sind nicht alle Beziehungen konform zum Model-View-Controller-Entwurfsmuster. Da keine Auswahl der Präsentation über die Steuerungseinheit erfolgt, ist dieses hierbei nicht vollständig eingehalten.

5.3.2 Analyse der Architektur mit dem Werkzeug CodeMap

Um weitere Erkenntnisse zur Architektur zu gewinnen kommt hier das Architekturanalyse-Werkzeug *CodeMap*, welches in *Microsoft Visual Studio Enterprise 2015* integriert ist, zum Einsatz. Dieses stellt verschiedene Strukturen der Anwendung und Beziehungen zwischen diesen grafisch dar, um die Analyse von Architekturen zu unterstützen. Der Stand des dabei verwendeten Prototyps ist die Version vom 23.08.2016.

Diagramm-Legende

Für unsere Betrachtungen wichtig sind die in Abbildung 5.4 und Abbildung 5.5 dargestellten Elemente und deren Beziehungen, welche in *CodeMap*-Diagrammen verwendet werden:

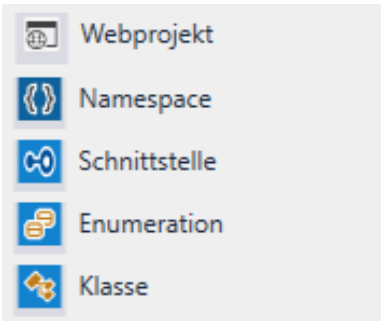


Abbildung 5.4: Legende Elemente

Abbildung 5.5: Legende Beziehungen

Das Webprojekt-Element umfasst dabei das gesamte Projekt. In diesem angeordnet sind die Namensräume (Namespaces), in welchen sich wiederum die Klassen, Schnittstellen und Aufzählungstypen (Enumerations) befinden können.

Vererbungen sind als grüne Linien dargestellt und Implementierungen als grün gestrichelte Linien. Aufrufe sind durch violette Linien abgebildet. Wird aus Feldern gelesen, so wird dies mit hellblau gestrichelten Linien angezeigt. Verweise, Rückgaben und verwendete Attribute sind als graue Linien dargestellt. Bei allen Beziehungen aus dieser Legende ist immer eine durch eine Pfeilspitze gekennzeichnete Richtung angegeben.

Analyse auf Ebene der Namensräume und Klassen

Ab hier werden die Architekturen des Prototyps auf der Ebene der Namensräume und Klassen sowie deren Beziehungen mit Hilfe von Visualisierungen durch *CodeMap* untersucht. Begonnen wird mit einem Klassendiagramm zur Übersicht über alle Klassen und einem weiteren Klassendiagramm für die Untersuchung von Vererbungsbeziehungen. Anschließend wird jeder Namensraum einzeln auf eine korrekte und sinnvolle Zuordung der Klassen überprüft. Aus den sich daraus ergebenden Verbesserungsvorschlägen wird ein angepasstes Klassendiagramm erstellt.
Anschließend werden bei den verbesserten Namensräumen deren Beziehungen untereinander betrachtet. Auch hier werden Verbesserungsvorschläge gemacht und die Auswirkungen gezeigt. Abschließend werden die Beziehungen der Klassen innerhalb der Namensräume analysiert.

Klassendiagramm mit Namensräumen

Bei einer ersten Betrachtung der Klassen in den Namensräumen ist zu erkennen, dass die beiden Namensräume *Microsoft.Office.Interop.Excel* und *Microsoft.Office.Interop.Word* ausschließlich Interfaces enthalten. Beim Öffnen des Quellcodes dieser Interfaces wird von *Microsoft Visual Studio Enterprise 2015* angegeben, dass die angezeigten Inhalte aus Metadaten erzeugt werden. Somit können diese beiden Namensräume bei der weiteren Betrachtung der projektinternen Architekturen vernachlässigt werden.

Die Abbildung 5.6 zeigt alle Klassen des Projektes, bei denen der Quellcode im Projekt vorhanden ist. Zusätzlich sind die Namensräume in denen diese angeordnet sind sowie das die Namensräume beinhaltende Assembly des Webprojekts zu sehen.

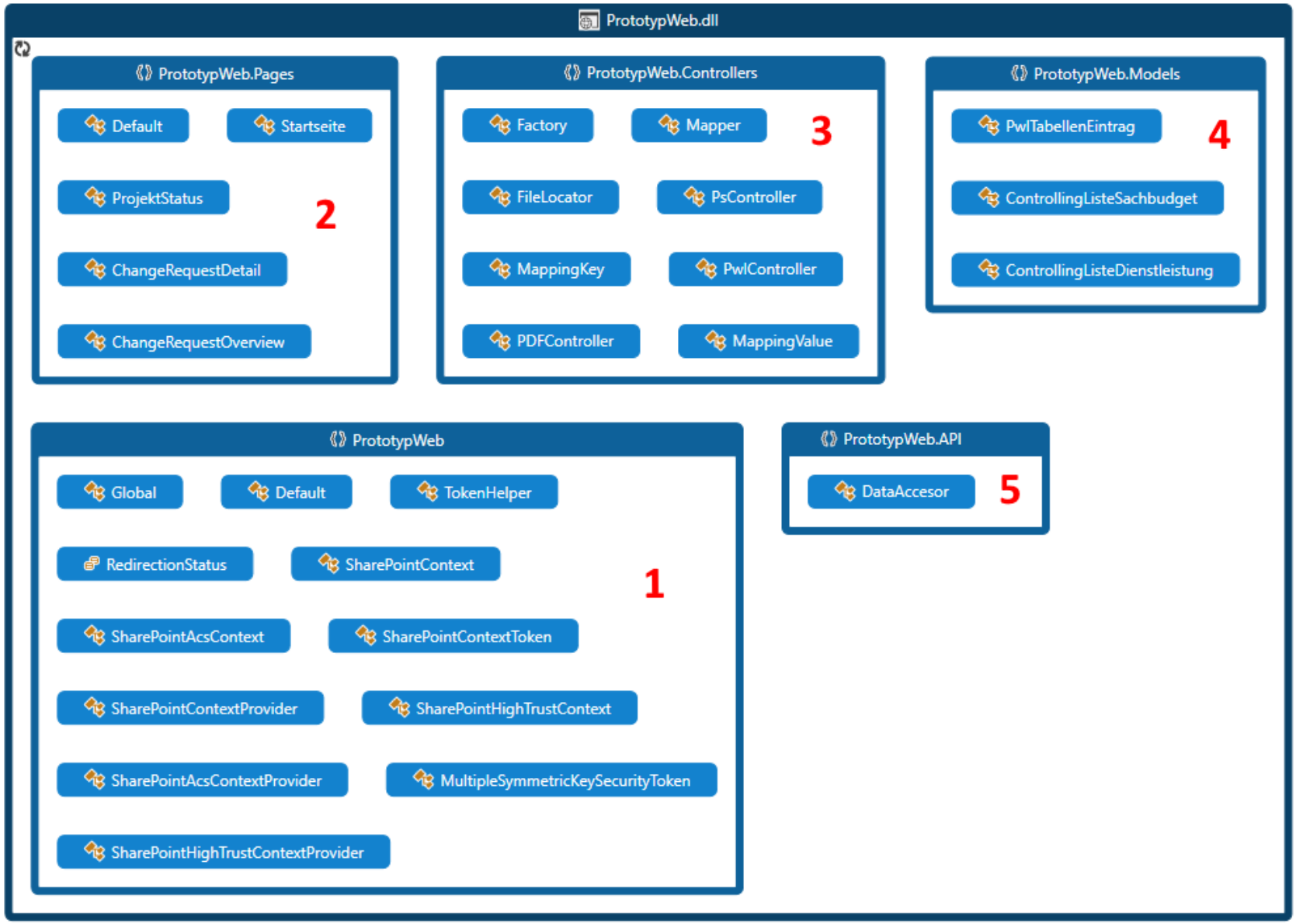


Abbildung 5.6: Relevantes Klassendiagramm mit Namensräumen

Vererbungsbeziehungen

Abbildung 5.7 zeigt die Vererbungsbeziehungen zwischen den Klassen des Projektes.

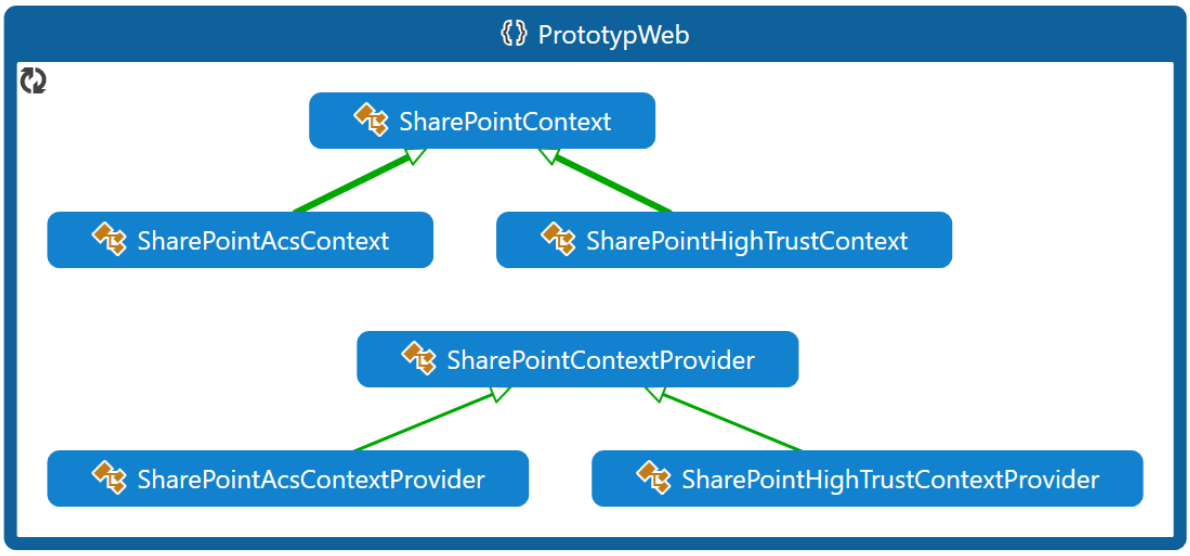


Abbildung 5.7: Vererbungsbeziehungen zwischen Klassen mit Namensräumen

Die Klasse *SharePointContext* hat zwei Spezialisierungen *SharePointAcsContext* und *SharePointHighTrustContext*. Der Benennung nach handelt es sich bei der ersten um eine Spezialisierung unter Verwendung des *Microsoft Azure Access Control Service* und bei der zweiten um eine Spezialisierung, welche gegen Sicherheitsrisiken stark abgesichert ist. Das gleiche Muster ist auch bei den entsprechenden Provider für den *SharePointContext* zu festzustellen. Die untersuchten Klassen sind allesamt generiert. Die nicht generierten Klassen untereinander besitzen keine Vererbungsbeziehungen.

Untersuchung des Namensraums PrototypWeb

Im Namensraum *PrototypWeb* mit der Nummer eins in Abbildung 5.6 befinden sich die dort dargestellten Klassen.

Der Quellcode der Klassen im Namensraum *PrototypWeb* lässt drei Gruppierungen zusammengehöriger Klassen erkennen. Zunächst existiert die Klasse *Global*, welche Aktivitäten beim Start der Applikation durchführt und bei dieser Implementierung die dabei erstellten Objekte bereitstellt. Des Weiteren gibt es eine Vielzahl von Klassen mit Bezug zu SharePoint. Schließlich bleibt eine Klasse mit dem Name *Default* übrig, die Webseiten-Quellcode beinhaltet.

Die Klasse *Global* befindet sich an der standardmäßig vorgegebenen Stelle.
Die Klasse *Default* gehört zur Präsentation der Applikation und befindet sich somit im falschen Namensraum. Daher wird empfohlen, sie in den Namensraum *PrototypWeb.Pages* zu verschieben. Dort existiert jedoch bereits eine Klasse mit dem gleichen Namen. Bei einem Blick auf die entsprechenden Abschnitte des Quellcodes stellt sich heraus, dass bei der Klasse im hier untersuchten Namensraum *PrototypWeb* eine fehlerhafte, zu korrigierende Benennung und

Namensraumzuordnung vorliegt. Die schon im Namensraum *PrototypWeb.Pages* vorhandene Klasse ist lediglich ein leerer Platzhalter, der entfernt werden kann.
Den Rest bilden die Klassen mit Bezug zu SharePoint. Diese sind keiner Einheit des Model-View-Controller-Entwurfsmusters eindeutig zuzurechnen und sollten der Übersicht halber zu einem eigenen Namensraum gehören.

Untersuchung der Rolle des Namensraums PrototypWeb.Pages

Der Namensraum *PrototypWeb.Pages* mit der Nummer zwei in Abbildung 5.6 beinhaltet die dort dargestellten Klassen.

Die Klassen des Namensraums *PrototypWeb.Pages* gehören zur Präsentation der Applikation und stellen Programmlogik für die zugehörigen Webseiten bereit. Damit befinden sie sich korrekter Weise in dem dafür vorgesehenen Namensraum.

Untersuchung der Rolle des Namensraums PrototypWeb.Controllers

Die in Abbildung 5.6 dargestellten Klassen des Namensraums *PrototypWeb.Controllers* mit der Nummer drei beinhalten die Zuordnung für das Auffinden der Quelldaten und deren Verarbeitung.

Damit sind diese eher der Einheit für das Modell als der Einheit für die Steuerung zuzuordnen und im entsprechenden Namensraum *PrototypWeb.Models* zu platzieren. Außerdem empfiehlt sich, dass später Funktionalitäten welche zur Geschäftslogik gehören, von den Funktionalitäten für den Datenzugriff getrennt werden. Dafür muss das Aggregieren von Daten in eigene Klassen ausgelagert werden.

Untersuchung der Rolle des Namensraums PrototypWeb.Models

Der Namensraum *PrototypWeb.Models* mit der Nummer vier in Abbildung 5.6 beinhaltet die dort aufgeführten Klassen.

Diese Klassen sind reine Datenhaltungsklassen und der Einheit für das Modell zuzurechnen. Der Übersicht halber sollen sie in einen Unternamensraum ausgelagert werden.

Untersuchung der Rolle des Namensraums PrototypWeb.API

Der in Abbildung 5.6 dargestellte Namensraum *PrototypWeb.API* mit der Nummer fünf beinhaltet lediglich eine Klasse *DataAccesor*.

Über diese Klasse finden die Zugriffe auf die unterschiedlichen Datenquellen statt. Damit ist diese Klasse der Persistenzschicht aus der Drei-Schichten-Architektur zuzuordnen. Insofern ist sie der Einheit für das Modell zuzurechnen und soll in einen neuen entsprechenden Unternamensraum verschoben werden. Auch empfiehlt es sich die Klasse später nach Funktionen für das Auslesen von SharePoint-Ressourcen und Excel-Dokumenten aufzuteilen.

Klassendiagramm nach Umordnung der Klassen und Namensräume

Nach Einarbeitung der Verbesserungsvorschläge in Bezug auf die Namensräume und zugehörigen Klassen sieht das Klassendiagramm wie in Abbildung 5.8 dargestellt aus.

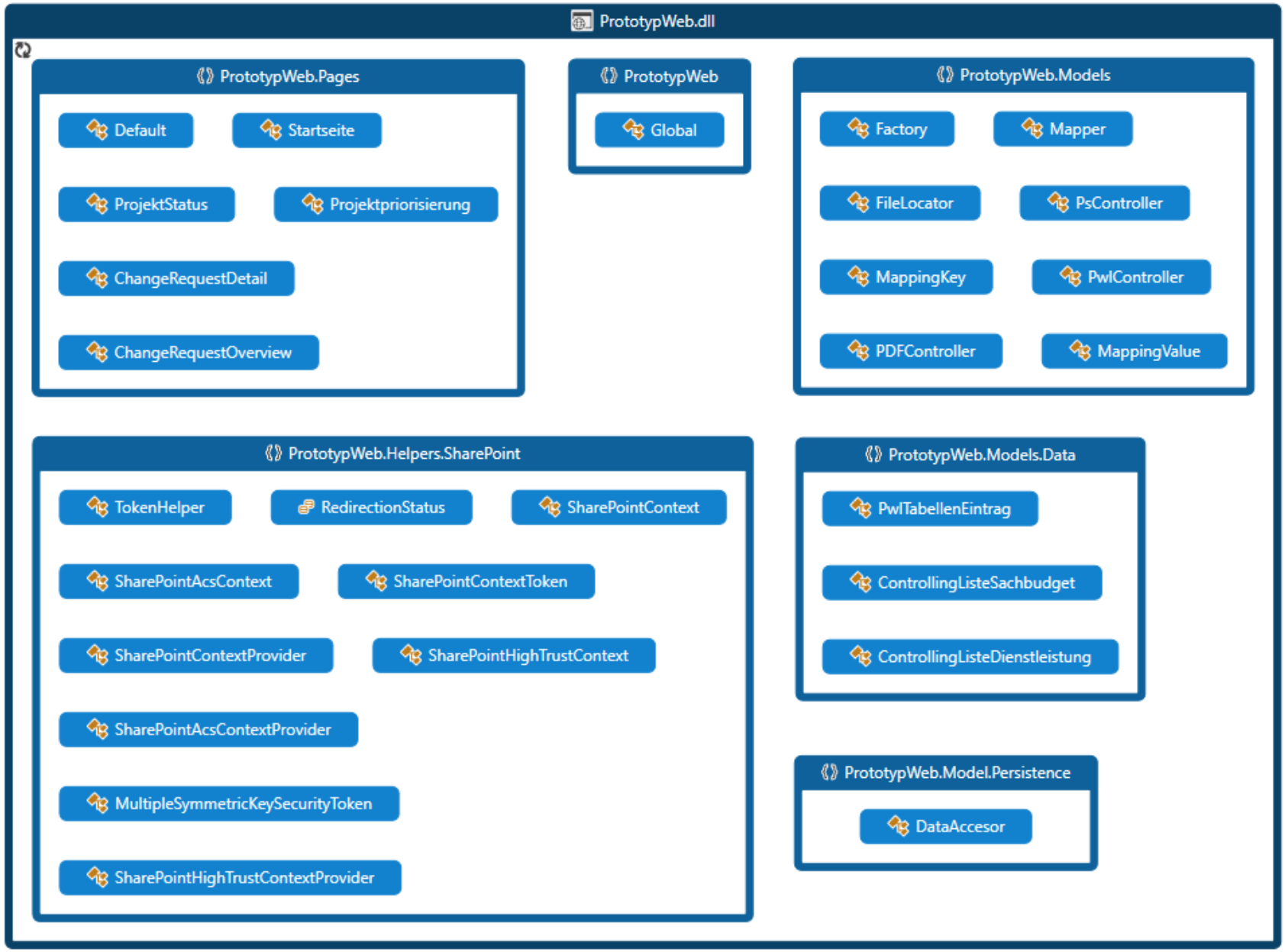


Abbildung 5.8: Klassendiagramm mit Namensräumen nach Umordung

Es existiert kein Namensraum *PrototypWeb.Controllers* für die Steuerungseinheit mehr. Der ursprüngliche Inhalt dieses Namensraums ist aufgrund seiner Funktionen der Modelleinheit zuzurechnen und wurde deshalb in den zugehörigen Namensraum versetzt. Die Datenhaltungsklassen wurden in einen Unternamensraum *PrototypWeb.Models.Data* verschoben. Die Klasse für die Persistenz wurde in den Namensraum *PrototypWeb.Models.Persistence* verschoben.

