POSAAM
Eine Methode zu mehr Systematik und Expertenunabhängigkeit in der qualitativen Architekturbewertung

David Bettencourt da Cruz

Institut für Informatik
der Technischen Universität München

POSAAM
Eine Methode zu mehr Systematik und Expertenunabhängigkeit in der qualitativen Architekturbewertung

David Bettencourt da Cruz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:	Univ.-Prof. Dr. Anne Brüggemann-Klein
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. Manfred Broy
2.	Univ.-Prof. Dr. Ralf Reussner Universität Karlsruhe (TH)

Die Dissertation wurde am 09. April 2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13. Mai 2009 angenommen.

Kurzbeschreibung

Die Bewertung von Softwarearchitekturen ist eine Form der analytischen Qualitätssicherung. Es ist bekannt, dass die Behebung von Fehlentwicklungen frühzeitig im Softwareentwicklungsprozess (z.B. in der Entwurfsphase) mit weniger Kosten verbunden ist, als die Behebung von Fehlentwicklungen zu einem späten Zeitpunkt im Softwareentwicklungsprozess (z.B. nach der Auslieferung des Systems). Ziel der Bewertung von Softwarearchitekturen ist die Erkennung von Fehlentwicklungen frühzeitig im Entwicklungsprozess, um diese Fehlentwicklungen frühzeitig beheben zu können und somit Kosten zu sparen.

Es existieren sowohl quantitative als auch qualitative Architekturbewertungsmethoden. Bei quantitativen Methoden werden durch die Kombination von initialen Schätzungen oder Erfahrungswerten mithilfe von Berechnungsvorschriften zu erwartende Ausprägungen von Qualitätsmerkmalen des zu erstellenden Systems ermittelt. Bei den qualitativen Methoden wird vorliegendes Wissen über die geeignete Gestaltung von Architekturen genutzt, um potenzielle Missstände in zu prüfenden Architekturen zu identifizieren.

In der vorliegenden Arbeit wird die qualitative Architketurbewertungsmethode POSAAM (**P**attern **O**rieneted **S**oftware **A**rchitecture **A**nalysis **M**ethod) entwickelt. Zu diesem Zweck wird in der Methode auf das in Architekturmustern gekapselte Expertenwissen zugegriffen. Architekturmuster beinhalten Wissen über geeignete Gestaltungen von Architekturen zur Lösung von Problemen einer Problemklasse. Die Methode beschreibt, wie das in Architekturmustern gekapselte Expertenwissen geeignet hinterlegt wird, um die systematische Nutzung des Expertenwissens während der Bewertung zu ermöglichen.

Durch die Nutzung von Mustern zum Zwecke der Architekturbewertung werden die Objektivität, Systematik, Nachvollziehbar- und Wiederholbarkeit im Vergleich zu existierenden Methoden gesteigert.

Danksagung

Obwohl Dissertationen selbstständig erarbeitet und verfasst werden, ist es kaum möglich eine solche Arbeit ohne die Diskussion, Interaktion und der Unterstützung von vielen Menschen zu gestalten. Diesen Menschen möchte ich meinen Dank aussprechen.

So möchte mich bei meinem Betreuer, Prof. Broy, bedanken. Er hat es verstanden, mich in den schwierigeren Zeiten, die manchmal bei der Anfertigung einer solchen Arbeit zu durchlaufen sind, zu unterstützen und die richtigen Impulse zu geben. Auch bei meinem Zweitgutachter, Prof. Reussner, möchte ich mich bedanken. Die äußerst unkomplizierte Form der Zusammenarbeit hat mir sehr gefallen. Beiden Professoren Danke ich für die wertvollen Diskussionen und Feedback zu meiner Arbeit.

Für die Anfertigung einer Dissertation benötigt man neben vielen anderen Dingen viel Zeit. Ich möchte mich bei meinen Kollegen am Lehrstuhl dafür bedanken, dass sie mir diese Zeit verschafft haben, indem sie mich bei der Projektarbeit wo nur möglich entlastet haben. Insbesondere im Rahmen der Arbeiten zum V-Modell XT haben mir Gernot Stenz und Marco Kuhrmann viel Arbeit abgenommen.

Die Hilfsbereitschaft all meiner Kollegen am Lehrstuhl ist bemerkenswert. Nicht nur bei der Bewältigung von Projektarbeit, sondern auch bei beliebigen anderen Situationen, von der selbstverständlichen Unterstützung bei technischen Themen (z.B. bei der manchmal recht mühseligen Arbeit mit Softwarewerkzeugen), über aufmunternde Worte, bis hin zu wertvollen Tipps im Umgang mit schwierigen Industriepartnern oder Studenten. Jeder hat immer ein offenes Ohr und bemüht sich immer zu helfen, wo er nur kann. Danke dafür.

Den Einblick in die Fallstudie „Batch-Frame“ haben mir Moritz Theile, Klaus Bergner und Marc Sihling ermöglicht. Dafür bedanke ich mich.

Bei Birgit Penzenstadler, Marco Kuhrmann und Markus Pister möchte ich mich dafür bedanken, dass sie sich die Zeit genommen haben, meine Arbeit zu lesen und meine Ideen kritisch zu hinterfragen und mir so in sehr fruchtbaren Diskussionen geholfen haben, meine Arbeit und meine Ideen weiter zu formen und zu festigen.

Zuletzt möchte ich mich bei meiner Familie bedanken. Bei meinen Eltern für ihre Unterstützung und Erziehung und bei meinem Bruder für seine Vorbildfunktion von Schulzeit an. Wahrscheinlich hat er mich mehr geprägt als ihm selbst bewusst ist.

München, im April 2009

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Definitionenverzeichnis

KAPITEL 1

Einleitung und Überblick

Die kontinuierliche analytische Qualitätssicherung ist ein nicht mehr wegzudenkender Teil der Softwareentwicklung. Zweck der analytischen Qualitätssicherung ist das Aufdecken von Mängeln, um entsprechende Gegenmaßnahmen einleiten zu können. Die Architekturbewertung ist ein Mittel der analytischen Qualitätssicherung, durch welches Mängel der Qualität der Architektur von Softwaresystemen frühzeitig entdeckt und behoben werden können.

In dieser Einleitung wird der Zusammenhang von Softwarearchitektur und Softwarequalität dargelegt und der dadurch entstehende Bedarf an Architekturbewertung motiviert. Im Anschluss wird der Beitrag der Arbeit herausgestellt und zu verwandten Arbeiten in Beziehung gesetzt. Schließlich wird ein Überblick über den Aufbau der Arbeit gegeben.

Inhalt

1.1. Softwarequalität und Softwarearchitektur

Zum Ende der sechziger, Anfang der siebziger Jahre entstanden erstmals Überlegungen, wie Software geeignet strukturiert werden kann. In den bekannten Arbeiten von *Parnas* „On the Criteria to be Used in Decomposing Systems into Modules“ [Par72] und *Stevens et al.* „Structured Design“ [SMC74] werden Vorschläge zur Strukturierung von Software gemacht, durch die das Qualitätsmerkmal Wartbarkeit positiv beeinflusst werden kann. Bereits in diesen Arbeiten wird auch der Zusammenhang der Strukturierung zu anderen Qualitätsmerkmalen (Wiederverwendbarkeit und Performanz) festgestellt. Inzwischen ist allgemein anerkannt, dass die Strukturierung und somit die Architektur einer Software einen Einfluss auf eine Vielzahl von Qualitätsmerkmalen haben kann [BCK03]. Allerdings kann die Softwarearchitektur alleine nie die Qualität eines Systems abschließend festlegen. Dies liegt daran, dass die Qualität eines Systems auch immer von dessen Implementierung abhängt. Werden beispielsweise in einem System oft mehrere tausend Elemente sortiert und dazu in der Implementierung Algorithmen mit quadratischer oder noch schlechterer Komplexität verwendet, wird das System auch dann kein gutes Zeitverhalten vorweisen, wenn die Architektur des Systems auf gutes Zeitverhalten ausgelegt ist.