Die Namensräume sind nun für eine Untersuchung ihrer Beziehungen untereinander vorbereitet.

Namensräume und Beziehungen zwischen ihnen

Die Namensräume, nach der Verbesserung ihrer internen Struktur und den damit einher gehenden Umbenennungen, sind aus Abbildung 5.8 zu entnehmen.

Die umgeordneten Namensräume können nun einfacher auf deren Beziehungen untereinander untersucht werden. Bei Anzeige aller Beziehungen ergibt sich das in Abbildung 5.9 gezeigte Diagramm.

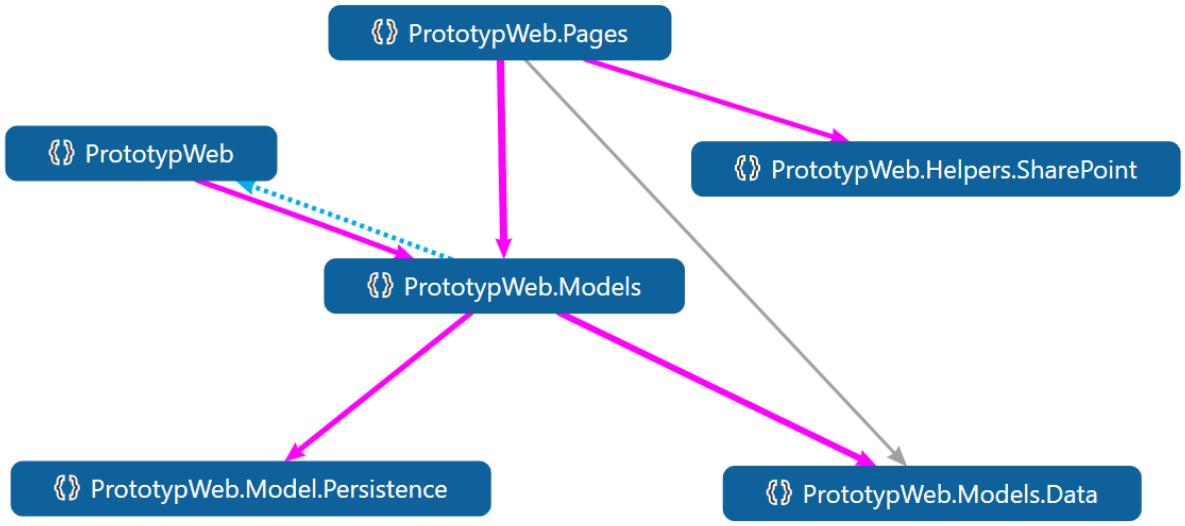


Abbildung 5.9: Beziehungen zwischen den umgeordneten Namensräumen

Man kann erkennen, dass auf die Helferklassen von SharePoint in *PrototypWeb.Helpers.SharePoint* nur vom Namensraum der Präsentationseinheit *PrototypWeb.Pages* aus zugegriffen wird. Des Weiteren greift der Namensraum der Präsentationseinheit nur noch auf die Namensräume zur Modelleinheit *PrototypWeb.Models* und *PrototypWeb.Models.Data* zu. Vom obersten Namensraum der Modelleinheit *PrototypWeb.Models* wird dabei, wie zu erwarten, auf dessen Unternamensräume *PrototypWeb.Models.Data* zugegriffen. Der oberste Namensraum der Anwendung *PrototypWeb* hat eine wechselseitige Beziehung zum Namensraum der Modelleinheit *PrototypWeb.Models*, welche in Abbildung 5.10 zum Zweck der ausführlicheren Untersuchung genauer dargestellt wird.

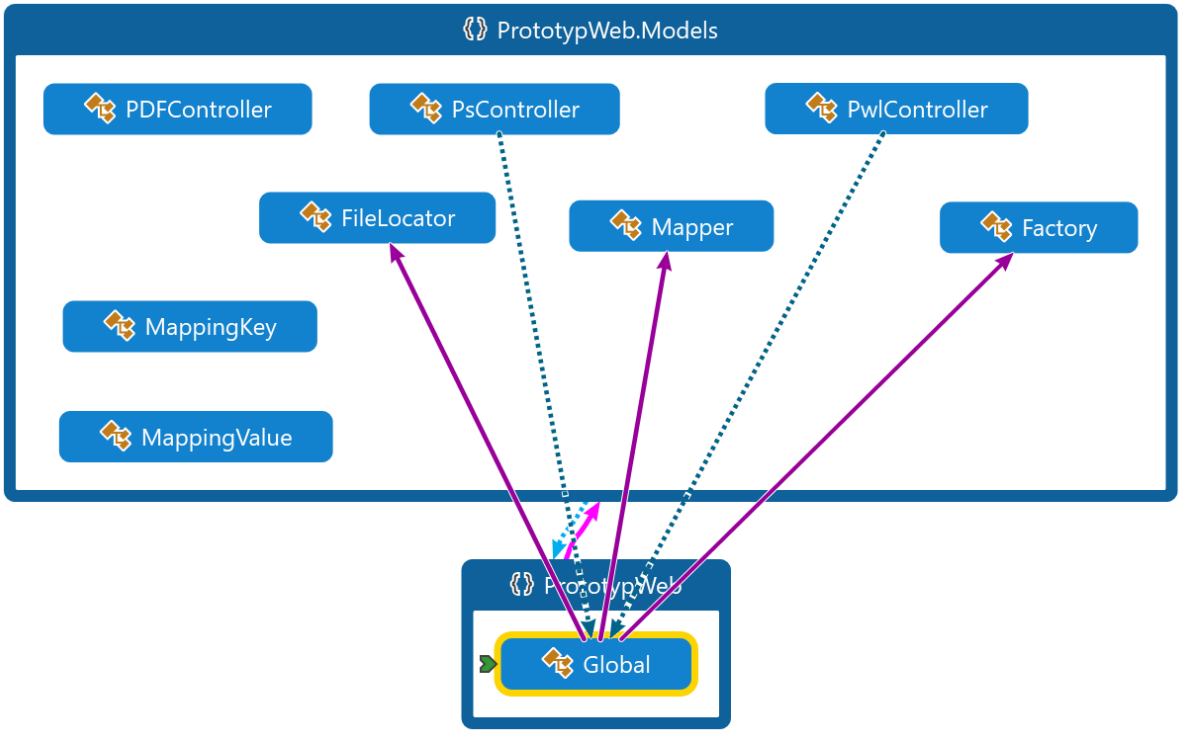


Abbildung 5.10: Verweis zwischen PrototypWeb und PrototypWeb.Models

Ein Blick in den Quellcode erklärt dieses Muster. In der Klasse *Global* wird Code, welcher nur einmal bei Anwendungsstart ausgeführt werden soll, in der dafür vorgesehenen Methode eingetragen. Dies wird hier genutzt, um über die entsprechenden Klassen verschiedene Zuordnungsdaten (*FileLocator* und *Mapper*) beim Start der Anwendung einmalig aus Dateien auszulesen, sowie nur einmal zu erstellende Klassen (*Factory*) zu verwalten. Die so erhaltenen Objekte werden durch Ablage in öffentlichen Klassenvariablen zugänglich gemacht. Auf diese greifen die nutzenden Klassen, welche sich nicht im selben Namensraum befinden, bei Bedarf lesend zu. Um die Beziehungen zwischen den Namensräumen übersichtlicher zu gestalten ist zu empfehlen, diese Funktionalitäten in eine eigenen Klasse, welche beim Start der Anwendung von der Klasse *Global* aufgerufen wird, auszulagern. Diese kann dann dem Namensraum zugewiesen werden, in welchem sich die diese Funktionalitäten nutzenden Klassen befinden. Auf diese Weise werden die Abhängigkeiten zwischen den betroffenen Namensräumen stark vereinfacht.

Mit dieser Optimierung ergibt sich das in Abbildung 5.11 dargestellte Diagramm.

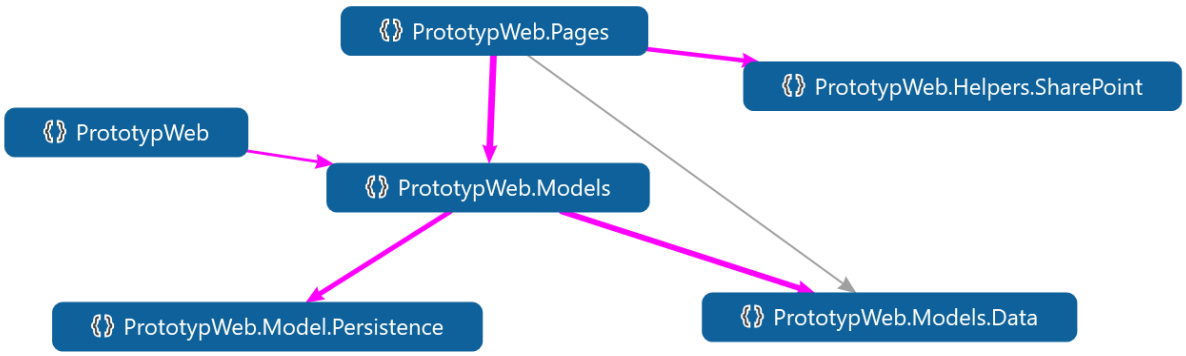


Abbildung 5.11: Optimierte Beziehungen zwischen den umgeordneten Namensräumen

Es ist zu sehen, dass die Beziehungen zwischen den Namensräumen nun klarer in Erscheinung treten. Damit ist die Verbesserung der Namensräume abgeschlossen.

Beziehungen der Klassen innerhalb der Namensräume

Die Namensräume *PrototypWeb*, *PrototypWeb.Pages*, *PrototypWeb.Models.Data* und *PrototypWeb.Models.Persistence* besitzen keine Beziehungen zwischen den in ihnen enthaltenen Klassen.

Die Beziehungen innerhalb des Namensraums *PrototypWeb.Helpers.SharePoint* sind in Abbildung 5.12 dargestellt.

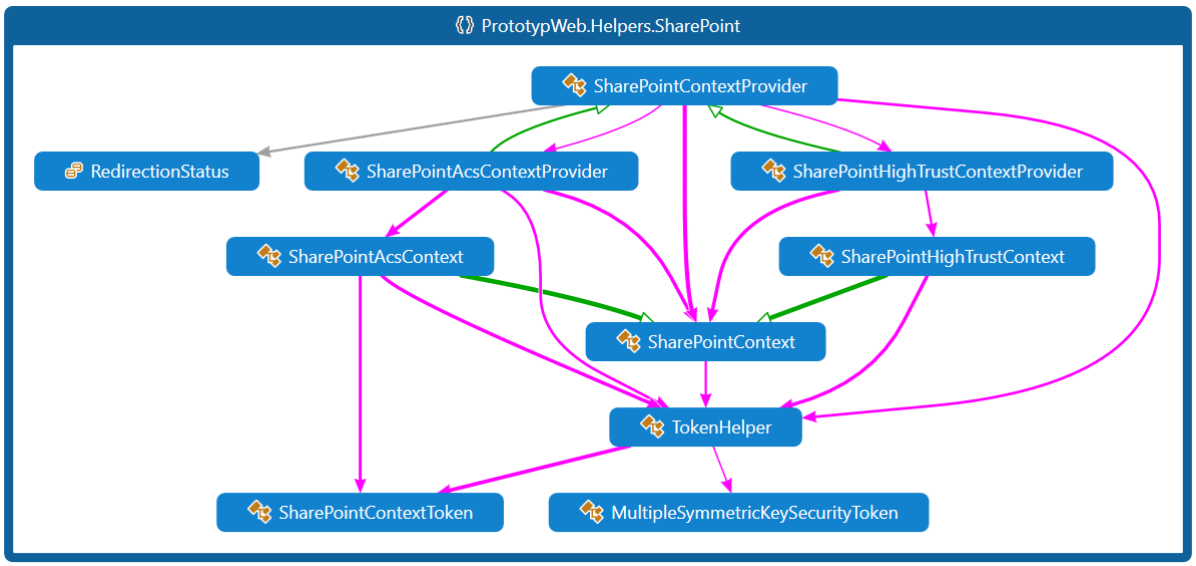


Abbildung 5.12: Beziehungen innerhalb von PrototypWeb.Helpers.SharePoint

Man erkennt hier sehr gut die internen Beziehungen der Klassen aus diesem Namensraum. Zunächst befindet sich da der SharePoint-Context mit seinen Varianten (*SharePointContext*, *Sha-*

rePointAcsContext und *SharePointHighTrustContext*). Diese werden über zugehörige Providerklassen (*SharePointContextProvider*, *SharePointAcsContextProvider* und *SharePointHighTrustContextProvider*) zur Verfügung gestellt. Schließlich greifen die SharePoint-Context-Klassen auf die zum Token-Helper gehörenden Klassen (*TokenHelper*, *SharePointContextToken* und *MultipleSymmetricKeySecurityToken*) zu. Die Providerklassen verfügen außerdem über einen eigenen Aufzählungstyp (*RedirectionStatus*).

Die Beziehungen innerhalb des Namensraums *PrototypWeb.Models* sind in Abbildung 5.13 dargestellt.

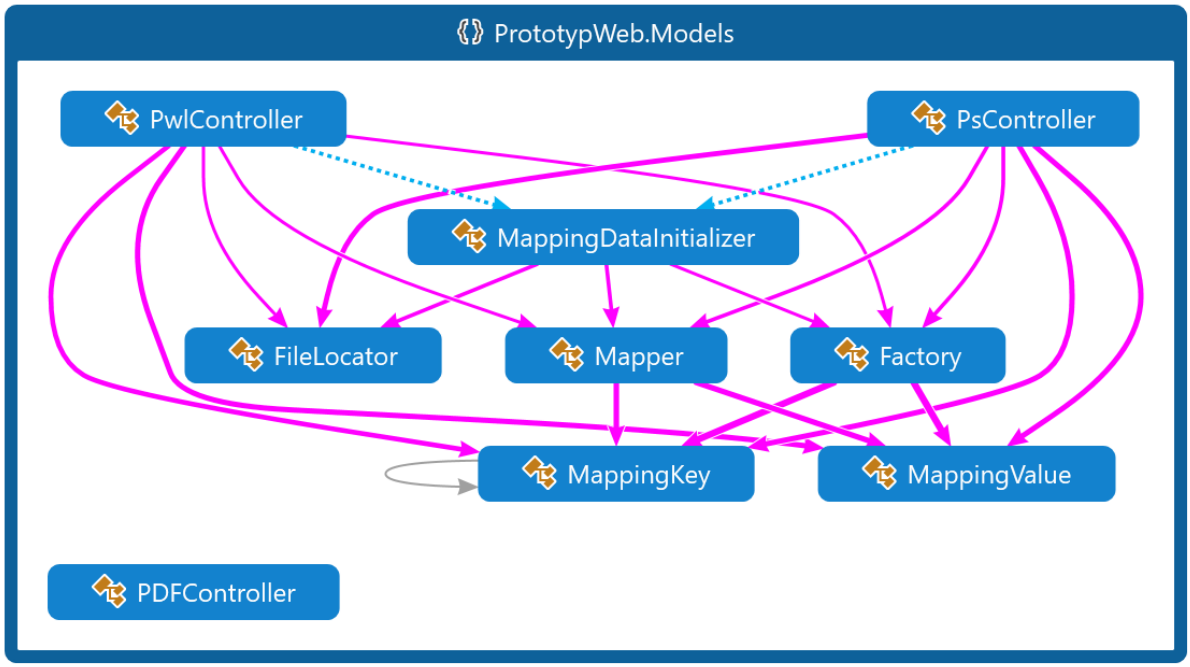


Abbildung 5.13: Beziehungen innerhalb von PrototypWeb.Models

Auch hier wird erkennbar, welche Zugriffsmuster in diesem Namensraum existieren. Die von außerhalb bei Anwendungsstart aufgerufene neue Klasse *MappingDataInitializer* erstellt und initialisiert die für die Datenzuordnung verwendeten Klassen (*FileLocator*, *Mapper* und *Factory*). Bei ihrer Erstellung holen sich die den Datenzugriff ausführenden Klassen (hier *PwlController* und *PsController*) dann die Verweise auf die Klassen für die Datenzuordnung (*FileLocator* und *Mapper*), um auf die entsprechenden Informationen zugreifen zu können. Mit diesen Informationen erstellen und befüllen sie, unter Verwendung der Klasse *Factory*, zugehörige Datenhaltungsklassen. Zur Klasse *PDFController* können keine Aussagen gemacht werden, da sich diese anhand ihres Quellcodes als noch unfertig erweist.

5.3.3 Analyse der Architektur mit messbaren Eigenschaften des Quellcodes

Mittels des Quellcodes können Eigenschaften der Architektur eines Systems untersucht werden. Dabei unterstützen Code-Metriken, die unterschiedliche Eigenschaften des Quellcodes quantifizieren.

Als ein Werkzeug zur intuitiven Darstellung solcher Code-Metriken wurde hier *CodeCity* für verschiedene Analysen ausgewählt. Um diese Darstellungen um konkrete Zahlen und weitere in *CodeCity* nicht zur Auswahl stehende Code-Metriken zu ergänzen, wurde außerdem noch eine Untersuchung mit dem Werkzeug *Codemetrik*, welches in *Microsoft Visual Studio Enterprise 2015* integriert ist, durchgeführt.

Für die Analyse der Klassen des Projekts sind die in Tabelle 5.2 aufgelisteten und erläuterten Code-Metriken ausgewählt worden. Bei den für *CodeCity* ausgesuchten Code-Metriken steht die Erstellung von möglichst intuitiv interpretierbaren und aufschlussreichen grafischen Darstellungen im Vordergrund. Bei *Codemetrik* sind nur die aussagekräftigsten Code-Metriken, welche von diesem Werkzeug unterstützt werden, einbezogen.

Anschließend an die folgende Auflistung der Code-Metriken wird der Prototyp des CIO-Navigators unter deren Verwendung analysiert. Der Stand des verwendeten Prototyps ist die Version vom 23.08.2016.

Tabelle 5.2: Verwendete Code-Metriken (teilweise nach [3])

Bezeichnung	**Beschreibung**	**Bedeutung**	**Verwendet in**
Codezeilen	Gibt die Anzahl von Zeilen des Quellcodes an. Die Zeilenanzahl basiert dabei auf dem Inline-Code (ohne beispielsweise Berücksichtigung von Klassen- und Methodenrümpfen).	Ein sehr hoher Wert kann darauf hinweisen, dass ein Typ oder eine Methode zu viele Aufgaben ausführt und aufgeteilt werden sollte. Er kann auch darauf hindeuten, dass der Typ oder die Methode schwierig zu verwalten ist. Generell ist die Anzahl der Codezeilen auch ein Indikator für die Größe eines Systems.	*CodeCity*, *Codemetrik*
Methodenanzahl	Gibt die Anzahl der Methoden im Code an.	Der Grad der Aufteilung des Codes und damit der Funktionalitäten wird angedeutet, besonders wenn diese Code-Metrik in Zusammenhang mit der Codezeilenanzahl gesetzt ist.	*CodeCity*
Zyklomatische Komplexität	Misst die strukturelle Komplexität des Codes. Sie wird durch Berechnung der Anzahl unterschiedlicher Codepfade im Programmfluss erstellt.	Für ein Programm mit komplexer Ablaufsteuerung sind mehr Komponententests erforderlich, wenn eine gute Codeabdeckung erzielt werden soll. Zudem verschlechtert sich die Wartbarkeit.	*Codemetrik*
Vererbungstiefe	Gibt die Anzahl der Klassendefinitionen an, die sich bis zum Ursprung der Klassenhierarchie erstrecken.	Je tiefer die Hierarchie, umso schwieriger ist unter Umständen zu erkennen, wo bestimmte Methoden und Felder definiert oder neu definiert werden.	*Codemetrik*
Klassenkopplung	Misst die Kopplung einer Klasse an andere Klassen durch Parameter, lokale Variablen, Rückgabetypen, Methodenaufrufe, generische oder Vorlageninstanziierungen, Basisklassen, Schnittstellenimplementierungen, für externe Typen definierte Felder sowie Attributdekorationen.	Eine enge Kopplung deutet auf einen Entwurf hin, der aufgrund zahlreicher gegenseitiger Abhängigkeiten zu anderen Typen nur schwer wiederzuverwenden und zu warten ist.	*Codemetrik*

5.3.4 Untersuchungen mit dem Werkzeug CodeCity

CodeCity ist ein programmiersprachen-unabhängiges, interaktives 3D-Visualisierungswerkzeug für die Analyse von objektorientierten Softwaresystemen. Es generiert aus Informationen über den Quellcode dreidimensionale Darstellungen, welche aus Grundflächen und Blöcken bestehen und Ähnlichkeit mit Städten besitzen [38]. Die Flächen stellen nach dieser Städte-Metapher Stadtviertel dar und die Blöcke repräsentieren Gebäude [39]. Die Blöcke werden unter Zuhilfenahme der Code-Metriken der Klassen generiert, die Grundflächen ergeben sich aus deren Namensräumen. Eine Untersuchung der Klassen des Prototyps mit Hilfe dieses Werkzeugs ist im Folgenden ausgeführt.

Darstellung zum Klassenumfang

Erste Analysen von Darstellungen mit *CodeCity* zeigen, dass es der Übersicht halber besser ist, automatisch generierte Klassen nicht zu berücksichtigen. Diese ziehen, falls sie in den dargestellten Code-Metriken hohe Werte besitzen, viel Aufmerksamkeit auf sich ohne eine sinnvolle Aussage zu haben. Im Fall des Prototypen sind davon alle Klassen in Zusammenhang mit dem *SharePointContext* betroffen. Der Einfachheit wegen sind auch Hilfsklassen der Datenhaltung, welche nicht der Geschäftslogik zuzurechnen sind weggelassen. Davon betroffen sind *PrototypWeb.Controllers.MappingKey* und *PrototypWeb.Controllers.MappingValue*. Es ergibt sich folgende Liste der dargestellten Klassen:

(A) *PrototypWeb.Pages.ProjektStatus*
(B) *PrototypWeb.Pages.Projektpriorisierung*
(C) *PrototypWeb.Pages.Startseite*
(D) *PrototypWeb.Pages.ChangeRequestOverview*
(E) *PrototypWeb.Pages.ChangeRequestDetail*
(F) *PrototypWeb.Controllers.PsController*
(G) *PrototypWeb.Controllers.PwlController*
(H) *PrototypWeb.Controllers.PDFController*
(I) *PrototypWeb.Controllers.FileLocator*
(J) *PrototypWeb.Controllers.Mapper*
(K) *PrototypWeb.Controllers.Factory*
(L) *PrototypWeb.Models.PwlTabellenEintrag*
(M) *PrototypWeb.Models.ControllingListeSachbudget*
(N) *PrototypWeb.Models.ControllingListeDienstleistung*
(O) *PrototypWeb.API.DataAccesor*
(P) *PrototypWeb.Global*

In Abbildung 5.14 sind diese Klassen in der Vogelperspektive abgebildet. Die grauen Flächen stehen dabei für die Namensräume der untersuchten Klassen. Die blauen Blöcke stehen für die Klassen selbst. Länge sowie Breite der Blöcke sind immer gleich und ergeben sich aus der

Anzahl der Codezeilen dieser Klassen. Die weißen Beschriftungen geben die Klassennamen an, die schwarzen die Bezeichnungen der Namensräume.

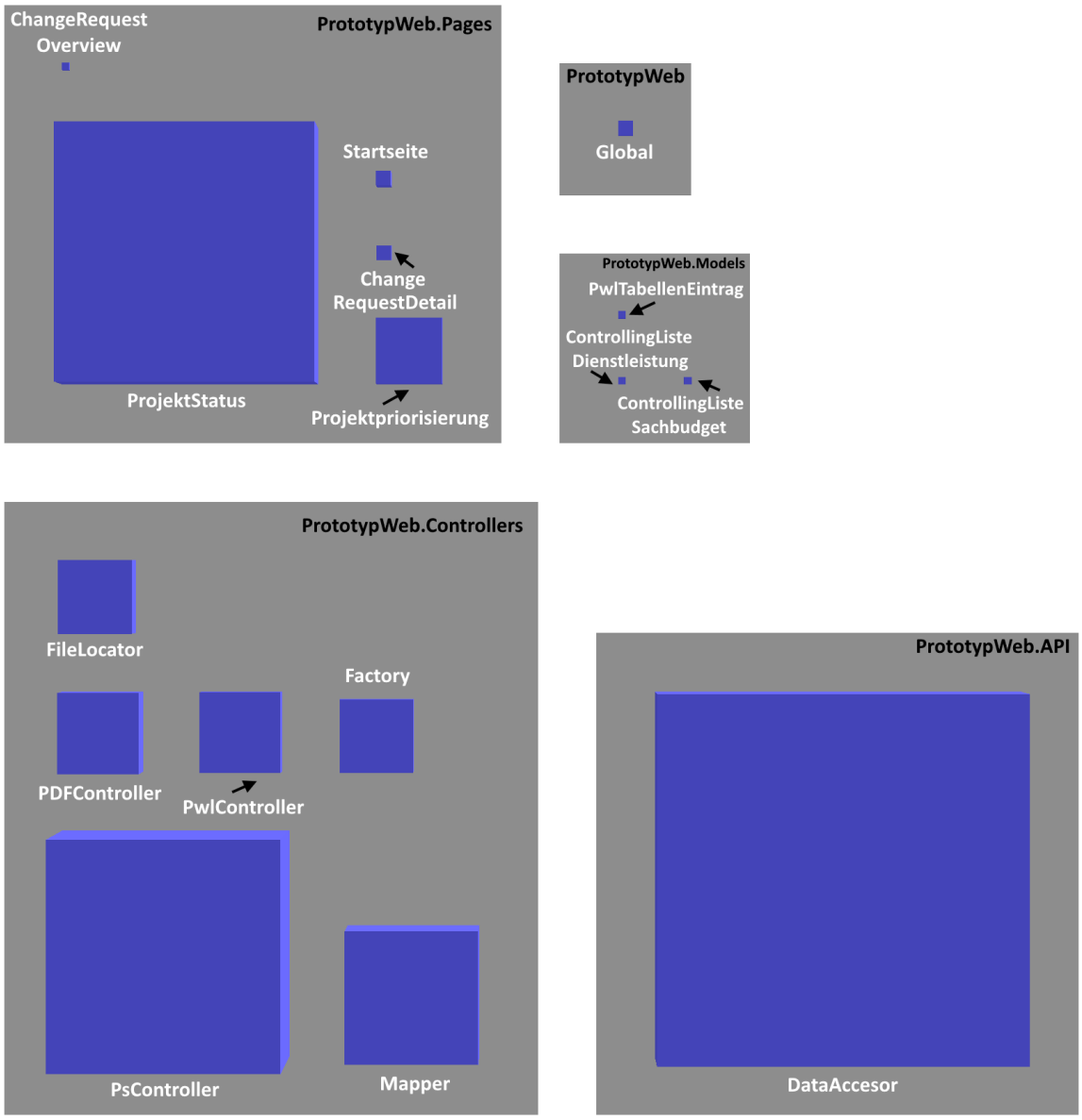


Abbildung 5.14: Darstellung für Anzahl der Codezeilen mit CodeCity von oben

Es ist zu erkennen, dass die Klasse *PrototypWeb.API.DataAccesor* mit Abstand die größte Anzahl an Codezeilen besitzt. Dies ist ein Hinweis darauf, dass diese Klasse viel Funktionalität beherbergt. Auch *PrototypWeb.Controllers.PsController* und *PrototypWeb.Pages.ProjektStatus* besitzen viel Quellcode. Bei der Klasse *PrototypWeb.Controllers.PsController* ist es aufgrund der umfangreichen Funktionalität, welche diese Klasse bereitstellt plausibel, dass diese viel Quellcode enthält. Dass die Webseiten-Code-Klasse für die Anzeige des Projektstatus (*PrototypWeb.Pages.ProjektStatus*) so umfangreich ausfällt, ist dem für die Befüllung der zugehörige Seite großen Aufwand geschuldet und damit nicht weiter interessant. Die Datenhaltungsklassen der Geschäftslogik (*PrototypWeb.Models.PwlTabellenEintrag*, *PrototypWeb.Models.ControllingListeSachbudget*, *PrototypWeb.Models.ControllingListeDienstleistung*)

sind wie zu erwarten von relativ kleinen Umfang. Auch die Klasse *PrototypWeb.Global* ist, wie aufgrund vom geringen Konfigurationsaufwand beim Start der Anwendung anzunehmen, klein ausgefallen.

Mit Abbildung 5.15 folgt eine isometrische Betrachtung des bereits in Abbildung 5.14 abgebildeten Modells. Die hier gut zu erkennende unterschiedliche Höhe der die Klassen darstellenden Blöcke bezieht sich auf die Anzahl der Methoden der zugehörigen Klassen. Bezüglich dieses Maßes sind keine weiteren Auffälligkeiten zu beobachten.

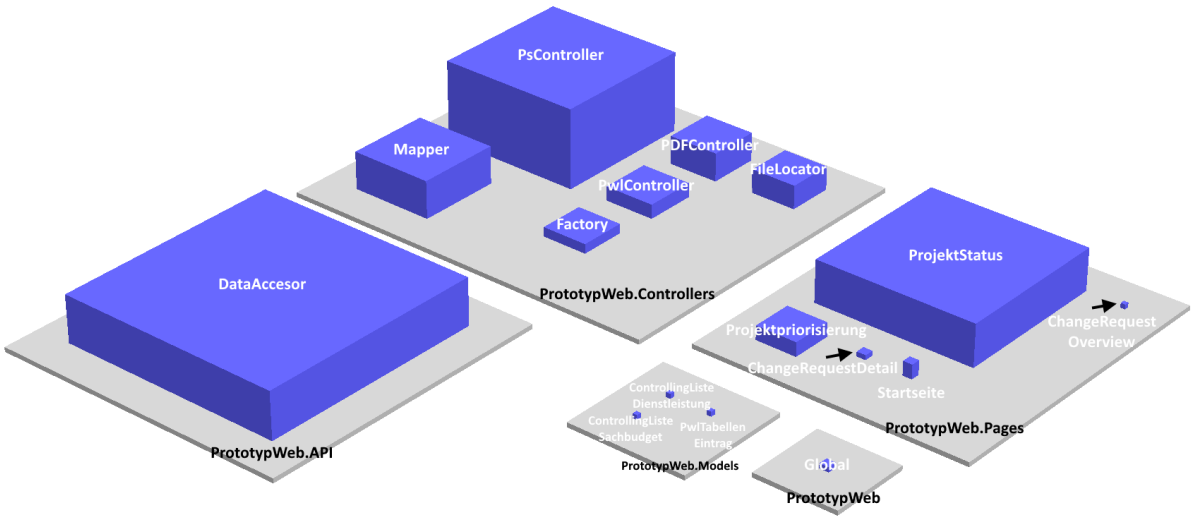


Abbildung 5.15: Isometrische Darstellung für Anzahl der Codezeilen und Methoden mit CodeCity

Die Bedeutung der abgebildeten Elemente ist analog zu Abbildung 5.14.

Methodendarstellung

Die folgende ebenfalls isometrische Abbildung 5.16 ist eine gute Ergänzung zu den vorherigen Abbildungen, da diese die Anzahl der Methoden pro Klasse abzählbar darstellt. Die Anzahl der Codezeilen der Klassen wird bei dieser Darstellung, im Unterschied zu den vorherigen Abbildungen, nicht berücksichtigt.

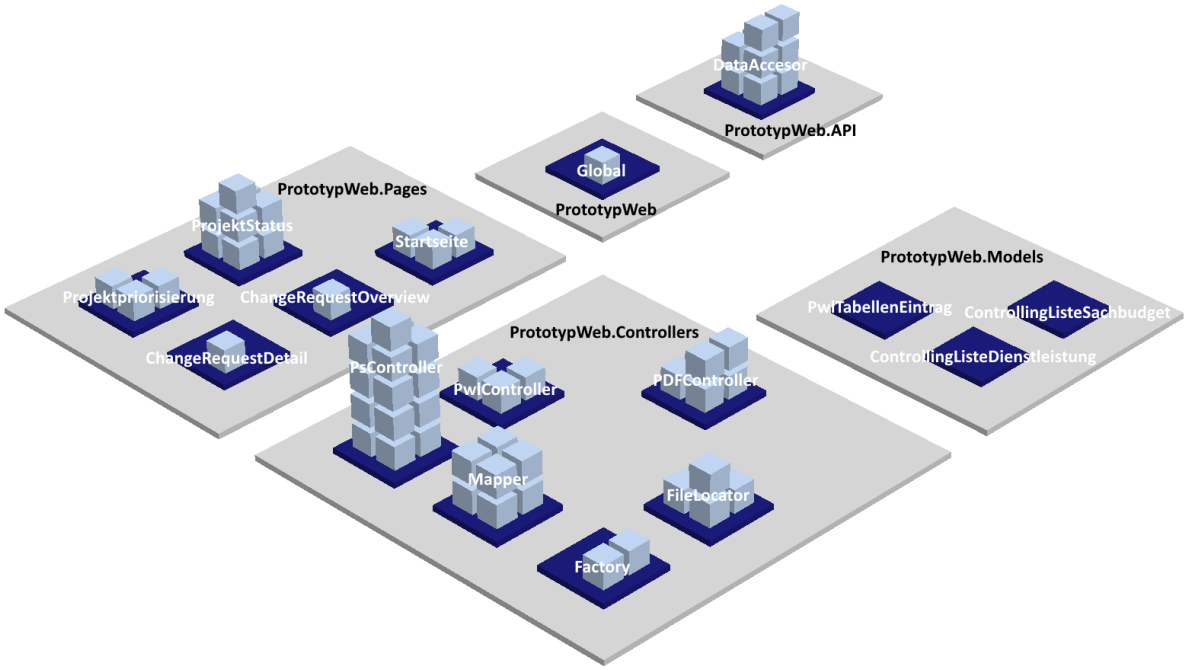


Abbildung 5.16: Isometrische Darstellung der Methoden pro Klasse mit CodeCity

Auch in dieser Grafik sind Namensräume als graue Flächen abgebildet. Die Darstellung der Klassen weist hier jedoch überall die gleiche Länge und Breite auf und hat damit keine weitere Aussage. Dafür wird die Anzahl der Methoden der zugehörigen Klassen durch aufeinander gestapelte Würfel dargestellt. Dadurch gibt die Höhe weiterhin die Anzahl der Methoden an, wobei sich diese nun genau abzählen lässt.

5.3.5 Untersuchungen mit dem Werkzeug Codemetrik

In Abbildung 5.17 sind die Ergebnisse einer Analyse mit dem Werkzeug *Codemetrik*, welches in *Microsoft Visual Studio Enterprise 2015* integriert ist, aufgeführt. Diese Auflistung macht die Werte für Codezeilen, zyklomatische Komplexität, Vererbungstiefe und Klassenkopplung auf einen Blick einsehbar und vergleichbar. Dabei gibt es einige Auffälligkeiten.

Hierarchie	Zyklomatische Komplexität	Vererbungstiefe	Klassenkopplung	Codezeilen
PrototypWeb (Debug)	570	4	217	1.228
{ } PrototypWeb	321	4	136	637
Default	13	4	32	28
Global	2	2	5	7
MultipleSymmetricKeySecurityToken	17	2	15	31
RedirectionStatus	0	1	0	0
SharePointAcsContext	25	2	12	38
SharePointAcsContextProvider	16	2	12	28
SharePointContext	32	1	13	44
SharePointContextProvider	36	1	17	77
SharePointContextToken	19	3	15	43
SharePointHighTrustContext	18	2	10	30
SharePointHighTrustContextProvider	14	2	9	16
TokenHelper	89	1	62	239
TokenHelper.AcsMetadataParser	16	1	16	32
TokenHelper.AcsMetadataParser.JsonEndpoint	7	1	0	7
TokenHelper.AcsMetadataParser.JsonKey	5	1	1	5
TokenHelper.AcsMetadataParser.JsonKeyValue	5	1	0	5
TokenHelper.AcsMetadataParser.JsonMetadataDocument	7	1	3	7
{ } PrototypWeb.API	52	1	74	149
DataAccesor	52	1	74	149
{ } PrototypWeb.Controllers	121	1	33	303
Factory	13	1	16	29
FileLocator	11	1	7	29
Mapper	20	1	9	45
MappingKey	11	1	2	18
MappingValue	15	1	3	25
PDFController	15	1	1	44
PsController	28	1	18	84
PwlController	8	1	19	29
{ } PrototypWeb.Models	43	1	1	43
ControllingListeDienstleistung	9	1	0	9
ControllingListeSachbudget	9	1	0	9
PwlTabellenEintrag	25	1	1	25
{ } PrototypWeb.Pages	33	4	23	96
ChangeRequestDetail	2	4	5	10
ChangeRequestOverview	2	4	7	2
Default	2	4	3	1
ProjektStatus	23	4	19	79
Startseite	4	4	7	4

Abbildung 5.17: Code-Metriken mit dem Werkzeug Codemetrik

In der Spalte **Hierarchie** ist die Hierarchie des Projektes mit Namensräumen und Klassennamen aufgelistet. Die erste Zeile zeigt in der Spalte **Codezeilen**, dass das gesamte Projekt nach *Codemetrik* 1228 Codezeilen enthält.