Analog verhält es sich auch, wenn die Implementierung auf ein Qualitätsmerkmal ausgelegt ist, die Softwarearchitektur jedoch nicht. Ist die Softwarearchitektur des Systems nicht auf gutes Zeitverhalten ausgelegt, so ist es auch durch geschickte Implementierungen der einzelnen Elemente des Systems nicht mehr möglich, das Zeitverhalten des Systems maßgeblich zu beeinflussen. Ist beispielsweise die Architektur eines Systems so ausgelegt, dass eine knappe Ressource oft ungenutzt oder blockiert ist, so wird auch eine effiziente Implementierung einzelner Teile des System nur minimalen Einfluss auf das Zeitverhalten haben können. Ähnlich verhält es sich mit anderen Qualitätsmerkmalen.

Bei der Entwicklung von Software wird das Wissen über den Zusammenhang von Softwarearchitektur und Softwarequalität genutzt, um die Qualität der zu entwickelnden Software zu steuern. Da nach oder während der Implementierung notwendig werdende Änderungen an der Architektur mit hohen Kosten verbunden sind, werden Softwarearchitekturen spezifiziert, die die geforderten Qualitätsmerkmale in gewünschter Weise beeinflussen, bevor mit der Implementierung begonnen wird. Genauer: Es werden Softwarearchitekturen spezifiziert, von denen geglaubt wird, dass sie das Erreichen einer gewünschten Ausprägung eines Qualitätsmerkmals nach fertiger Implementierung ermöglichen bzw. erleichtern.

Um sicher zu stellen, dass die spezifizierte Softwarearchitektur auch tatsächlich die Qualitätsmerkmale in gewünschter Weise ermöglicht, ist eine analytische Qualitätssicherung der Architekturspezifikation sinnvoll. Die analytische Qua-

litätssicherung einer Architekturspezifikation oder -dokumentation bezeichnet man auch als Architekturbewertung oder Architekturevaluation[1]. Der Zweck einer Architekturbewertung wird im folgenden Abschnitt ausführlicher motiviert.

1.2. Architekturbewertung

Wie im vorangegangenen Abschnitt motiviert, besteht ein Zusammenhang zwischen der Architektur eines Systems und dessen Qualitätseigenschaften. Die Architekturspezifikation ist eines der ersten Artefakte, die das zu entwickelnde System beschreiben und somit eines der ersten Artefakte, die eine Prüfung der Qualitätseigenschaften des zu entwickelnden Systems ermöglichen. Durch die frühzeitige Prüfung können frühzeitig Missstände aufgedeckt und behoben werden. Dadurch können Kosten gespart werden. Der Aspekt des Nutzens der Architekturbewertung wird im Abschnitt 1.2.1 beleuchtet. Im Abschnitt 1.2.2 wird auf die unterschiedlichen Formen der Architekturbewertung eingegangen.

1.2.1. Nutzen der Architekturbewertung

Eine Architekturbewertung während des Entwicklungsprozesses durchzuführen, kostet Zeit und Personalaufwand und somit Geld. Wie bei jeder Investition ist es deshalb notwendig, die entstehenden Kosten dem zu erwartendem Nutzen gegenüber zu stellen. Bezüglich der Kosten einer Architekturevaluation berichten *Abowd et al.* in [ABC+97, S. 5] von Aufwänden zwischen 14 (SEI) und 70 (AT&T) Personentagen für Softwareprojekte mit einem Umfang von 5-100 KLOC. Dabei ist anzumerken, dass die Aufwände abhängig von der angewendeten Methode, von der Erfahrung der beteiligen Stakeholder bei der Teilnahme an Evaluationen und von der Erfahrung des Evaluationsteams bei der Durchführung von Evaluationen sind. *Clements*, *Kazman* und *Klein* nennen Abschätzungen zwischen 32 und 70 Personentagen für die ATAM-Evaluationen von Projekten [CKK02, S.39 ff].

Der Nutzen, den eine Architekturbewertung bringt, wird von [ABC+97, S. 6] in fünf Kategorien unterteilt:

- Finanzieller Nutzen,
- Nutzen durch Steigerung des Verständnis und der Dokumentation des Systems,

[1] Die beiden Begriffe Architekturbewertung und -evaluation werden im weiteren Verlauf der Arbeit synonym verwendet.

- Nutzen durch Erfassung von Missständen des Systems,
- Nutzen durch Verdeutlichung und Priorisierung von Anforderungen und
- Nutzen durch eine verbesserte Lernkurve der Organisation.

Finanzieller Nutzen. Es gibt keine wissenschaftlich fundierten Studien mit exakten Zahlen zu den finanziellen Vorteilen einer Architekturevaluation. *Abowd et al.* beziehen sich in ihrer Arbeit auf Erfahrungswerte von Unternehmen, die Architekturevaluationen durchgeführt haben. Die Schätzungen besagen, dass nach der Durchführung einer Architekturevaluation die Projektkosten um etwa 10% reduziert wurden. Weitere finanzielle Vorteile begründen *Abowd et al.* mit Erfahrungen von an einer Architekturevaluation Beteiligten, die aussagen, dass bei der Architekturevaluation Fehler entdeckt wurden, die zu erheblichen Kosten geführt hätten, wären diese nicht entdeckt worden. Schließlich nennen *Abowd et al.* Projekte, bei denen keine Architekturevaluation durchgeführt wurde, bei welchen die Mängel in der Architektur zum Scheitern der Projekte geführt haben.

Verbessertes Verständnis und Dokumentation. In Softwareentwicklungsprojekten ist die Architekturdokumentation oft zu knapp (oder nicht vorhanden) oder zu umfangreich. Im Rahmen der Architekturevaluation sind Architekten und Entwickler damit konfrontiert sich mit der Architekturdokumentation zu befassen bzw. diese für die Nutzung während einer Architekturevaluation vorzubereiten. Dadurch entstehen sowohl eine verbesserte Architekturdokumentation als auch ein besseres Verständnis der Architektur des Systems bei den Entwicklern.

Erfassung von Missständen. Der dritte Nutzen, den *Abowd et al.* identifizieren, ist die frühe Entdeckung von Missständen. Dies betrifft sowohl Missstände der Architektur als auch Missstände in den spezifizierten Anforderungen. Durch die frühe Entdeckung von Missständen, können Gegenmaßnahmen entsprechend früh im Entwicklungsprozess eingeleitet werden. Die frühe Einleitung von Gegenmaßnahmen ist verglichen mit einer späteren Korrektur der Implementierung kostengünstiger. Der Vorteil der frühzeitigen Erfassung von Missständen trägt damit maßgeblich zu den finanziellen Vorteilen bei.

Verdeutlichung und Priorisierung von Anforderungen. Ein weiterer Nutzen, den *Abowd et al.* in der Durchführung von Architekturevaluationen sehen, ist eine Qualitätssicherung der gestellten Anforderungen. Es ist möglich, Anforderungen zu definieren, die im Widerspruch zueinander stehen, d.h., dass sie nicht

gleichzeitig erfüllt werden können. Eine Architekturevaluation kann diesen Widerspruch verdeutlichen, da die Durchführung der Architekturevaluation zum Ergebnis haben kann, dass es keine Architektur gibt, die allen gestellten Anforderungen gerecht wird.

Verbesserte Lernkurve. Wird eine Architekturevaluation in den Entwicklungsprozess einer Organisation aufgenommen, werden sich die Architekten bereits auf eine Architekturevaluation vorbereiten und die Architektur anders entwerfen und dokumentieren, als wenn sie nicht mit einer Architekturevaluation rechnen würden. Sie werden in ihrem Entwurf der Architektur systematischer und nachvollziehbarer vorgehen, um zu erwartende Fragen bei der Evaluation beantworten zu können. Dadurch wird die Lernkurve beim Entwurf und der Bewertung von Softwarearchitekturen in der Organisation verbessert.

Die genannten Vorteile verdeutlichen, dass eine Architekturevaluation einen Mehrwert für Softwareentwicklungsprojekte liefern kann.

1.2.2. Verfahren der Architekturbewertung

Für die analytische Qualitätssicherung der Architektur gibt es sowohl *quantitative* als auch *qualitative* Verfahren.