Die maximale Vererbungstiefe überschreitet nicht den Wert vier. Ohne Berücksichtigung der Webseitenklassen und der generierten Klassen erbt tatsächlich nur noch die Klasse *PrototypWeb.Global*.

Die automatisch generierten Klassen unter dem Aufzählungspunkt *SharePointContext (mit Provider)* in Tabelle 5.3 tragen zu einem Großteil der ermittelten zyklomatischen Komplexität bei. Unter Ausblendung dieser Klassen weisen die Klassen mit vielen Codezeilen, wie bei diesen zu erwarten, eine höhere zyklomatische Komplexität auf als die kleineren Klassen. Lediglich die Klasse *PrototypWeb.Controllers.Mapper* hat für die Größe der Klasse einen erhöhten Wert im Verhältnis zu den anderen Klassen, was auf eine komplizierte Verschachtelungsstruktur in dieser Klasse hindeutet.

Bei der Klassenkopplung besitzt unter allen Klassen *PrototypWeb.API.DataAccesor* mit Abstand den höchsten Wert. Dies deutet wieder auf eine schlechte Einhaltung des Prinzips *Separation of Concerns* bei dieser Klasse hin. Die Datenhaltungsklassen haben bei dieser Code-Metrik wie zu erwarten sehr niedrige Werte.

5.3.6 Beschreibung der Architektur

Unter Berücksichtigung der durchgeführten Architekturanalysen ergibt sich die in Abbildung 5.18 dargestellte Architekturbeschreibung des Prototyps in der Version vom 23.08.2016. Diese ist, wie in [22] empfohlen, möglichst einfach gehalten. Deshalb wird bei der Notation dafür auf die Verwendung von Standards wie *UML* verzichtet. Ein weiterer Grund ist, dass die später folgende Visualisierung der Szenariointeraktionen auf dieser Darstellung aufbaut und dort der Flächeninhalt der abgebildeten Komponenten variabel ist und eine wichtige Rolle einnimmt.

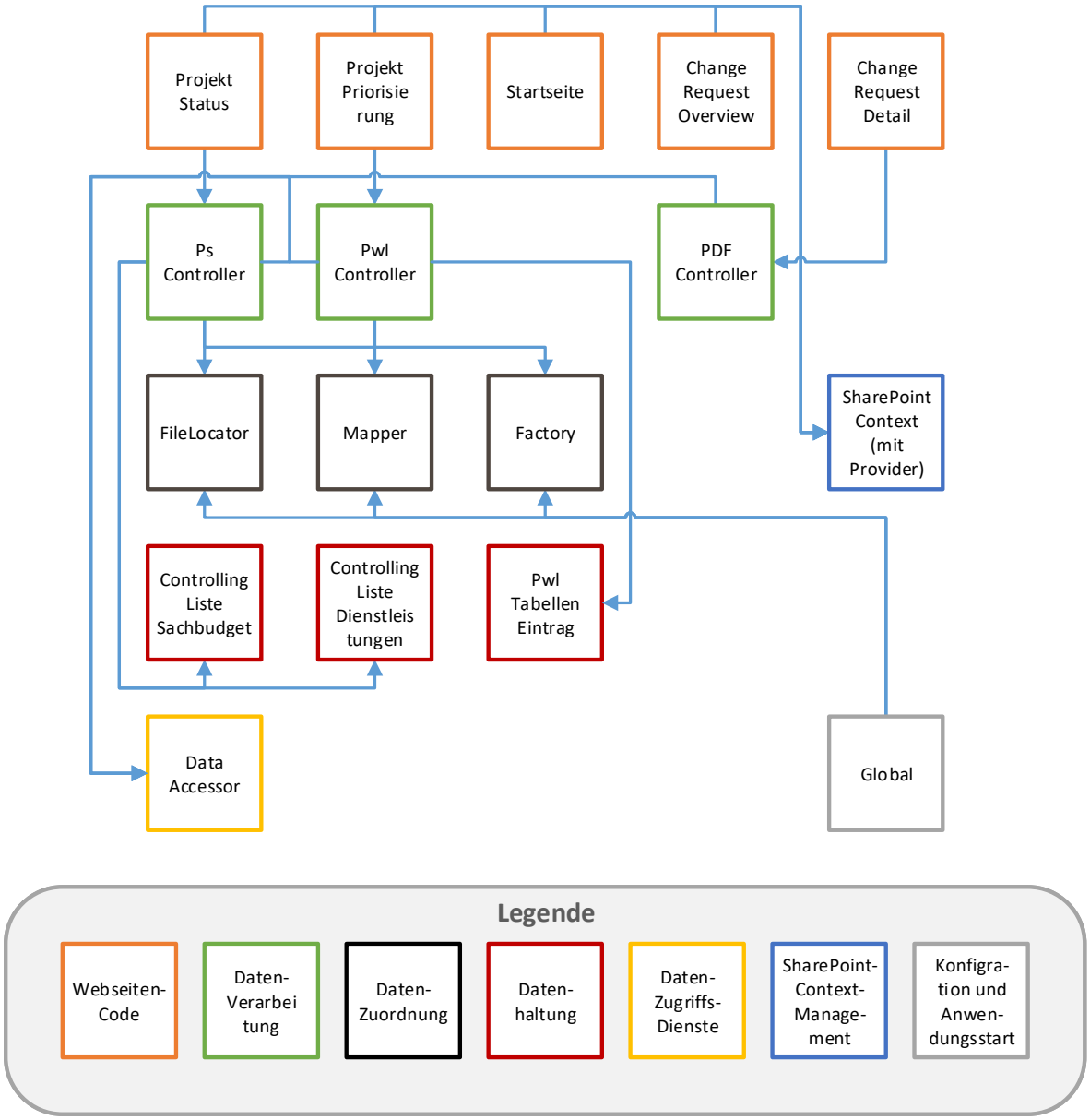


Abbildung 5.18: Architekturbeschreibung für SAAM

Die Quadrate stehen hierbei für die Komponenten und die Pfeile für Aufrufsbeziehungen. Die Zuordnung der Klassen zu diesen Komponenten ist in Tabelle 5.3 angegeben.

Tabelle 5.3: Zuordung von Klassen zu Komponenten

Komponente	Zugeordnete Klassen
Projektstatus	PrototypWeb.Pages.ProjektStatus
ProjektPriorisierung	PrototypWeb.Pages.Projektpriorisierung
Startseite	PrototypWeb.Pages.Startseite
ChangeRequestOverview	PrototypWeb.Pages.ChangeRequestOverview
ChangeRequestDetail	PrototypWeb.Pages.ChangeRequestDetail
PsController	PrototypWeb.Controllers.PsController
PwlController	PrototypWeb.Controllers.PwlController
PDFController	PrototypWeb.Controllers.PDFController
FileLocator	PrototypWeb.Controllers.FileLocator
Mapper	PrototypWeb.Controllers.Mapper
Factory	PrototypWeb.Controllers.Factory
PwlTabellenEintrag	PrototypWeb.Models.PwlTabellenEintrag
ControllingListeSachbudget	PrototypWeb.Models.ControllingListeSachbudget
ControllingListeDienstleistung	PrototypWeb.Models.ControllingListeDienstleistung
DataAccessor	PrototypWeb.API.DataAccesor
SharePointContext (mit Provider)	PrototypWeb.SharePointContext, PrototypWeb.RedirectionStatus, PrototypWeb.SharePointContextProvider, PrototypWeb.SharePointAcsContext, PrototypWeb.SharePointAcsContextProvider, PrototypWeb.SharePointHighTrustContext, PrototypWeb.SharePointHighTrustContextProvider, PrototypWeb.TokenHelper, PrototypWeb.SharePointContextToken, PrototypWeb.MultipleSymmetricKeySecurityToken
Global	PrototypWeb.Global

5.4 SAAM Schritt 2 - Entwicklung von Szenarien

In diesem Schritt werden die Szenarien für die Abbildung auf die Architekturbeschreibung entwickelt.

5.4.1 Identifizierung der Stakeholder

Um zu überprüfen, ob alle Stakeholder beim Aufstellen von Anforderungen oder Szenarien berücksichtigt worden sind, müssen diese zunächst identifiziert werden.

Die folgenden Stakeholder spielen bei fast jedem Softwareprojekten eine bedeutende Rolle.

Interne Stakeholder:

- EntwicklerIn
- TesterIn
- MaintainerIn
- ArchitektIn

Bei der Untersuchung der vorhandenen Anforderungsdokumente sind folgende Stakeholder identifiziert worden. Innerhalb der Auflistung sind Synonyme in Klammern angegeben.

Externe Stakeholder (Einzelpersonen):

- Chief Information Officer (CIO / BereichsleiterIn IT / BL IT)
- AbteilungsleiterIn (AL)
- ProjektleiterIn (PL)
- MitarbeiterIn im Bereich IT
- MitarbeiterIn vom Vertragsmanagement

Externe Stakeholder (Gruppen):

- Vorstand
- Change Advisory Board (CAB)
- Abteilung Projekt-/Prozessmanagement (PPM)
- Bereich IT
- Change-Antragsteller

Externe Stakeholder (indirekt beteiligt):

- Datenschutzbeauftragter
- Lenkungsausschuss im Bereich IT
- Projekt-Controlling
- Lenkungsgremien im Klinikum

Stakeholder-Map

Abbildung 5.19 zeigt die Stakeholder-Map, welche Vererbungsverhältnisse aufzeigt und die identifizierten Stakeholder ihren Aufgabenbereichen zuordnet.

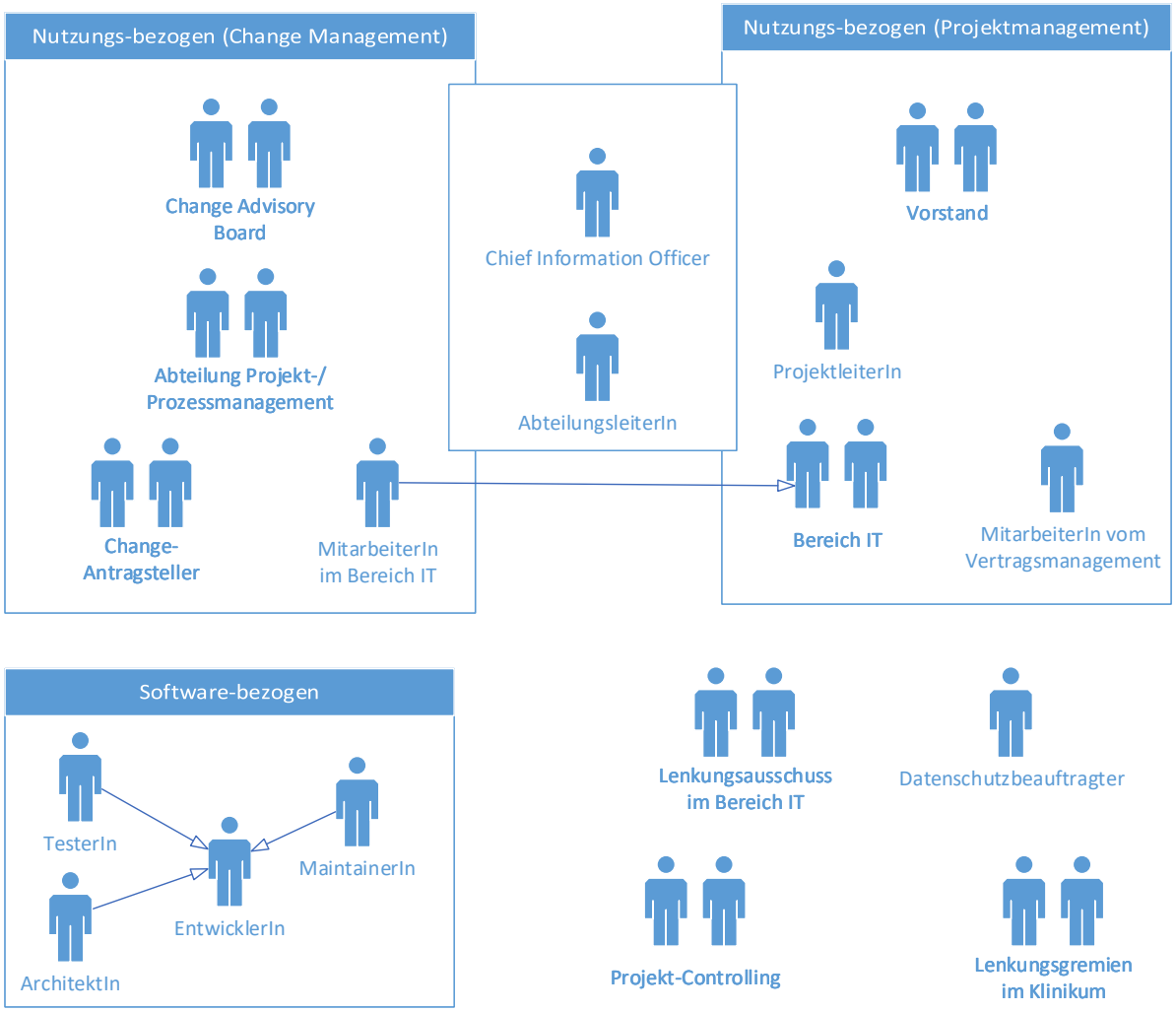


Abbildung 5.19: Einordung der Stakeholder mit Stakeholder-Map

Alle Personen in dieser Abbildung stehen für beteiligte Stakeholder. Ein abgebildeter Mensch stellt hier eine Einzelperson dar. Zwei direkt neben einander abgebildete Menschen eine Gruppe. Die Pfeile zeigen Vererbungsverhältnisse zwischen den Stakeholdern an. Die Person an der Pfeilspitze ist diejenige, von der dabei Eigenschaften geerbt werden. Die Rechtecke bilden eine grobe Zuordnung in Bezug auf die Aufgabenverteilung, wobei überlappende Rechtecke die Zugehörigkeit zu allen teilweise überdeckten Rechtecken darstellen.

5.4.2 Funktionale Anforderungen

Die in Tabelle 5.4 aufgelisteten funktionalen Anforderungen wurden aus den Ergebnissen des Requirements-Engineering, welches im Rahmen des SNIK-Projektes durchgeführt wurde, abgeleitet. Diese Anforderungen wurden nach Prinzipien der aufgabenorientierten Softwareentwicklung (TORE und DsTORE) ermittelt und dokumentiert. Die zugrunde liegenden Artefakte sind Interviewprotokolle, sowie Task- und Subtask-Beschreibungen der Stakeholder. Die funktionalen Anforderungen sind der Ausgangspunkt für die Erstellung und Überprüfung der Funktionalitäten. Damit kommt ihnen dabei eine große Bedeutung zu.

Tabelle 5.4: Funktionale Anforderungen an den CIO-Naviagtor nach Stakeholdern

ID	Stakeholder	Funktionale Anforderung
F1	Chief Information Officer	Festlegung der Reihenfolge von Projekten
F2	Alle nutzungsbezogenen Stakeholder	Zeigen einer Übersicht mit Darstellung der kombinierten Projekt- und Projektwarteliste
F3	Alle nutzungsbezogenen Stakeholder	Sortiermöglichkeiten und Filtermöglichkeiten für die neu gestaltete kombinierte Projekt- und Projektwarteliste
F4	Chief Information Officer, Vorstand	Alle Informationen für eine Repriorisierung der Projekte in der Warteliste, in Abstimmung mit dem Vorstand, werden in einer Sicht dargestellt.
F5	Chief Information Officer, Vorstand	Bei der Aufnahme neuer Projekte in die Warteliste wird ein Bericht an den Vorstand über diese Projekte im Format der Projektwarteliste erstellt
F6	Chief Information Officer	Änderungen an Projekten der Projektwarteliste können direkt in der Übersicht durchgeführt werden
F7	Chief Information Officer	Bei Änderungen an Projekten der Projektwarteliste wird eine Änderungsprotokoll erstellt
F8	Alle nutzungsbezogenen Stakeholder	Darstellung des Projektstatus mehrerer Projekte in einer zentralen Sicht
F9	Alle nutzungsbezogenen Stakeholder	Darstellung des Status der laufende Projekte mit Ampelstatus unter Einbeziehung der Parameter Zeit, Fortschritt, Budget, Umfang/umgesetzte Arbeitspakete usw.
F10	ProjektleiterIn	Der Status der Projekte wird durch den Projektleiter gesetzt
F11	ProjektleiterIn, AbteilungsleiterIn, MitarbeiterIn vom Vertragsmanagement, Chief Information Officer	Öffnen der Projekt-Webseite in der Übersicht über den aktuellen Status des Projekts
F12	Chief Information Officer, ProjektleiterIn	Anzeige von Projektstatus und zugehörigen relevanten Projektinformationen an einer Stelle. Zu den Projektinformationen zählen unter anderem der Projektplan als Ort von Statusverletzungen, der aktuelle Stand des Budgets und die Anzeige des Projektauftraggebers als Adressat für Eskalationen
F13	Chief Information Officer, ProjektleiterIn	Darstellung eines Gantt-Diagramms über einen Datumsausschnitt. Darin sind die laufenden Vorgänge des Projekts mit ihrem Zieldatum als Tabelle sichtbar
F14	Change Advisory Board, Chief Information Officer	Ansicht von Change-Anträgen mit Markierung noch fehlender Informationen

Tabelle 5.4: Funktionale Anforderungen an den CIO-Naviagtor nach Stakeholdern

ID	Stakeholder	Funktionale Anforderung
F15	Change Advisory Board, AbteilungsleiterIn, Abteilung Projektmanagement/Prozessmanagement, Chief Information Officer	Anzeige laufender Change-Anträge, welche noch abzuklären sind
F16	Change Advisory Board, MitarbeiterIn im Bereich IT, AbteilungsleiterIn	Prüfen des Vorhandenseins der Change-Bewertung und der Abteilungsleiter-Statements. Entscheidung über Wiedervorlage, wenn Statements oder Change-Bewertung nicht vorliegen
F17	Change Advisory Board, MitarbeiterIn im Bereich IT, AbteilungsleiterIn	Anzeige aller zu einem bestimmten Change-Antrag vorhandenen Change-Bewertungen und Statements in einer Sicht
F18	Chief Information Officer	Anzeige des Bereichsbudgets mit Darstellung des Restbudgets des laufenden Jahres und allokierte Kosten der Folgejahre von heute bis 4 Jahre in die Zukunft
F19	Change Advisory Board	Anzeige des Budgets zum Prüfen des Fortschritts im Rahmen des Controllings bei der Abarbeitung von Changes im operativen Bereich
F20	Change Advisory Board	Eintragung von Projekten, welche aus Changes hervorgehen in der noch unverbindlichen Projektwarteliste für das Folgejahr
F21	Change Advisory Board	Abschließen eines Change-Antrags. Kann beispielsweise durch Aufnahme in ein bestehendes Projekt erfolgen
F22	Change Advisory Board, Chief Information Officer, Change-Antragsteller	Erstellung des Mitteilungstextes zum Ergebnis eines Change-Antrags über ein entsprechendes Template. Der Antrag wird dabei mit weiteren nötigen Informationen wie Urheber und Titel versehen
F23	Chief Information Officer, Change-Antragsteller	Versenden des Resultates eines Change-Antrags an den Antragsteller per E-Mail

Die grau hinterlegten Anforderungen F5, F6, F7, F10, F11, F13, F20, F21, F22 und F23 werden im Rahmen des Prototyps des CIO-Navigators nicht umgesetzt. Dies liegt beispielsweise an der Entscheidung, dass dieser Daten ausschließlich anzeigen und nicht verändern soll. Auch wird zurzeit von diesem kein Workflow unterstützt.