Quantitative Verfahren zur Architekturbewertung

Bei quantitativen Verfahren werden die spezifizierten Architekturrepräsentationen mit Annotationen versehen, die Annahmen, die durch Messungen oder Schätzungen entstanden sind, repräsentieren. Mit Hilfe dieser Annahmen werden Folgerungen über Qualitätsmerkmale gezogen. Quantitative Verfahren basieren üblicherweise auf formalen Regeln zur Berechnung oder Herleitung eines quantitativen Ergebnisses. Aufgrund der Nutzung formaler Darstellungen und Regeln sind quantitative Verfahren in ihrer Natur systematischer und dadurch auch nachvollziehbarer und wiederholbarer als die qualitativen Verfahren. Es existieren quantitative Verfahren, die aufgrund der Nutzung formaler Darstellungen und Regeln automatisierbar oder in Teilen automatisierbar sind. Die quantitativen Verfahren haben drei Schwachstellen:

1. Quantitative Verfahren sind üblicherweise auf ein Qualitätsmerkmal ausgerichtet und betrachten nicht die Abhängigkeiten zwischen Qualitätsmerkmalen. Ein Beispiel ist Software Performance Engineering [Smi90].

2. Es gibt Qualitätsmerkmale, für die die Aussagekraft quantitativer Verfahren bezüglich der tatsächlichen Ausprägung des durch die Verfahren analysierten Qualitätsmerkmals, nicht ausreichend ist (vgl. Ausführungen in [BR08]).
3. Die Durchführung quantitativer Verfahren gibt keine Hinweise auf mögliche Alternativen. Quantitative Verfahren haben zum Ziel, für eine bestehende Architektur die Ausprägung eines Qualitätsmerkmals quantitativ abzuschätzen. Zwar können mit quantitativen Verfahren zwei bestehende Alternativen verglichen werden, Hinweise auf nicht bestehende Alternativen werden im Laufe des Verfahrens nicht gegeben.

Es gibt qualitative Verfahren, bei welchen diese Schwachstellen nicht auftreten.

Qualitative Verfahren zur Architekturbewertung

Bei den *qualitativen* Verfahren kommen Erfahrungswerte zum Einsatz. So werden beispielsweise Fragebögen oder Checklisten, die aus der Erfahrung von Architekten oder aus der gesammelten Erfahrung des Unternehmens entstehen, eingesetzt. Auch Expertenreviews gehören zu den qualitativen Verfahren.

Die Schwachpunkte der quantitativen Verfahren werden durch qualitative Verfahren adressiert. Durch den Einsatz von Experten zur Bewertung verliert die Bewertung jedoch an Systematik und dadurch auch an Wiederhol- und Nachvollziehbarkeit. Zudem sind qualitative Verfahren nicht automatisierbar.

Bei szenariobasierten Verfahren werden Nutzungsszenarien, in denen wünschenswerte Qualitätsmerkmale gekapselt werden, für die Architekturbewertung genutzt. Szenarien werden in einigen qualitativen Verfahren zur Architekturbewertung verwendet, um die Anforderungen an die zu bewertenden Architektur zu formulieren. Die bekanntesten qualitativen szenariobasierten Verfahren sind ATAM [KKB+98, KKC00, CKK02] und SAAM [KBWA94, KABC96, CKK02]. Von diesen Verfahren existieren diverse Erweiterungen und Anpassungen. Es handelt sich bei diesen Verfahren um methodisch geleitete Reviewverfahren, die außer des Schwerpunkts der Qualitätssicherung von Softwarearchitekturen zwei weitere Schwerpunkte haben:

1. Die Erhebung und Priorisierung von Anforderungen mit aktiver Teilnahme der Stakeholder.
2. Die Gestaltung von Social Engineering, um die Interaktion der an der Evaluation beteiligten Stakeholder so reibungslos wie möglich zu gestalten.

Den Schwachpunkten der mangelnden Systematik, Nachvollzieh- und Wiederholbarkeit von qualitativen Verfahren zur Architekturbewertung wird in dieser

Arbeit entgegen getreten. Der Schwerpunkt der Arbeit wird von der Erhebung von Anforderungen und dem Social Engineering weg und hin zu einer methodischeren Architekturbewertung verlegt. Der Beitrag der Arbeit wird im nächsten Abschnitt beleuchtet.

1.3. Beitrag der Arbeit

Der Beitrag der Arbeit liegt in der Entwicklung eines qualitativen Verfahrens zur analytischen Qualitätssicherung von Softwarearchitekturen, bei welchem der Schwerpunkt auf die systematische, nachvollziehbare und wiederholbare Durchführung der Bewertung gelegt wird. Zusätzlich wird im Verfahren die Abhängigkeit von Experten im Vergleich zu vergleichbaren qualitativen Architekturbewertungsmethoden reduziert.