5.4.3 Qualitätsanforderungen

Da es sich beim CIO-Navigator um einen Prototyp handelt, liegt der Schwerpunkt bei der Umsetzung auf den Funktionalitäten, welche durch die funktionalen Anforderungen beschrieben

werden. Für die Entwicklung der fertigen Software sind jedoch auch die Qualitätsanforderungen von großer Bedeutung. Deshalb sind diese hier aufgestellt, obwohl ein Großteil von ihnen in der betrachteten Version nicht umgesetzt wird.

Die in Tabelle 5.5 dargestellten Qualitätsanforderungen stammen aus der Dokumentation zum Projekt (leicht überarbeitet aufzufinden in Abschnitt A.3 und zu diesen einzeln referenziert durch Buchstaben in Klammern). Der Fokus des Requirement-Engineerings für den CIO-Navigator lag jedoch bisher nicht auf den Qualitätsanforderungen, so dass diese nur ansatzweise ermittelt wurden.

Tabelle 5.5: Qualitätsanforderungen an den CIO-Naviagtor aus Anforderungsdokument nach Stakeholdern

ID	Stakeholder	Qualitätsanforderung
Q1	Alle nutzungsbezogenen Stakeholder	Rollenunterstützung, um verschiedenen Funktionalitäten und Restriktionen für unterschiedliche Personengruppen zu ermöglichen (A)
Q2	Alle nutzungsbezogenen Stakeholder	Unterstützung der Anzeige von reduzierten bzw. ausgeblendeten Informationen (B)
Q3	Bereich IT	Verschiedene Sichten für unterschiedliche Rollen, unter anderem für Info-Terminal auf dem Gang (C)
Q4	MaintainerIn	Erweiterbarkeit um neue Ansichten für andere Nutzeraufgaben (D)
Q5	Chief Information Officer	Realisiert als Erweiterung von Microsoft SharePoint (E)
Q6	Alle nutzungsbezogenen Stakeholder	Realisierung als webbasiertes System zur unabhängigen Nutzung (F)
Q7	Chief Information Officer	Konfigurierbarkeit für veränderliche Datenquellen in Bezug auf Ort und Struktur (G)
Q8	Chief Information Officer	Integration möglichst vieler bestehender Systeme, wie SAP, Microsoft SharePoint, SCM (H)
Q9	Chief Information Officer	Unterstützung von verschiedenen heterogenen Infrastrukturen (I)
Q10	Datenschutzbeauftragter	Beachtung der Datenschutzbestimmungen bei Mitarbeiterdaten (J)
Q11	Alle nutzungsbezogenen Stakeholder	Aktualität der Daten: Projektaktivitäten: Tagesaktuell. SLA: Stundengenau. Service Desk: Ad hoc. (M)

Auch hier werden die grau hinterlegten Anforderungen Q1, Q3, Q8 und Q9 im Rahmen des Prototyps des CIO-Navigators nicht umgesetzt. Gründe hierfür sind ein zu hoher Aufwand für die Implementierung im Prototypen oder eine nachgelagerte Rolle der Anforderung.

Aus Interviewprotokollen ergeben sich zusätzlich die in Tabelle 5.6 aufgelisteten Qualitätsanforderungen.

Tabelle 5.6: Qualitätsanforderungen an den CIO-Naviagtor aus Interviewprotokollen nach Stakeholdern

ID	Stakeholder	Qualitätsanforderung
Q12	Alle nutzungsbezogenen Stakeholder	Darstellung der kombinierten Projekt- und Projektwarteliste soll schön sein
Q13	Alle nutzungsbezogenen Stakeholder	Schneller Informationsrückfluss über den Status der Projekte

Die grau hinterlegte Anforderung Q13 wird im Rahmen des Prototyps des CIO-Navigators nicht umgesetzt. Der Grund hierfür ist, dass dies außerhalb des Einflussbereichs der Anwendung liegt.

5.4.4 Aufstellung der Szenarien

Mit den in Tabelle 5.5 und Tabelle 5.6 angegebene Qualitätsanforderungen werden an dieser Stelle die in Tabelle 5.7 aufgelisteten Szenarien aufgestellt.

Tabelle 5.7: Szenarien aus Qualitätsanforderungen

ID	Abgedeckte Stakeholder	Szenario	Quelle
S1	Alle nutzungsbezogenen Stakeholder	Anmeldung mit einer Rolle, bei der die Budget-Daten nicht sichtbar sind	Anforderung Q1
S2	Alle nutzungsbezogenen Stakeholder	Ausblendung von im Moment nicht gewünschten Informationen	Anforderung Q2
S3	Bereich IT	Anzeige auf einem Info-Terminal in einem Gang im IT-Bereich	Anforderung Q3
S4	MaintainerIn	Erweiterung um eine komplett neue Ansicht für weitere unterstützende Funktionen	Anforderung Q4
S5	Chief Information Officer	Änderung des Ortes eines der zugrunde liegenden PDF-Dokumente	Anforderung Q7
S6	Chief Information Officer	Änderung der Informationen für das Auffinden eines Feldes in einer der zugrunde liegenden Excel-Tabelle	Anforderung Q7

Die Anzahl der hier auf diese Weise gewonnenen Szenarien ist für aussagekräftige Ergebnisse im weiteren Verlauf der Durchführung des SAAM-Verfahrens allerdings nicht ausreichend.

Daher werden weitere mögliche Quellen für Szenarien mit einbezogen. Das aus dem ATAM-Verfahren [23] stammende Konzept des Utility-Baums hat sich in diesem Rahmen als geeignet herausgestellt und wird daher zur Erstellung weiterer Szenarien übernommen. Ein Utility-Baum ist das Resultat einer Technik zum generieren von Szenarien, bei der diese aus unterschiedlichen Qualitätseigenschaften abgeleitet werden. Dabei werden außerdem die Wichtigkeit und Schwierigkeit der Umsetzung der einzelnen Szenarien bewertet. Der Utility-Baum kann so lange erweitert werden, bis der gewünschte Umfang an Szenarien erreicht ist. In Abbildung 5.20 ist der aufgestellte Utility-Baum abgebildet.

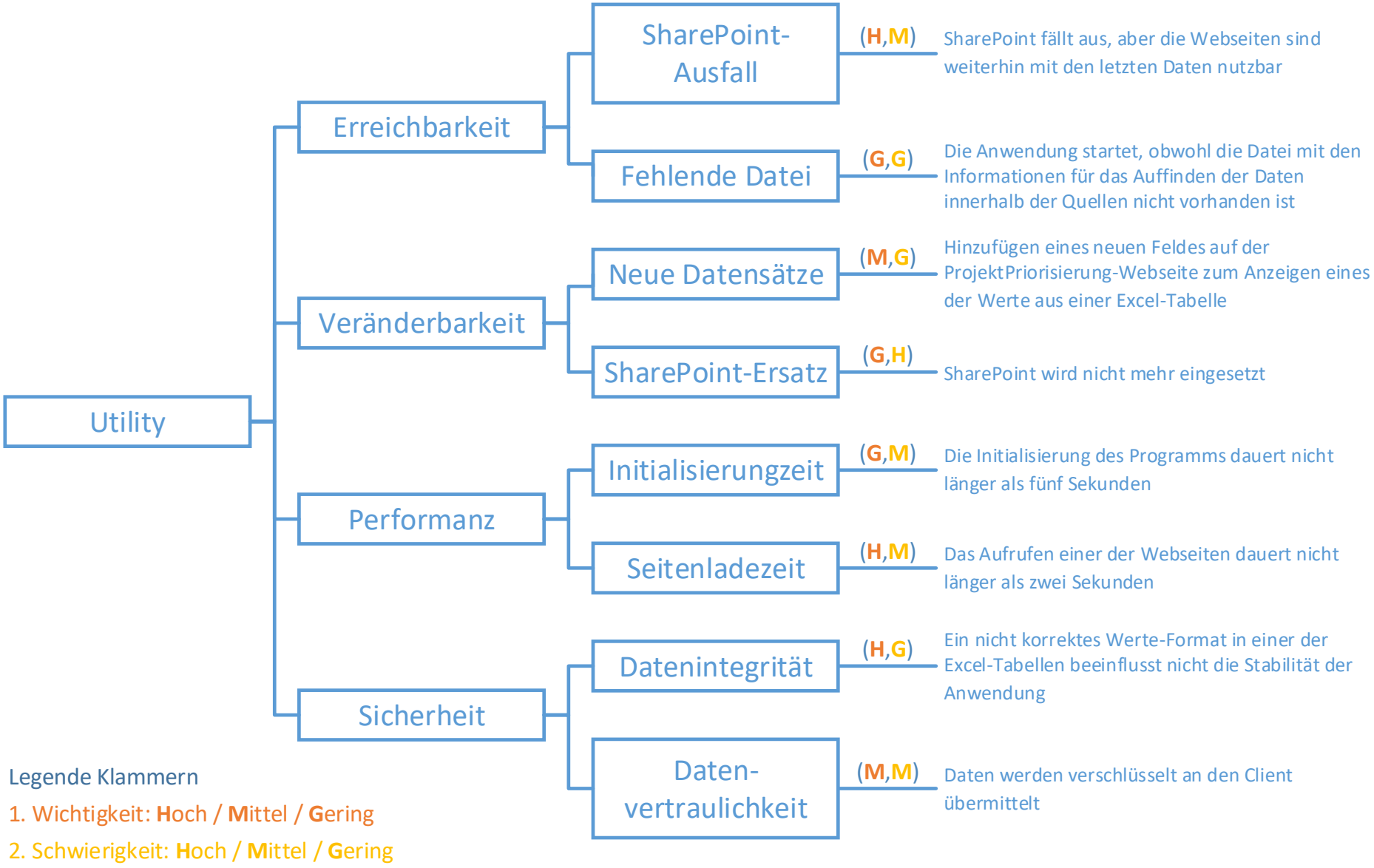


Abbildung 5.20: Utility-Baum nach ATAM

Aus dem Utility-Baum ergeben sich die in Tabelle 5.8 aufgelisteten weiteren Szenarien.

Tabelle 5.8: Szenarien aus Utility-Baum

ID	Abgedeckte Stakeholder	Szenario	Quelle
S7	Alle nutzungsbezogenen Stakeholder	SharePoint fällt aus, aber die Webseiten sind weiterhin mit den letzten Daten nutzbar	Utility-Baum
S8	Alle nutzungsbezogenen Stakeholder	Die Anwendung startet, obwohl die Datei mit den Informationen für das Auffinden der Daten innerhalb der Quellen nicht vorhanden ist	Utility-Baum
S9	MaintainerIn	Hinzufügen eines neuen Feldes auf der *Projekt-Priorisierung*-Webseite zum Anzeigen eines Wertes aus einer Excel-Tabelle	Utility-Baum
S10	MaintainerIn	SharePoint wird nicht mehr eingesetzt	Utility-Baum
S11	Alle nutzungsbezogenen Stakeholder	Die Initialisierung des Programms dauert nicht länger als fünf Sekunden	Utility-Baum
S12	Alle nutzungsbezogenen Stakeholder	Das Aufrufen einer der Webseiten dauert nicht länger als zwei Sekunden	Utility-Baum
S13	Alle nutzungsbezogenen Stakeholder	Ein nicht korrektes Werteformat in einer der Excel-Tabellen beeinflusst nicht die Stabilität der Anwendung	Utility-Baum
S14	Datenschutzbeauftragter	Daten werden verschlüsselt an den Client übermittelt	Utility-Baum

5.5 SAAM Schritt 3 - Bewertung der Szenarien

Die aufgestellten Szenarien werden auf ihre Unterstützung durch die vorhandene Architektur überprüft. Wird ein Szenario ohne Änderungen an der Architektur nicht unterstützt (*indirektes Szenario*), so werden die nötigen Änderungen für die Ermöglichung des Szenarios ermittelt. Diese können mit der Abänderung vorhandener Komponenten oder der Einführung neuer Komponenten einhergehen. Die Ergebnisse der Untersuchung sind in Tabelle 5.9 aufgeführt.

Tabelle 5.9: Bewertung der Szenarien

Szenario	**Direkt / Indirekt**	**Erklärung / Nötige Änderungen**
S1	Indirekt	Die *DataAccessor*-Komponente muss so abgeändert werden, dass bei verschiedenen Rollen die Werte für das Budget gar nicht erst beschafft werden. Die *PsController*-Komponente und die *ProjektStatus*-Komponente müssen angepasst werden, so dass diese auch bei nicht vorhandenen Budget-Daten funktionieren.
S2	Direkt	Ist durch die Filterfunktion bereits realisiert.
S3	Direkt	Es muss eine regelmäßige Auffrischung integriert werden, damit die Anzeige auf dem Info-Terminal aktuell bleibt. Dies ist durch Anpassung der Webseiten ohne Änderung des Quellcodes möglich.
S4	Indirekt	Zunächst ist die Webseite für die neue Ansicht mit einer neuen eigenen Komponente zu versehen, wie diese auch die anderen Webseiten besitzen. Außerdem wird eine neue entsprechende Datenverarbeitungskomponente benötigt. Mindestens eine neue zugehörige Datenhaltungskomponente muss ebenfalls eingeführt werden.
S5	Direkt	Lässt sich bereits über die Datei mit den Zuordnungen zu den Quelldateien konfigurieren.
S6	Direkt	Lässt bereits sich über die Datei mit den Zuordnungen zu den Werten innerhalb der Quelldateien konfigurieren.
S7	Indirekt	Die Quelldaten müssen zwischengespeichert werden, damit bei einem Ausfall von SharePoint wenigstens die zuletzt geholten Daten noch da sind. Dazu muss die *DataAccessor*-Komponente so angepasst werden, dass diese die Daten behält. Beim Zugriff auf diese Daten darf dabei allerdings die Prüfung der Berechtigung des Zugreifenden nicht vergessen werden. Diese muss ebenfalls in der *DataAccessor*-Komponente ergänzt werden.
S8	Indirekt	Die Datenverarbeitungskomponenten, welche auf die Komponenten *FileLocator* und *Mapper* zugreifen, müssen angepasst werden um mit der Ausnahme, dass die Datei nicht nicht vorhanden ist, umzugehen.
S9	Indirekt	Die *ProjektPriorisierung*-Komponente muss so erweitert werden, dass sie das neue Feld befüllt. Außerdem muss der neue Wert in die *PwlTabellenEintrag*-Komponente eingefügt werden.
S10	Indirekt	Die *DataAccessor*-Komponente muss so verändert werden, dass sie für ihre Zugriffe kein SharePoint mehr benötigt. Die *SharePointContext*-Komponente kann ganz gestrichen werden.
S11	Indirekt	Die aktuelle Zeit für das Initialisieren beträgt sieben Sekunden. Es gibt keine Möglichkeit das Initialisieren durch Veränderung der Komponenten maßgeblich zu beschleunigen, da dieses von der Umgebung der Webanwendung abhängt.
S12	Indirekt	Die Ladezeit kann durch die zeitaufwendigen Excel-Zugriffe in der *DataAccessor*-Komponente nicht für alle Webseiten erreicht werden. Der Umstand könnte möglicherweise durch die Nutzung einer anderen Bibliothek für diese Excel-Zugriffe behoben werden.
S13	Indirekt	Zur stabilen Behandlung dieser Ausnahme muss die *Factory*-Komponente um weitere Logik zum Prüfen ergänzt werden.
S14	Direkt	Ist bereits so konfiguriert.

5.6 SAAM Schritt 4 - Untersuchung von Szenariointeraktionen

In Tabelle 5.10 sind die einzelnen vorhandenen oder noch benötigten Komponenten aufgelistet. Zu diesen sind die Szenarien, welche Änderungen an den entsprechenden Komponenten mit sich bringen oder zu den entsprechenden neuen Komponenten führen, angegeben.

Tabelle 5.10: Szenario-Interaktionen

Komponente	**Änderung bei Szenario**	**Anzahl der Änderungen**
Projektstatus	S1	1
ProjektPriorisierung	S9	1
Startseite	-	0
ChangeRequestOverview	-	0
ChangeRequestDetail	-	0
PsController	S8	1
PwlController	S8	1
PDFController	-	0
FileLocator	-	0
Mapper	-	0
Factory	S13	1
PwlTabellenEintrag	S9	1
ControllingListeSachbudget	S1	1
ControllingListeDienstleistung	-	0
DataAccessor	S1, S7, S10, S12	**4**
SharePointContext (mit Provider)	S10	1
Global	-	0
Neuer Webseiten-Code A	S4	1
Neue Datenverarbeitung A	S4	1
Neue Datenhaltung A	S4	1

Es ist zu erkennen, dass die benötigten Änderungen für die *indirekten Szenarien* sich gleichmäßig auf die verschiedenen Komponenten verteilen. Lediglich auf die *DataAccessor*-Komponente werden mehrere Szenarien abgebildet. Die Abbildung mehrerer Szenarien auf die selbe Komponente wird als Szenariointeraktion bezeichnet. Möglichen Bedeutungen von vorhandenen oder nicht vorhandenen Szenariointeraktionen sind in der im folgenden Schritt erstellten Gesamtbewertung angegeben.

Abbildung 5.21 zeigt eine Übersicht über die Komponenten, welche im Rahmen der Realisierung der Szenarien verändert werden müssen. Die Größe der Fläche gibt hierbei das Ausmaß der Szenariointeraktionen an.