Das in dieser Arbeit entwickelte Verfahren wird als POSAAM (**P**attern **O**riented **S**oftware **A**rchitecture **A**nalysis **M**ethod) bezeichnet, da es das in Mustern (engl. *Pattern*) gekapselte Expertenwissen für die Architekturbewertung verwendet. POSAAM wurde in Teilen bereits in [dCP08] vorgestellt.

Um die Systematik, Nachvollziehbarkeit und Wiederholbarkeit zu erreichen, werden in POSAAM folgende Neuerungen erarbeitet:

1. Der Begriff des Musters wird neu definiert. Diese Definition expliziert die Unterteilung eines Musters in wiederkehrende und variierende Anteile. Diese Differenzierung wird an verschiedenen Stellen von POSAAM genutzt.

2. Zu Mustern gibt es Informationen, die in den üblichen Musterbeschreibungen implizit (d.h. in der textuellen Beschreibung eingegliedert) hinterlegt sind, aber nicht explizit in den Beschreibungen hervorgehoben werden. Zu diesen Informationen gehören z.B. die Einflüsse von Mustern auf Qualitätsmerkmale oder die Beziehungen zwischen Mustern. Diese Informationen werden identifiziert und in einer Beschreibung hinterlegt, die für die Verwendung im entwickelten Evaluationsverfahren geeignet ist.

3. Zusätzlich zu den Informationen, die implizit in den üblichen Musterbeschreibungen hinterlegt sind, gibt es zu Mustern Informationen, die in üblichen Musterbeschreibungen nicht hinterlegt sind, für eine Evaluation nach POSAAM jedoch von Nutzen sein können. Diese Informationen werden identifiziert und in geeigneten Musterbeschreibungen hinterlegt.

4. Es wird ein Konzept erarbeitet, welches für die Evaluation verwendet wird, wenn keine Muster zur Evaluation herangezogen werden können. Dieses

Konzept basiert auf der Nutzung von Prinzipien, mit welchen die Qualitätsmerkmale durch Gestaltung der Architektur eines Systems beeinflusst werden können.

5. Die erarbeiteten Strukturen (Begriffe für Elemente, die für eine Bewertung nach POSAAM herangezogen werden, sowie deren Beziehungen untereinander) werden in einer Ontologie dargelegt.

6. Es wird gezeigt, wie die neu repräsentierten Informationen von Mustern im Evaluationsverfahren POSAAM verwendet werden und wie bei der Evaluation vorzugehen ist, wenn entsprechende Informationen nicht vorliegen.

Eine Besonderheit von POSAAM ist, dass bei POSAAM im Gegensatz zu anderen qualitativen Architekturbewertungsmethoden von der Erhebung von Anforderungen im Rahmen der Durchführung des Bewertungsprozesses abgesehen wird. Anforderungen müssen in POSAAM in definierter Form vorliegen. Analog verhält es sich bei der Architekturbeschreibung bzw. Architekturspezifikation des zu evaluierenden Systems. Auch diese muss in definierter Form vorliegen. Dadurch wird der Schwerpunkt von POSAAM weg von der Anforderungserhebung und der nachträglichen Dokumentation bereits vorliegender Systeme, hin zu der Herstellung eines Zusammenhangs zwischen Anforderungen und der vorliegenden Architektur verschoben. Zu diesem Zweck beginnt POSAAM eingangs mit einer Prüfung der gelieferten Eingaben, d.h. sowohl einer Prüfung der vorliegenden Architekturbeschreibung als auch einer Prüfung der vorliegenden Anforderungen.

Die Arbeit validiert das vorgeschlagene Verfahren (POSAAM) anhand einer Fallstudie. Eine vollständige empirische Validierung von POSAAM ist im Rahmen dieser Arbeit nicht möglich, da zum einen für den effizienten Einsatz von POSAAM eine Werkzeugunterstützung von Nöten ist und zum anderen die Wissensbasis auf der POSAAM aufbaut erst über mehrere Iterationen von Architekturevaluationen aufgebaut wird. Durch die Fallstudie wird jedoch die Anwendbarkeit der zentralen Konzepte von POSAAM demonstriert.