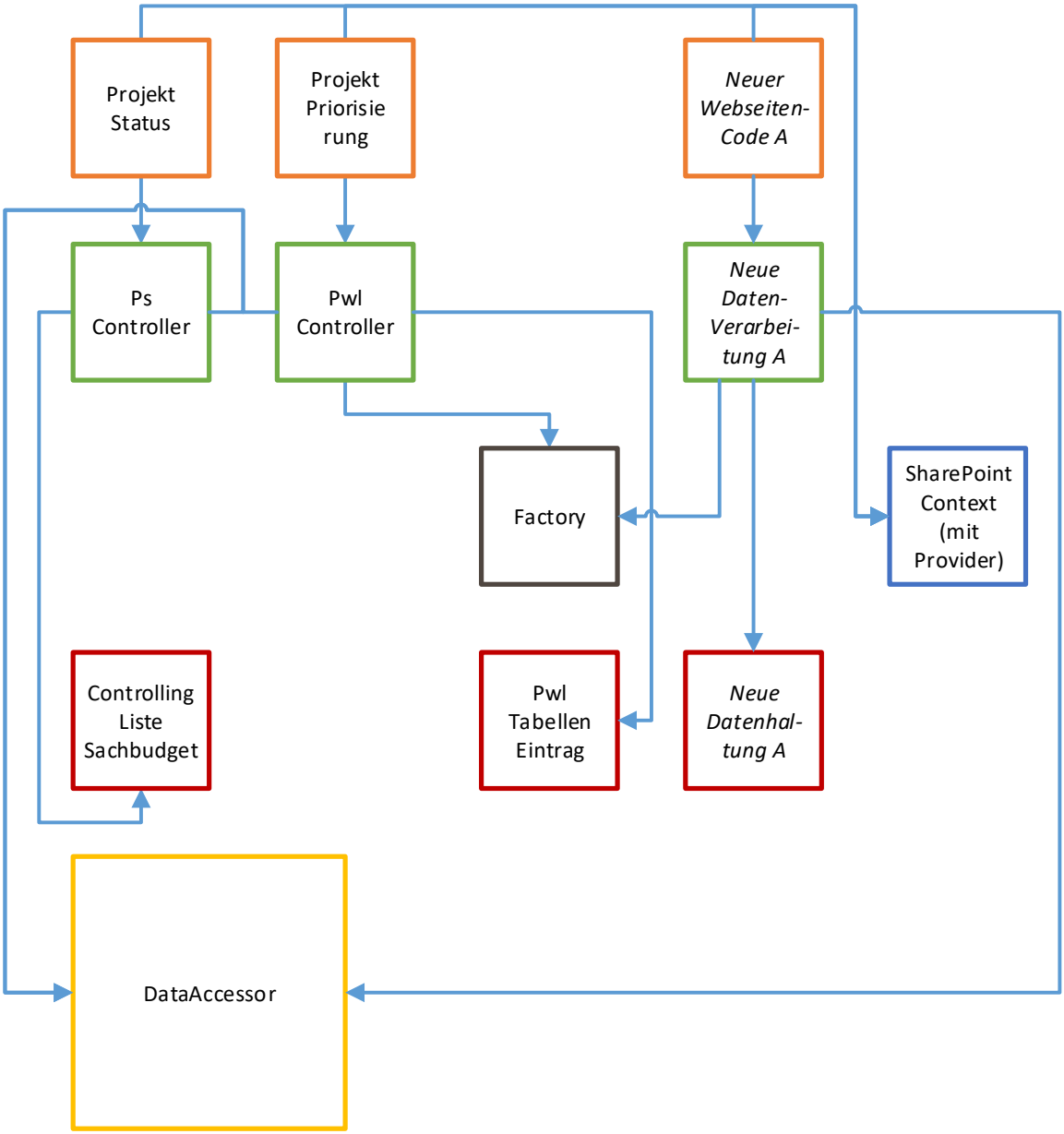


Abbildung 5.21: Szenariointeraktionen abgebildet auf die Architektur

Die Legende zur Abbildung ist die gleiche wie in Abbildung 5.18. Komponenten mit kursiver Beschriftung sind neu einzuführende Komponenten. Der Flächeninhalt der Komponenten ist hier äquivalent zur vorher ermittelten Anzahl der darauf abgebildeten Szenarien und damit zu den Szenariointeraktionen.

5.7 SAAM Schritt 5 - Erstellung der Gesamtbewertung

In diesem Abschnitt ist die Einordnung der *indirekten Szenarien* nach ihrer Wichtigkeit durchgeführt, welche in Tabelle 5.11 dargestellt ist. Zur Einschätzung der Wichtigkeit wurde ein Vertreter der Stakeholder zur Bedeutung der Szenarien und der Wahrscheinlichkeit ihres Eintretens befragt. Die Ergebnisse dieser Befragung sind bei der Begründung in Klammen mit angegeben.

Tabelle 5.11: Einordnung der indirekten Szenarien nach ihrer Wichtigkeit

Position	Szenario	Begründung
1	S8	Das Programm darf nicht abstürzen wenn Probleme mit der Datei, welche die Informationen für das Auffinden der Daten innerhalb der Quellen enthält, vorhanden sind. Eine entsprechende Meldung muss dann angezeigt werden. (sehr wichtig, wahrscheinlich)
2	S1	Bei manchen Nutzungen, beispielsweise beim geplanten Info-Terminal auf dem Gang, dürfen Budget-Daten keinesfalls angezeigt werden, da sonst Einschränkungen für die Weitergabe dieser Informationen nicht beachtet würden. (wichtig, wahrscheinlich)
3	S13	Da die zugrundeliegenden Excel-Daten manuell geändert werden, ist die Wahrscheinlichkeit einer Veränderung der Struktur hoch. Mit solch einer Änderung muss das Anwendungssystem umgehen können. Es darf nicht die Stabilität beeinflusst werden. (wichtig, wahrscheinlich)
4	S9	Änderungswünsche in Bezug auf die Anzeige weiterer Informationen durch das Anpassen von Ansichten sind wahrscheinlich und nehmen daher einen relativ hohen Stellenwert ein. (wichtig, wahrscheinlich)
5	S12	Die Aufrufdauer der Webseiten ist für die geplante Nutzung noch angemessen. (weniger wichtig, sehr wahrscheinlich)
6	S11	Die Anwendung wird selten neu initialisiert und dies findet in angemessener Zeit statt. (weniger wichtig, sehr wahrscheinlich)
7	S4	Änderungswünsche in Bezug auf die Anzeige weiterer Informationen durch die Einführung komplett neuer Anzeigen sind wahrscheinlich und sind daher in Betracht zu ziehen. (weniger wichtig, wahrscheinlich)
8	S7	Im Falle eines Ausfalls wäre es angenehm, wenn die Anwendung mit den letzten erhaltenen Daten weiterhin funktionieren würde. (weniger wichtig, unwahrscheinlich)
9	S10	Der Fall, dass SharePoint nicht mehr eingesetzt wird jedoch die Anwendung weiterhin ist sehr unwahrscheinlich. (unwichtig, unwahrscheinlich)

Ebenso ist für die Szenariointeraktionen eine Einordnung nach Anzahl der interagierenden Szenarien durchgeführt, welche in Tabelle 5.12 gezeigt wird.

Tabelle 5.12: Einordnung der Szenariointeraktionen nach Anzahl der interagierenden Szenarien

Position	Komponente der Szenariointeraktion
1	DataAccessor
2	PsController
3	PwlController
4	Factory
5	Projektstatus
6	ProjektPriorisierung
7	PwlTabellenEintrag
8	ControllingListeSachbudget
9	SharePointContext (mit Provider)
10	Neue Datenverarbeitung A
11	Neuer Webseiten-Code A
12	Neue Datenhaltung A

Eine Betrachtung der *direkten Szenarien* zeigt des Weiteren, dass vor allem die Bedürfnisse der BereichsleiterIn der IT und des Datenschutzbeauftragten bereits gut abgedeckt sind. Die Szenarien welche den MaintainerInnen zuzuordnen sind, verursachen nötige Änderungen an vergleichsweise vielen Komponenten.

Die sehr gleichmäßige Verteilung der Szenarien auf die verschiedenen Komponenten deutet generell auf eine gute Einhaltung des Prinzips *Separation of Concerns* hin. Szenariointeraktionen mit verschiedenartigen Szenarien, welche ein Hinweis auf eine schlechte Einhaltung dieses Prinzips sind, finden sich nur bei der *DataAccessor*-Komponente. Diese Komponente sollte daher aufgeteilt werden.

Abschließend kann durch die Auswertung der Szenarien die Aussage getroffen werden, dass die Stabilität der Anwendung im Vordergrund steht. Ebenfalls sehr wichtig ist die Datensicherheit einzustufen. Bedeutsam ist auch die Einarbeitung von Änderungswünschen. Schließlich ist eine ausreichende Performanz der Anwendung ein nicht zu vernachlässigender Faktor. Aus den Ergebnissen zu den Szenariointeraktionen kann geschlossen werden, dass die Komponente *DataAccessor* überdacht und entsprechend angepasst werden sollte.

6 Ergebnisse der Architekturbewertung und Bewertung

In diesem Kapitel werden auf der Basis der zuvor gesammelten Informationen Verbesserungsvorschläge für die Softwarearchitektur des CIO-Navigators gegeben. Die Verknüpfungen von den gegebenen Empfehlungen zu den zugrunde liegenden Architekturentscheidungen und Anforderungen gestalten sich übersichtlich und werden in den Ausführungen angegeben. Zum Abschluss dieses Kapitels findet eine Bewertung statt.

6.1 Angestrebte Komponentenarchitektur

Unter Berücksichtigungen der bisherigen Empfehlungen für Änderungen an der vorliegenden Architektur wird an dieser Stelle eine Anpassung der grundlegende Architektur (siehe Abbildung 5.2) nach dem in Abbildung 6.1 dargestellten Komponentendiagramm vorgeschlagen.

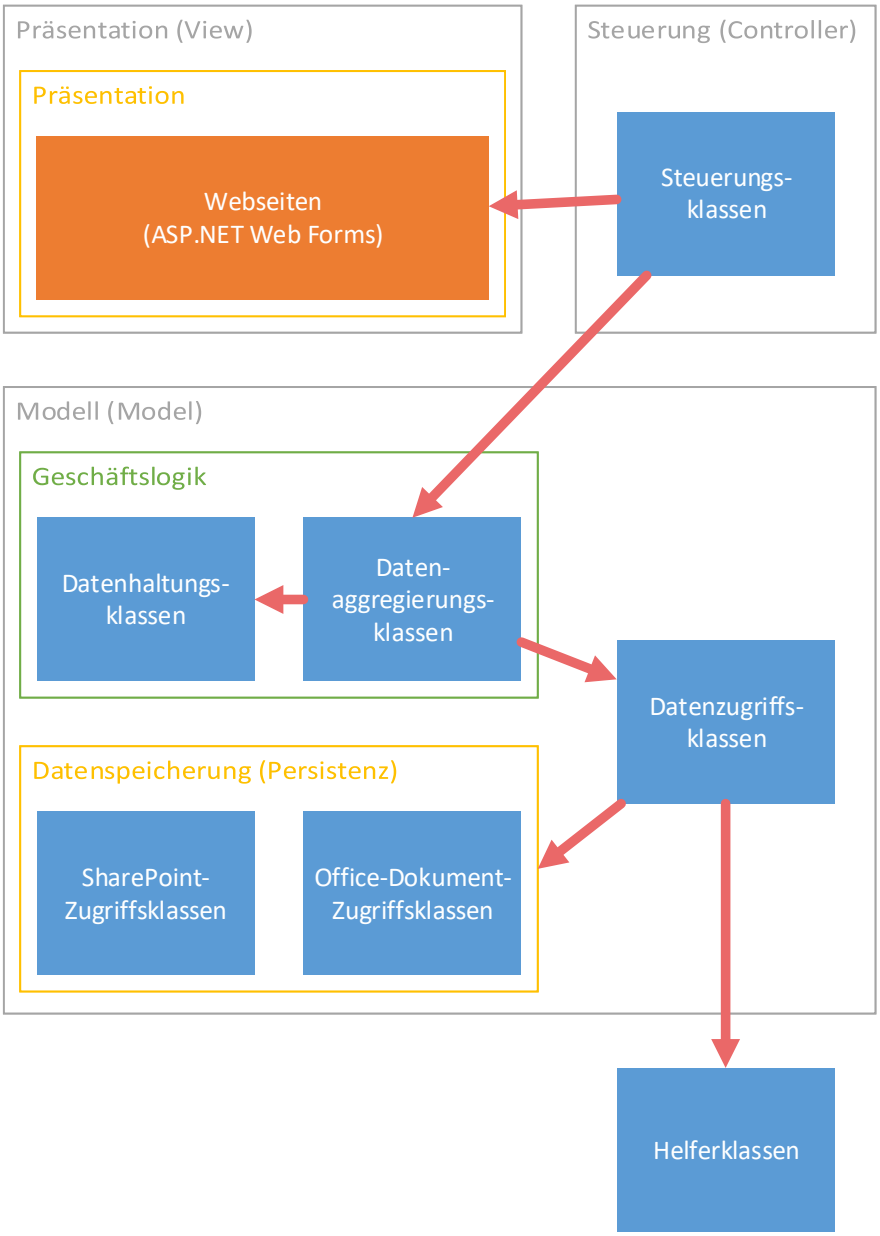


Abbildung 6.1: Grundlegende Zielarchitektur mit Komponenten und Beziehungen

Das Diagramm zeigt den angestrebten Aufbau der Komponentenarchitektur. Die Bedeutung der Elemente und Beziehungen ist analog zu der in Abbildung 5.2. Es existieren nun Steuerungsklassen, welche Benutzeranfragen verarbeiten. Die Beschaffung der Daten für die Präsentation, ursprünglich im hinterlegten Code für die Webseiten angestoßen, wird nun durch die Steuerungsklassen ausgeführt. Der restliche Quellcode zu den Webseiten wird von entsprechenden Anweisungen innerhalb der Webseiten abgelöst. Der Modelleinheit zugerechnet wird nun die gesamte Geschäftslogik, mit den Klassen für das Aggregieren der Daten und den Datenhaltungsklassen. Außerdem sind dort auch die Klassen, welche der Persistenz zugehörig sind, eingeordnet. Der Zugang zu diesen Klassen erfolgt über die sich ebenfalls in der Modelleinheit befindenden Datenzugriffsklassen. Einige Helferklassen werden keiner Einheit des Model-View-Controller-Entwurfsmusters zugerechnet. Die Steuerungseinheit greift zur Verarbeitung von

Anfragen auf die Modelleinheit zu und übergibt die erhaltenen Daten an die Präsentationseinheit. Dieser Aufbau und diese Zugriffsabhängigkeiten sind nun konform zur hier angestrebten Form des Model-View-Controller-Entwurfsmusters.

6.2 Angestrebtes Model-View-Controller-Entwurfsmuster

Die vorliegende Architektur nutzt das Framework *ASP.NET Web Forms*. Für eine bessere Unterstützung des Model-View-Controller-Entwurfsmusters wird hier zusätzlich der Einsatz von *ASP.NET MVC* empfohlen. Dieses Framework ist mit *ASP.NET Web Forms* zusammen verwendbar und stellt Funktionalitäten für die Verwendung des Model-View-Controller-Entwurfsmusters bereit. Der Aufwand, *ASP.NET MVC* in eine Anwendung mit *ASP.NET Web Forms* nachträglich einzubauen, ist nicht groß.

In *ASP.NET MVC* werden beim Aufruf der Steuerung die benötigten Daten aus dem Modell abgerufen und an die Präsentation gesendet. Damit werden die Daten aus dem Modell indirekt an die Präsentation übergeben. Außerdem können Eingaben auch unter Verwendung der Präsentationslogik getätigt werden, von wo sie direkt an das Modell übergeben und dort gespeichert werden. Das resultierende System ist in Abbildung 6.2 gezeigt:

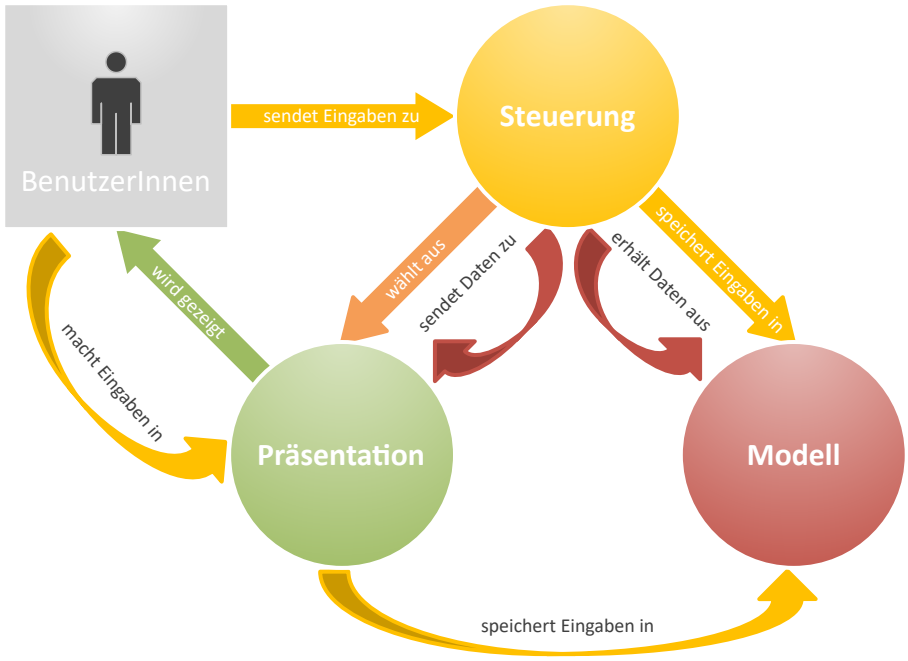


Abbildung 6.2: Spezifisches Model-View-Controller-Entwurfsmuster

Die Bedeutung der Elemente und Beziehungen ist analog zu Abbildung 5.3, welche den vorgefundenen Zustand des Model-View-Controller-Entwurfsmusters darstellt.

6.2.1 Umstellung auf ASP.NET MVC

Im Rahmen dieser Arbeit ist die Machbarkeit einer Umstellung des Prototyps in der Version vom 23.08.2016 auf das *ASP.NET MVC*-Framework evaluiert. Durch die Lauffähigkeit von Teilen

des Prototyps unter Verwendung von *ASP.NET MVC* ist die Durchführbarkeit dieser Umstellung demonstriert worden. Aufgrund dessen wird die Verwendung von *ASP.NET MVC* empfohlen.

Ein Ergebnis aus dieser Umstellung ist außerdem, dass bei der Nutzung von *ASP.NET MVC* das Format der Webseiten *ASP.NET Web Forms* (mit Dateiendung *.aspx*) weiterhin verwendet werden kann und nicht zwingend auf das aktuellere Format *ASP.NET Razor* (mit Dateiendung *.cshtml* in C#) umgestellt werden muss. Unter anderem müssen jedoch kleine Anpassungen beim hinterlegten Code zu den Webseiten durchgeführt werden.

6.3 Empfohlene Anordnung der Klassen in den Namensräumen

Eines der Ziele der Anordnung von Klassen in Namensräumen ist die Aufteilung von Klassen nach Zusammengehörigkeit. Die Vereinfachung von Beziehungen zwischen diesen Namensräumen hat sich als hilfreich herausgestellt. Diese Beziehungen werden dann übersichtlich und einfach zu überprüfen.

Anhand der Untersuchungen wird die in Abbildung 5.8 dargestellte Einordnung der Klassen in die angegebenen Namensräume empfohlen. Die Klasse *Global* stellt Funktionalitäten bereit, welche aus dem Namensraum *PrototypWeb.Models* heraus genutzt werden und macht so die Beziehungen zwischen den beteiligten Namensräumen sehr komplex. Daher wird vorgeschlagen, dass diese Funktionalitäten in eine eigenen Klasse, welche von der Klasse *Global* aufgerufen wird, ausgelagert wird.

6.4 Empfehlungen aus der Durchführung von SAAM

Die Architekturbewertung mit dem SAAM-Verfahren lieferte folgende Ergebnisse: Fünf von vierzehn Szenarien werden bereits ohne Änderungen an der Architektur unterstützt. Die Bedürfnisse der BereichsleiterIn der IT und des Datenschutzbeauftragten sind durch diese bereits gut abgedeckt. Die Szenarien der MaintainerInnen machen Änderungen an besonders vielen Komponenten nötig. Die Szenarien lassen sich gleichmäßig den Architekturkomponenten zuordnen, was auf eine gute Einhaltung des Prinzips *Separation of Concerns* hindeutet. Lediglich die Komponente *DataAccessor* bildet eine große Ausnahme, da bei vier Szenarien Änderungen an dieser Komponente nötig werden. Es wird empfohlen, diese Komponente sinnvoll nach Funktionalitäten aufzuteilen. So könnten unter andrem Zugriffe auf SharePoint und Office-Dokumente getrennt werden.

6.5 Empfehlungen aus der Analyse der Softwarearchitektur mit messbaren Eigenschaften des Quellcodes

Aus den Untersuchungen mit Hilfe von Code-Metriken (in Unterabschnitt 5.3.3) lassen sich folgende Schlussfolgerungen ziehen: Automatisch generierter Quellcode soll der Übersicht halber bei den Betrachtungen nicht berücksichtigt werden. Die Klasse *PrototypWeb.API.DataAccesor*

fällt gleich durch ihre große Anzahl an Codezeilen und ihren hohen Wert bei der Klassenkopplung auf, was ein Hinweis auf Probleme mit dem Prinzip *Separation of Concerns* ist. Anhand der zyklomatischen Komplexität ist außerdem zu erkennen, dass die Klasse *Prototyp-Web.Controllers.Mapper* eine komplizierte Verschachtelungsstruktur besitzt, welche aufgelöst werden sollte.

6.6 Bewertung

Ein Ziel dieser Arbeit ist, eine Architekturanalyse für den CIO-Navigator durchzuführen und Empfehlungen zur Optimierung der Architektur zu geben. Der CIO-Navigator ist ein Entscheidungsunterstützungssystem für das Informationsmanagement eines Krankenhauses. In diesem Rahmen wurden eine wissenschaftliche Literaturrecherche und eine ausführliche Evaluation des vorhandenen Prototyps des CIO-Navigators durchgeführt. Auf dieser Grundlage sind die zuvor vorgestellten Empfehlungen zur Umgestaltung der Architektur ermittelt worden. Damit berücksichtigen diese einerseits den aktuellen Stand der Wissenschaft auf diesem Gebiet und bauen andererseits auf einem schon teilweise lauffähigem System auf. Dadurch wird ein Bezug zur Praxis hergestellt.

Eine Literaturrecherche mit der in Abschnitt 3.1 beschriebene Forschungsfrage wurde nach wissenschaftlichen Standards durchgeführt und dokumentiert. Ein wichtiges Ergebnis ist, dass es zur Architektur von Entscheidungsunterstützungssystemen zurzeit noch wenige wissenschaftliche Erkenntnisse gibt. Dies schlägt sich auch in der eher geringen Anzahl an Veröffentlichungen zu dieser Thematik nieder. Dennoch konnten neben Besonderheiten von Entscheidungsunterstützungssystemen auch einige bereits veröffentlichte Architekturen in diesem Umfeld ausgemacht werden. Dabei lässt sich auf eine hohe Projektabhängigkeit schließen. Auch die inzwischen äußerst hohe Bedeutung von webbasierten Architekturen für solche Systeme wird deutlich. Außerdem wird wiederholt die Wichtigkeit von konsistenten und gut präsentierten Daten betont.

Bei der Analyse der Systemarchitektur zeigt sich, dass der Einsatz des Daten-Virtualisierung-Systems *Teiid* beim CIO-Navigator nicht zu empfehlen ist.

Die Analyse der Softwarearchitektur beginnt mit der Erfassung des vorgefundenen Aufbaus auf grundlegender Ebene. Danach wird die Softwarearchitektur genauer untersucht und damit die Beschreibung der Architektur im Architekturbewertungsverfahren weiter vorbereitet. Eine verwendete Methode der Architekturanalyse ist die Erstellung und Auswertung von Code-Metriken. Hier kann, vor allem auf Basis von Auffälligkeiten, schnell auf Eigenschaften oder Problematiken des Quellcodes geschlossen werden. Verfahren zur Visualisierung mit Software-Städten (Software Cities) können hierbei das Erfassen dieser Auffälligkeiten weiter beschleunigen. Eine wichtige Empfehlung aus der Untersuchung der Softwarearchitektur ist, erst die Strukturen der betrachteten Namensräume zu optimieren, bevor die Beziehungen zwischen diesen und den in diesen Namensräumen gelegenen Klassen untersucht werden. Auch die Unterstützung durch qualitativ hochwertige Werkzeuge hat sich im Zuge der Untersuchungen als äußerst hilfreich erwiesen.

Zur Unterstützung der zur Entwicklung der Szenarien gehörenden Anforderungsanalyse ist eine Aufstellung und Einordnung der ermittelten Stakeholder durchgeführt worden. Damit lässt sich zudem ein Bild vom personellen Umfeld, in dem der CIO-Navigator eingesetzt wird, gewinnen. Für die Anforderungsanalyse wird die Dokumentation des CIO-Navigators verwendet, die aufgrund ihrer hohen Qualität eine gute Quelle für Anforderungen darstellt. Ein Teil der Szenarien für die Architekturbewertung leitet sich aus den gefundenen Anforderungen ab.

Durch die systematische Vorgehensweise eines passendes Architekturbewertungsverfahrens werden insbesondere Aspekte, welche vergessen oder vernachlässigt worden sind, aufgedeckt. Für das hier gewählte SAAM-Verfahren kann dessen gute Eignung zur Analyse von Architekturen kleineren Umfangs bestätigt werden. Besonders die Einhaltung des Prinzips *Separation of Concerns* wird bei diesem genauer beleuchtet. Die Ergebnisse des Verfahrens werden in dieser Arbeit durch die Ergebnisse der damit einhergehenden Architekturanalysen bekräftigt, was das Vertrauen in dieses Verfahren erhöht.

Auf Grundlage der genannten Analysen werden auf Basis der Verbesserungsvorschläge im Rahmen dieser Arbeit (in Kapitel 6) Empfehlungen gegeben. Zusätzlich zu diesen konkreten Empfehlungen für den CIO-Navigator werden mit der Darlegung der Analysevorgänge und Bewertungsvorgänge Ansätze für die Untersuchung und Ausgestaltung gleichartiger Anwendungen angeboten.

7 Fazit

Entscheidungsunterstützungssysteme unterscheiden sich in ihrer Architektur nicht signifikant von anderen Informationssystemen und sind in ihrer Ausgestaltung in hohem Maße abhängig von den Anforderungen des jeweiligen Projekts. In der Literatur finden sich Hinweise auf Trends, wie die zunehmende Nutzung von Web-Technologien und den Einsatz von serviceorientierten Architekturen. Der Erfolg von Web-Technologien beruht auf den vielen Vorteilen, welche diese beim Einsatz in Entscheidungsunterstützungssystemen mit sich bringen. Zu diesen zählen beispielsweise das Anbieten von Funktionalitäten über das Internet, Plattformunabhängigkeit und die Begünstigung der Integration von veralteten Systemen. Damit sorgen diese, bei entsprechenden Anwendungen, insbesondere für eine längere Nutzbarkeit. Somit ist die Umsetzung des CIO-Navigators als webbasiertes System als gute Entscheidung zu bewerten.

Bei der Durchführung des Architekturbewertungsverfahrens SAAM wurde im Rahmen der beinhalteten Architekturanalyse folgendes festgestellt: Die Strukturierung in sinnvolle Komponenten ermöglicht nicht nur die einfache Durchführung von späteren Anpassungen und Änderungen, sondern unterstützt auch die Analyse der Softwarearchitektur. Strukturierter Quellcode ist besser aufgeteilt und abgegrenzt als nicht strukturierter. Auch die Struktur und die Beziehungen der Namensräume sind hilfreich für das Verständnis eines Systems, vor allem wenn die Beziehungen zwischen Namensräumen und Klassen werkzeuggestützt dargestellt werden können. Nach einer Verbesserung der Struktur sind die aussagekräftigen Gruppierungen und Beziehungen besser zu erkennen. Damit wird die untersuchte Architektur einfacher verständlich. Sogar beim gut durchdachten Prototyp des CIO-Navigators konnte bei einer genaueren Betrachtung der Softwarearchitektur Verbesserungspotential gefunden werden. So konnten einige Vereinfachungen vorgeschlagen sowie einige Auffälligkeiten ausgemacht werden. Auf dieser Grundlage kann die Struktur des Quellcodes weiter verbessert werden.

Im Zusammenhang mit der Untersuchung des Model-View-Controller-Entwurfsmusters beim Prototypen wird aufgrund der vielen Variationen bei der Ausgestaltung des Model-View-Controller-Entwurfsmusters festgestellt, dass dieses lediglich ein vages Konzept repräsentiert. Die Benennung, welche jedoch ein konkretes und eindeutiges Entwurfsmuster nahelegt, erweist sich somit als irreführend. Die Umsetzung, egal ob manuell oder durch ein entsprechendes Framework, kann höchst unterschiedlich ausgestaltet sein. Beim CIO-Navigator wird für die Umsetzung des Model-View-Controller-Entwurfsmusters der Einsatz des Frameworks *ASP.NET MVC* empfohlen. Dieses wäre nach dem geringen Aufwand der im Rahmen dieser Arbeit probeweise durchgeführten, teilweisen Umstellung zu schließen nicht besonders schwer zu implementieren.

Code-Metriken, als Zahlen oder in visualisierter Form, liefern schnelle Einblicke in unterschiedliche Eigenschaften der Architektur. So können beispielsweise Klassen mit besonders viel

Quellcode oder einer besonders komplizierten Verschachtelungsstruktur schnell ausfindig gemacht werden. Damit stellen diese Code-Metriken eine gute Ergänzung zu anderen Verfahren zur Architekturanalyse dar. Auffällig sind beim CIO-Navigator die Komponente für die Zuordnung der Werte aus den Quellen (*Mapper*) sowie die zentrale Komponente für den Datenzugriff (*DataAccessor*). Diese sollten daher unter Berücksichtigung von Aspekten guter Softwarearchitektur, wie beispielsweise *Separation of Concerns*, überdacht werden.

Architekturbewertungsverfahren bieten durch ihre systematische Vorgehensweise unter anderem eine Absicherung gegen nicht bedachte aber dennoch wichtige Aspekte von Architekturen. Zusätzlich beinhalten sie weitere Werkzeuge zur Untersuchung von Architekturen und tragen zu einer Verbesserung der Dokumentation bei. Auch die Abstimmung zwischen den Stakeholdern wird während der Durchführung des Verfahrens ermöglicht und unterstützt. Vor allem noch nicht bedachte Aspekte der System- und Softwarearchitektur, Stakeholder und Anforderungen können damit aufgedeckt werden, was das Vertrauen in die Qualität des CIO-Navigators weiter erhöht. Das ausgewählte SAAM-Verfahren im Speziellen liefert durch die Offenlegung von Szenariointeraktionen weitere Ergebnisse in Bezug auf die Qualität der Softwarearchitektur. Auf Grundlage dieser Ergebnisse muss beim CIO-Navigator eine Veränderung der Komponente *DataAccessor* in Betracht gezogen werden. Dies steht im Einklang mit den Resultaten aus der Analyse mit Codemetriken. Eine Anpassung bzw. Aufteilung dieser Komponente vereinfacht voraussichtlich eventuell folgende Änderungen oder Erweiterungen.

Die vorgeschlagenen Empfehlungen für die Optimierung der Architektur bauen auf den Ergebnissen der durchgeführten Analysen auf und besitzen damit eine, teilweise sogar mehrfach abgesicherte Grundlage. Die Vorgehensweise zur Ermittlung dieser Empfehlungen kann natürlich auf ähnliche Projekte übertragen werden, um dort eine vergleichbar hohe Qualität der Architektur der Software zu erhalten.

7.1 Ausblick auf zukünftige Arbeiten

Es wäre sicher aufschlussreich, die Architektur des CIO-Navigators nach Einarbeitung der Verbesserungsvorschläge erneut zu überprüfen. Damit könnte die Validität der gemachten Vorschläge und gezogener Schlussfolgerungen untersucht werden. Auch könnten die Ergebnisse verschiedener Architekturbewertungsverfahren gegenübergestellt werden, um vor allem Unterschiede zwischen diesen zu evaluieren.

Allgemein besteht nach Auswertung der Literaturrecherche im Bereich der Entscheidungsunterstützungssysteme weiterer Forschungsbedarf. Dieser äußert sich beispielsweise in der Vielzahl von unterschiedlichen veröffentlichten Methoden für die Implementierung von Entscheidungsunterstützungssystemen. Der anhaltende, schnelle Fortschritt bei den Web-Technologien birgt ebenfalls noch viel Potential für Erkenntnisgewinn, vor allem im Bereich der vermehrt zum Einsatz kommenden serviceorientierten Architekturen.

Die Analyseverfahren zur Softwarearchitekturbewertung sind dagegen seit längerem ausgereift. In diesem Bereich findet der Fortschritt zurzeit durch immer bessere Werkzeuge für die Analyse der Softwarearchitektur statt. Dies erhöht unter anderem die Geschwindigkeit dieses Prozesses und damit auch insbesondere der Softwareentwicklung. Inwiefern sich dies auf den Umfang und die Eigenschaften von Entscheidungsunterstützungssystemen auswirken wird, ist dabei noch nicht abzusehen.

Literaturverzeichnis

[1] ISO/IEC/IEEE Systems and software engineering – Architecture description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) pp. 1–46 (2011)

[2] Semantisches Netz des Informationsmanagements im Krankenhaus (2015), http://se.ifi.uni-heidelberg.de/research/projects/snik.html

[3] Codemetrikwerte: Visual Studio 2015 (2016), https://msdn.microsoft.com/de-de/library/bb385914.aspx

[4] Bayani, M.: Web-based Decision Support Systems: A conceptual performance evaluation. In: 2013 IEEE 17th International Conference on Intelligent Engineering Systems (INES). pp. 21–26 (2013)

[5] Behnam, S.A., Badreddin, O.: Toward a Care Process Metamodel: For Business Intelligence Healthcare Monitoring Solutions. In: Proceedings of the 5th International Workshop on Software Engineering in Health Care. pp. 79–85. SEHC '13, IEEE Press, Piscataway, NJ, USA (2013), http://dl.acm.org/citation.cfm?id=2663575.2663595

[6] Bhargava, H.K., Krishnan, R.: The World Wide Web: Opportunities for operations research and management science. INFORMS Journal on Computing 10(4), 359–383 (1998)

[7] Bhargava, H.K., Power, D.J., Sun, D.: Progress in Web-based decision support technologies. Decision Support Systems 43(4), 1083–1095 (2007), http://www.sciencedirect.com/science/article/pii/S0167923605001016

[8] Bianco, P., Lewis, G., Merson, P., Simanta, S.: Architecting Service-Oriented Systems (2011), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9829

[9] Boza, A., Ortiz, A., Cuenca, L.: A framework for developing a web-based optimization decision support system for intra/inter-organizational decision-making processes. In: Balanced Automation Systems for Future Manufacturing Networks, pp. 121–128. Springer (2010)

[10] Chen, H., Zhang, X., Chi, T.: An Architecture for Web-based DSS. In: Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems. pp. 75–79. SEPADS'07, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2007), http://dl.acm.org/citation.cfm?id=1353801.1353815

[11] Druzdzel, M.J., Flynn, R.R.: Decision Support Systems. In: Encyclopedia of Library and Information Science. Taylor & Francis, New York (2010)

[12] Eicker, S., Hegmanns, C., Malich, S.: Auswahl von Bewertungsmethoden für Softwarearchitekturen. ICB Research Report (14) (2007)

[13] Forgionne, G.A.: Decision-Making Support Systems: Achievements and Challenges for the New Decade: Achievements and Challenges for the New Decade. IGI Global (2002)

[14] Geoffrion, A., Maturana, S.: Generating optimization-based decision support systems. In: Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences. vol. 3, pp. 439–448. IEEE (1995)

[15] Gorry, G.A., Morton, M.S.: A framework for management information systems. Sloan Management Review 30(3), 49–61 (1989)

[16] Gräber, S., Ammenwerth, E., Brigl, B., Dujat, C., Große, A., Häber, A., et al.: Rahmenkonzepte für das Informationsmanagement in Krankenhäusern: Ein Leitfaden (2003)

[17] Haas, P.: Aspekte der Betriebs- und Managementunterstützung. In: Medizinische Informationssysteme und Elektronische Krankenakten, pp. 6–9. Springer-Verlag, Berlin, Heidelberg (2005)

[18] Haas, P.: Funktionale Kompetenz und Unterstützungsdimensionen. In: Medizinische Informationssysteme und Elektronische Krankenakten, pp. 49–60. Springer-Verlag, Berlin, Heidelberg (2005)

[19] Haux, R.: Health information systems – past, present, future. International Journal of Medical Informatics 75(3–4), 268–281 (2006), http://www.sciencedirect.com/science/article/pii/S1386505605001590

[20] ISO/IEC: ISO/IEC 25000:2014 - Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE (2014)

[21] Kawamoto, K., Lobach, D.F.: Proposal for fulfilling strategic objectives of the US roadmap for national action on decision support through a service-oriented architecture leveraging HL7 services. Journal of the American Medical Informatics Association 14(2), 146–155 (2007)

[22] Kazman, R., Abowd, G., Bass, L., Clements, P.: Scenario-based analysis of software architecture. IEEE software 13(6), 47–55 (1996)

[23] Kazman, R., Klein, M., Clements, P.: ATAM: Method for Architecture Evaluation (2000), http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=5177

[24] Kücherer, C., Jung, M., Jahn, F., Schaaf, M., Tahar, K., Paech, B., Winter, A.: System Analysis of Information Management. In: INFORMATIK 2015: Informatik, Energie und Umwelt. Lecture Notes in Informatics, vol. P-246, pp. 783–796 (2015)

[25] Lackes, R., Siepermann, M., Schewe, G.: Gabler Wirtschaftslexikon: Enterprise-Resource-Planning-System (2016), http://wirtschaftslexikon.gabler.de/Archiv/17984/enterprise-resource-planning-system-v12.html

[26] Lackes, R., Siepermann, M., Schewe, G.: Gabler Wirtschaftslexikon: Informationssystem (2016), http://wirtschaftslexikon.gabler.de/Archiv/9241/35/Archiv/9241/informationssystem-v14.html

[27] Lin, H.C., Wu, H.C., Chang, C.H., Li, T.C., Liang, W.M., Wang, J.Y.W.: Development of a real-time clinical decision support system upon the web mvc-based architecture for prostate cancer treatment. BMC Medical Informatics and Decision Making 11(1), 1–11 (2011), http://dx.doi.org/10.1186/1472-6947-11-16

[28] Liu, Q., Liu, G.: Research on the Framework of Decision Support System Based on ERP Systems. In: 2010 Second International Workshop on Education Technology and Computer Science (ETCS). vol. 1, pp. 704–707 (2010)

[29] Liu, S., Duffy, A.H.B., Whitfield, R.I., Boyle, I.M.: Integration of decision support systems to improve decision support performance. Knowledge and Information Systems 22(3), 261–286 (2010), http://dx.doi.org/10.1007/s10115-009-0192-4

[30] Loya, S.R., Kawamoto, K., Chatwin, C., Huser, V.: Service Oriented Architecture for Clinical Decision Support: A Systematic Review and Future Directions. J. Med. Syst. 38(12), 1–22 (2014), http://dx.doi.org/10.1007/s10916-014-0140-z

[31] Mohktar, M.S., Lin, K., Redmond, S.J., Basilakis, J., Lovell, N.H.: Design of a decision support system using open source software for a home telehealth application. In: Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2011 2nd International Conference on. pp. 298–303 (2011)

[32] Power, D.J., Kaparthi, S.: Building Web-based decision support systems. Studies in Informatics and Control 11(4), 291–302 (2002)

[33] Pullum, L.L., Symons, C.T., Patton, R.M., Beckerman, B.G.: Architecture-level Dependability Analysis of a Medical Decision Support System. In: Proceedings of the 2010 ICSE Workshop on Software Engineering in Health Care. pp. 83–88. SEHC '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1809085.1809096

[34] Rohloff, M.: An Integrated View on Business- and IT-architecture. In: Proceedings of the 2008 ACM Symposium on Applied Computing. pp. 561–565. SAC '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1363686.1363822

[35] Sen, A., Banerjee, A., Sinha, A.P., Bansal, M.: Clinical decision support: Converging toward an integrated architecture. Journal of biomedical informatics 45(5), 1009–1017 (2012)

[36] Solms, F.: What is Software Architecture? In: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference. pp. 363–373. SAICSIT '12, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2389836.2389879

[37] Struss, P.: A conceptualization and general architecture of intelligent decision support systems. In: 19th International Congress on Modelling and Simulation (MODSIM2011). pp. 2282–2288 (2011)

[38] Wettel, R., Lanza, M.: Visualizing Software Systems as Cities. In: 2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2007). pp. 92–99. IEEE (2007)

[39] Wettel, R., Lanza, M.: CodeCity. In: 1st International Workshop on Advanced Software Development Tools and Techniques (WASDeTT 2008). Citeseer (2008), http://scg.unibe.ch/download/wasdett/wasdett2008-paper14.pdf

[40] Zhang, H.T., Yang, Q., Lai, C.S., Lai, L.L.: New trends for Decision Support Systems. In: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 1373–1378 (2012)

A Anhang

In diesem Abschnitt befinden sich im Rahmen dieser Arbeit wichtige, zusätzliche Informationen.