1.4. Verwandte Arbeiten

In [Mal08] hat *Malich* parallel zur Entstehung der vorliegenden Arbeit auch eine Form der Strukturierung von in Mustern gekapselten Informationen entwickelt, die den besseren Entwurf sowie die bessere Evaluation von Softwarearchitekturen unterstützen soll (vgl. Punkt 2 der Beiträge dieser Arbeit). *Malich* nennt seine Strukturierung ein Wissensmodell. Analog zur Arbeit von *Malich* werden in POSAAM die Muster zu Qualitätsmerkmalen und -teilmerkmalen in Beziehung

gesetzt. Im Unterschied zur Arbeit von *Malich* werden in der vorliegenden Arbeit die Muster mit Hilfe von wiederkehrenden und variierenden Anteilen beschrieben (Punkt 1 der Beiträge dieser Arbeit). Dadurch ist eine präzisere Darstellung des Zusammenhangs zwischen Mustern und Qualitätsmerkmalen möglich und somit wiederum eine besser geleitete Evaluation. Zusätzlich werden in der vorliegenden Arbeit auch Mechanismen entwickelt, die bei einer Evaluation verwendet werden können, wenn die Evaluation auf der Basis von Mustern nicht erfolgreich verläuft. Diese Mechanismen basieren auf der Verwendung von Prinzipien (Punkt 4 der Beiträge dieser Arbeit).

In [EA08] schlagen *Erfanian und Aliee* eine Ontologie für die Unterstützung der Bewertung von Softwarearchitekturen vor und zeigen kurz auf, wie diese in einer Architekturbewertung wie z.B. ATAM verwendet werden kann. Die in der vorliegenden Arbeit präsentierte Ontologie unterscheidet sich in einigen Punkten von der Ontologie von *Erfanian und Aliee*. Insbesondere wird die Unterteilung von Mustern in wiederkehrende und variierende Anteile und die Aufnahme von Prinzipien (Punkte 1 und 4 der Beiträge dieser Arbeit) in der Ontologie von *Erfanian und Aliee* nicht berücksichtigt.

Ali Babar und *Zhu et al.* präsentieren in [Bab04] und [ZBJ04] Vorschläge, wie welche architekturrelevanten Information aus Mustern extrahiert werden können (Punkt 2 der Beiträge der vorliegenden Arbeit). In der vorliegenden Arbeit werden zusätzlich zu den in den Arbeiten vorgeschlagenen Informationen noch weitere Informationen aus Mustern und Mustersammlungen extrahiert.

Mit den Attribute Based Architectural Styles (ABAS) [KKB+99, KK99] haben *Klein et al.* einen Vorschlag gemacht, wie der Einfluss der Gestaltung von Mustern auf Qualitätsmerkmale in ein „Argumentationsrahmenwerk“ (engl. *reasoning framework*) eingebettet werden kann. Dadurch stellen sie einen Zusammenhang zwischen Mustern und Qualitätsmerkmalen und deren Ausprägungen in Mustern her (Punkt 2 der Beiträge dieser Arbeit). In der vorliegenden Arbeit wird der Zusammenhang zwischen Mustern und Qualitätsmerkmalen hergestellt, indem die Einflüsse der wiederkehrenden und variierenden Anteile der Muster auf Qualitätsmerkmale betrachtet werden.

1.5. Aufbau der Arbeit

Im Kapitel 2 „Grundlagen und Definitionen“ werden die zentralen Themen der Arbeit, Architektur bzw. System- und Softwarearchitektur, Qualität und Softwarequalität sowie Muster, eingeführt und Begriffe dieser Themen definiert. Es wird aufgezeigt, wie System- und Softwarearchitekturen repräsentiert bzw. beschrieben werden können. Einige bekannte Qualitätseigenschaften von Softwaresystemen werden vorgestellt und es wird gezeigt, wie durch geeignete Maßnahmen der

Strukturierung des Systems Einfluss auf die eingeführten Qualitätseigenschaften genommen werden kann. Des Weiteren werden Muster bzw. Architekturmuster beschrieben. Es werden Möglichkeiten der Klassifikation von und Beziehungsstrukturen zwischen Mustern aufgezeigt.