A.1 Glossar (Begriffsdefinitionen)

Im folgenden Glossar werden für diese Arbeit wesentliche Begriffe und deren Definition aufgeführt.

ad-hoc Ohne Vorbereitung, speziell für einen Zweck oder spontan aus einer Situation heraus entstanden.
[Englisch: ad hoc]

Assembly Eine Zusammenstellung übersetzter Programmdateien.
[Englisch: assembly]

Deployment Prozess zur Installation von Software in eine Umgebung, welche verteilt aufgebaut sein kann.
[Synonyme: Softwareverteilung]
[Englisch: deployment]

Code-Behind Ein Verfahren zur Trennung von Darstellungsregeln und Anwendungslogik. Im Gegensatz zum so genannten Inline-Code, bei dem sich Layoutelemente und Befehle vermischen, werden hier Programmbefehle in gesonderte Dateien ausgelagert.
[Englisch: code behind]

Enterprise-Resource-Planning-System Ein Enterprise-Resource-Planning-System (ERP-System) unterstützt sämtliche in einem Unternehmen ablaufenden Geschäftsprozesse [25]. Es enthält Module für die Bereiche Beschaffung, Produktion, Vertrieb, Anlagenwirtschaft, Personalwesen, Finanz- und Rechnungswesen usw., die über eine gemeinsame Datenbasis miteinander verbunden sind [25].
[Englisch: enterprise resource planning]

Entscheidungsfindung Das Treffen einer Wahl für eine Menge von Handlungsalternativen [37]. Die Entscheidungsfindung umfasst das Setzen von Zielen, das Einholen relevanter Informationen, die Bestimmung von Alternativen, die Entwicklung von Entscheidungskriterien und die Wahl einer Option.
[Englisch: decision making]

Entscheidungsunterstützungssystem Ein interaktives computerbasiertes System, welches dafür ausgelegt ist, EntscheidungsträgerInnen bei der Identifikation und Lösung von Entscheidungsproblemen zu unterstützen und beim Treffen von Entscheidungen zu helfen [32]. Dies kann unter anderem unter Zuhilfenahme von Kommunikationstechnologien, Daten, Dokumenten, Wissen und Modellen geschehen [32].
Der Begriff umfasst dabei jegliche Art von Computeranwendung, welche die Fähigkeiten einer Person oder Gruppe bezüglich des Treffens von Entscheidungen erweitert.
[Englisch: Decision Support System (DSS)]

Entwurfsmuster Eine bewährte Lösungsschablone für wiederkehrende Entwurfsprobleme. In der Softwarearchitektur ein spezifisches Design bezogen auf die Architektur von Softwarekomponenten.
[Synonyme: Architekturansätze]
[Englisch: design pattern]

Framework Ein Framework stellt einen Rahmen zur Verfügung, innerhalb dessen eine Anwendung erstellt wird, wobei unter anderem durch die in dem Framework verwendeten Entwurfsmuster auch die Struktur der individuellen Anwendung beeinflusst wird.
[Synonyme: Programmiergerüst]
[Englisch: framework]

Informationsmanagement Die Zielsetzungen des Informationsmanagements konzentrieren sich auf die Unterstützung der Unternehmensziele, mit dem Schwerpunkt auf die Informatik und deren Leistungen, wie insbesondere die Bereitstellung der Informations- und Kommunikationssysteme.
[Englisch: information management]

Informationssystem Summe aller geregelten betriebsinternen und -externen Informationsverbindungen sowie deren technische und organisatorische Einrichtung zur Informationsgewinnung und -verarbeitung [26].
[Englisch: information system]

Intranet Ein Rechnernetz, welches unabhängig von öffentlichen Netzen benutzt werden kann und nicht öffentlich zugänglich ist.
[Synonyme: internes Netzwerk]
[Englisch: intranet]

Komponente Struktureller Bestandteil eines Systems, der mit anderen Bestandteilen zusammenwirken kann.
[Synonyme: Bestandteil]
[Englisch: component]

Modellierung Der Prozess der Formalisierung eines Problems. Es soll eine abstrakte Repräsentation eines tatsächlich existierenden Systems gefunden werden, welche dieses System so stark wie möglich vereinfacht [11]. Dies geschieht, indem unnötige Details weggelassen werden und wesentliche Beziehungen erhalten bleiben [11].
[Englisch: modeling]

Round-Trip–Netzwerkverzögerung Die Verzögerung welche bei einer Netzwerkverbindung auftritt, wenn ein Signal von der Quelle zum Ziel und zurück gesendet wird.
[Englisch: round-trip network delay]

Softwarearchitektur Eine Softwarearchitektur ist die Struktur einer Software, welche die Komponenten, welche die Funktionen der Applikation bereit stellen, einhalten.
[Englisch: software architecture]

Stakeholder Alle internen und externen Personengruppen, die vom Verlauf oder Ergebnis eines Prozesses oder Projektes gegenwärtig oder in Zukunft, direkt oder indirekt betroffen sind.
[Synonyme: Interessengruppen, Anspruchsgruppen, Teilhaber]
[Englisch: stakeholder]

Systemarchitektur Eine Systemarchitektur umfasst Komponenten und deren Beziehungen, welche zusammengenommen das betrachtete Gesamtsystem ergeben.
[Synonyme: Systemaufbau]
[Englisch: system achitecture]

Virtuelle Maschine Virtualisiertes System, welches in einer abgeschotteten virtuellen Umgebung läuft. Meist handelt es sich dabei um ein vollständiges Betriebssystem, welches sich exakt so verhält, als sei es direkt auf der Hardware installiert.
[Englisch: virtual machine]

Web-Technologien Technologien, welche die Internet-Kommunikation nutzen, um ihre Aufgabenstellungen zu erfüllen. Web Technologien sind die technischen Plattformen in der Webentwicklung und können als Werkzeuge der Webentwickler angesehen werden.
[Englisch: web technologies]

Workflow Ein Workflow ist eine definierte Abfolge von Aktivitäten in einem Arbeitssystem einer Organisation.
[Synonyme: Arbeitsablauf]
[Englisch: workflow]

A.2 Umsetzungen von Architekturen

Konkrete Umsetzungen von in der Literaturrecherche dieser Arbeit beschriebenen Architekturen sind an dieser Stelle angeführt.

A.2.1 Architektur von Web-Services nach Chen, Zhang, Chi

Bei der Umsetzung der Web-Services in [10] wird folgender Aufbau verwendet: Eine Service-Factory erzeugt anhand von Konfigurationsangaben Realisierungen von Diensten [10]. Diese setzten alle die Service-Schnittstellen-Definitionen (Service Interfaces Definitions) um [10]. Die Webanwendung hängt dabei nur von der Service-Fabrik und den Service-Schnittstellen-Definitionen ab [10]. Dieses Design entkoppelt die Webanwendungs-Schicht von den konkreten Realisierungen der Dienste [10]. Es wird unter anderem einfacher, die analytischen Werkzeuge des Entscheidungsunterstützungssystems zu erweitern [10].

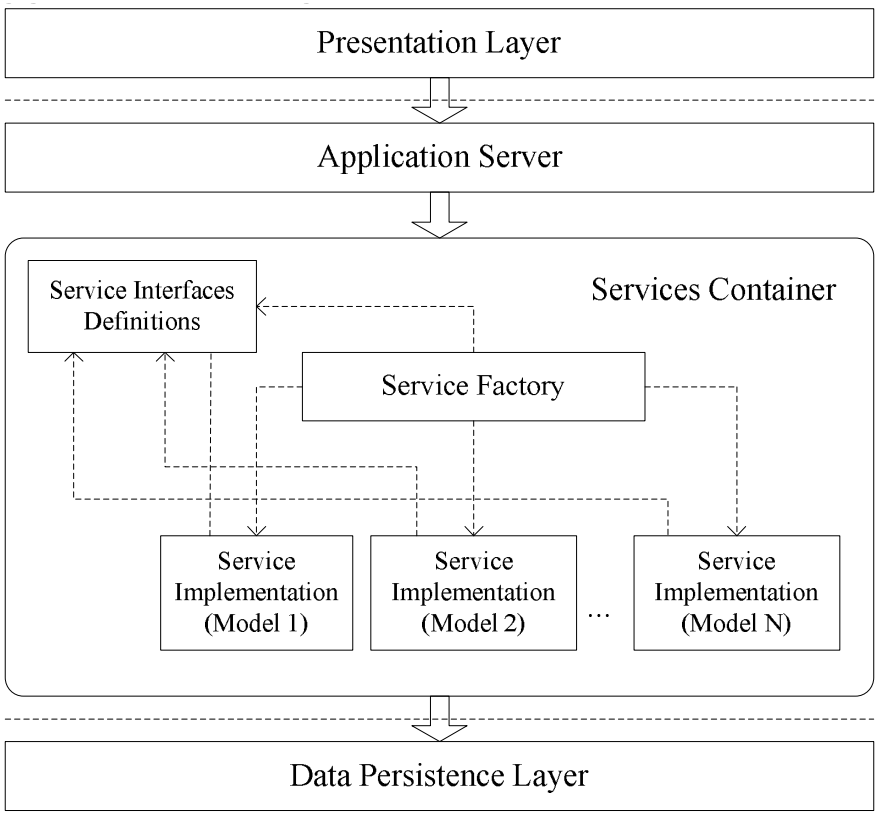


Abbildung A.1: Mögliche Verwendung von Web-Services in der Anwendungslogik-Schicht (Quelle: [10])

Die in Abbildung A.1 vorgeschlagene Architektur erlaubt es, mit allen Featuren der Anwendung zu interagieren ohne zu wissen, wie genau die verschiedenen Komponenten untereinander interagieren [10]. Unter die bekannten Vorteile dieser Architektur fallen hohe Flexibilität des Systems, einfache Wartbarkeit und einfaches Upgrade sowie reduzierte Zeit von Entwicklungszyklen durch Wiederverwendung existierender Komponenten [10].

Die Architektur besitzt jedoch auch Nachteile [10]. Erstens wird die Komplexität bei der Systementwicklung erhöht [10]. Zweitens ist eine vollständige Entkopplung jeder Schicht beinahe unmöglich [10]. Beispielsweise können einige Änderungen, welche in der Struktur oder bei den Formaten von Parametern stattfinden, die Notwendigkeit von Anpassungen, sowohl in der Anwendungslogik-Schicht als auch in der Präsentations-Schicht, mit sich bringen [10].

A.2.2 Technologien des Frameworks nach Mohktar et al.

Eines der betrachteten Systeme [31] ist unter Verwendung der JBoss-Seam-Technologie implementiert. Dies ist eine Software-Technologie, welche eine Auswahl an Codes und Diensten zu einer Umgebung zusammenfügt, welche es einfacher macht, Webanwendungen zu schreiben. Dieses javabasierte Framework lässt sich mit vielen anderen Produkten verbinden. Darunter fallen Hibernate (eine Java-Bibliothek für die Verbindung zu Datenbanken), JBoss Business Process Management (jBPM), ein grafisches Modellierungs-Werkzeug für das Prozessmanagement, sowie Drools (ein Produktionsregelsystem) und JavaServer Faces (JSF). Das Entscheidungsunterstützungssystem verwendet Hibernate als Lösung, um in der Datenzugriffs-Schicht die Daten in der Datenbank auf Datenhaltungsobjekte abzubilden. Diese Abbildung erlaubt es, die Datenhaltungsobjekte in der Anwendungslogik-Schicht und Frontend-Schicht einzusetzen. Die Benutzerschnittstelle des Frontends beruht auf der JSF-Technologie, um Daten, welche aus Eingaben in Bedienungselementen stammen, zu verarbeiten. Das Verhalten dieser Elemente, so wie Knöpfe oder Verweise, wird durch an diese Elemente gebundene Methoden bereit gestellt. Die jBPM Process Definition Language (jPDL) ist die Sprache, mit der das Prozessmodell erstellt wird. Für das Regelsystem in jedem Prozess wird Drools verwendet [31].

Die Vorteile in der Verwendung von Seam im Gegensatz zu anderen Web-Frameworks (so wie beispielsweise Spring oder Struts) sind folgende: Zum einen ist Seam ein Konversations-Framework was bedeutet, dass Informationen über Transaktionen hinweg erhalten bleiben. Dies begünstigt eine erfolgreiche Ausführung über die Interaktionen hinweg. Zum anderen besitzt Seam die Fähigkeit, auf die Daten aus dem Domänen-Modell innerhalb der Benutzerschnittstellen-Schicht zuzugreifen. Dies wiederum vermeidet den Zugang zu serverseitigen Funktionalitäten durch BenutzerInnen und verbessert damit die Sicherheit des Systems.

Das Framework ist unter ausschließlicher Verwendung von quelloffener Software entwickelt.

A.2.3 Technologien der Integration von Quelldaten nach Lin et al.

Zur Umsetzung einer Integration von Quelldaten kommen bei einem Entscheidungsunterstützungssystem [27] folgende drei Techniken der Datenbank-Software zum Einsatz. Dynamische Sichten (Views), gespeicherte Prozeduren und Auslöser (Trigger). Mit diesen Techniken werden automatisch Datentransaktionen ausgeführt, um die diversen Daten zu integrieren.

Damit wird ein in Echtzeit arbeitendes Entscheidungsunterstützungssystem verwirklicht, welches in der Lage ist, Daten aus vielzähligen Datenbanken zu integrieren. Dieses Entscheidungsunterstützungssystem erhält somit die Fähigkeit, sofortiges Feedback bereitzustellen.

A.3 Qualitätsanforderungen der Anwendung aus Dokumentation

Hier sind leicht überarbeitete Qualitätsanforderungen an den CIO-Navigator, welche aus einem entsprechenden Dokument entnommen sind, aufgelistet.

(A) Ein Rollenkonzept ist notwendig, um verschiedenen Funktionalitäten und Restriktionen für unterschiedliche Personengruppen zu ermöglichen.

(B) Der Navigator soll die Anzeige von reduzierten/ausgeblendeten Information unterstützen. Dadurch ist es möglich, die Benutzerschnittstelle als im Gang aufgehängtes Dashboard für die Allgemeinheit im Bereich IT (Bereich 1) oder aber für den Vorstand zu präsentieren. Dafür ist die Konfigurierbarkeit der Anzeige notwendig.

(C) Modularer Aufbau: Der Navigator sollte nicht nur für einzelne Personen nutzbar sein, sondern für den gesamten Bereich. Verschiedene Sichten für unterschiedliche Rollen, unter anderem als Info-Terminal auf dem Gang.

(D) Der Navigator soll um neue Ansichten für andere Nutzeraufgaben erweiterbar sein.

(E) Der Navigator soll als Erweiterung von Microsoft SharePoint realisiert werden.

(F) Der Navigator soll zur unabhängigen Nutzung möglichst webbasiert realisiert werden.

(G) Der Navigator muss insbesondere für veränderliche Datenquellen (in Bezug auf Ort bzw. Struktur) konfigurierbar sein.

(H) Der Navigator sollte so viele bestehende Systeme wie möglich integrieren (SAP, SharePoint, SCM).

(I) Heterogenität von verschiedenen Infrastrukturen muss unterstützt werden (Jedes Krankenhaus hat andere Prozesse, Tools und Schnittstellen. Die Prozessunterstützung wird für kaum realisierbar gehalten, der Einsatz des CIO-Navigators als Dashboard dagegen schon).

(J) Bei Mitarbeiterdaten muss auf den Datenschutz geachtet werden.

(K) Es gibt keine Anforderungen bezüglich des Datenschutzes von Patienten, da keine Patientendaten visualisiert werden.

(L) Es existieren keine besonderen Anforderungen an die Verfügbarkeit des Navigators.

(M) Folgende Aktualität der Daten wird erwartet:
Projektaktivitäten: Tagesaktuell.
SLA: Stundengenau.
Service Desk: Ad hoc.

Selbstständigkeitserklärung

Hiermit versichere ich, **Tim Bittersohl**, dass ich die Master-Thesis mit dem Titel **"Analyse und Optimierung der Softwarearchitektur für einen CIO-Navigator als Entscheidungsunterstützungssystem für das Informationsmanagement eines Krankenhauses"** bei **Prof. Dr. Barbara Paech** als Erstbetreuerin und **Christian Kücherer** als Zweitbetreuer selbstständig und nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Zitate sowie der Gebrauch fremder Quellen, Texte und Hilfsmittel habe ich nach den Regeln wissenschaftlicher Praxis eindeutig gekennzeichnet.

Heidelberg, den 14. Dezember 2016 _____