Ein Überblick über den Stand der Technik zur qualitativen Architekturbewertung wird in Kapitel 3 gegeben. Neben den bekanntesten qualitativen Architekturbewertungsmethoden SAAM und ATAM werden auch weniger bekannte Methoden kurz vorgestellt. Die vorgestellten Methoden werden bezüglich diverser Kriterien in einer Taxonomie eingeordnet und miteinander verglichen. Es folgt eine Stellungnahme bezüglich der in ATAM vorhandenen methodischen Anleitung zur Architekturevaluation. Abschließend werden weitere verwandte Arbeiten etwas ausführlicher beschrieben. Zu diesen Arbeiten gehören das pattern-basierte Wissensmodell von *Malich* sowie die Ontologie zur Architekturbewertung von *Erfanian und Aliee*.

Die in dieser Arbeit erarbeitete Methode, POSAAM, wird in Kapitel 4 dargelegt. Es werden die durchzuführenden Aktivitäten erläutert. Es wird dargestellt, welche Produkte von den Aktivitäten vorausgesetzt und welche Produkte von den Aktivitäten erzeugt bzw. verändert werden. Die Produktstrukturen und -gestaltungen werden dargelegt. Schließlich wird POSAAM in die existierenden qualitativen Architekturbewertungsmethoden eingeordnet.

In Kapitel 5 wird die Anwendung zentraler Konzepte von POSAAM anhand einer Fallstudie demonstriert.

Schließlich werden die erarbeiteten Ergebnisse in Kapitel 6 rückblickend zusammengefasst. Zudem wird gezeigt, an welchen Stellen weitere Arbeiten anknüpfen können.

KAPITEL 2

Grundlagen und Definitionen

Vor der Beschreibung der in diesem Dokument erarbeiteten Methode zur Architekturbewertung, werden in diesem Kapitel Grundlagen und Definitionen erläutert. Es werden Grundlagen der Qualität von Software, der Softwarearchitektur und ihrer Beschreibungsmöglichkeiten und der Architekturmuster gelegt. Termini mit besonderem Stellenwert für die vorliegende Arbeit werden definiert.

Inhalt

2.1. Softwarequalität

Qualität ist keine per se messbare Größe. Es existieren unterschiedliche Ansätze um Qualität zu definieren und zu erfassen [Gar84]. Für das vorliegende Dokument werden die folgenden Definitionen gegeben:

> **Definition 1 (Qualität)**
> *Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Produkts oder einer Dienstleistung, die sich auf deren Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen [DIN95].*

Diese Definition lässt sich analog auf Software-Qualität übertragen.

> **Definition 2 (Softwarequalität)**
> *Softwarequalität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte oder vorausgesetzte Erfordernisse zu erfüllen [Joi01][1].*

Um Software zu entwickeln, die von hoher Qualität ist, muss demnach bekannt sein, welche Erfordernisse erwartet (vorausgesetzt) werden. In der Praxis der Softwareentwicklung wird deshalb vor Beginn der Entwicklungsarbeiten festgestellt, welche Erfordernisse erwartet werden. Dies geschieht in der Disziplin des Requirements Engineering (siehe z.B. [RR06]). Die festgestellten Erfordernisse werden anschließend für die Produktion bzw. Entwicklung festgelegt, üblicherweise als Qualitätsanforderungen. Zusätzlich muss während des Entwicklungsprozesses sicher gestellt werden, dass die Qualitätsanforderungen auch erfüllt werden.

Nachdem die Erfordernisse in Form von Qualitätsanforderungen festgelegt wurden, müssen diese im Entwicklungsprozess des Software-Produkts berücksichtigt werden. Die dazu existierenden Möglichkeiten werden in *konstruktive* und *analytische* Qualitätssicherungsmechanismen klassifiziert.

Als konstruktive Qualitätssicherungsmechanismen werden all jene Mechanismen verstanden, die den Prozess der Entwicklung an sich beeinflussen, um die Qualität des resultierenden Produkts zu verbessern.

Qualitätssicherungsmechanismen, bei denen Artefakte des Entwicklungsprozesses auf ihre bestehende Qualität hin analysiert werden, gelten als analytische Qualitätssicherungsmechanismen.

[1] Deutsche Übersetzung in Anlehnung an [Bal98]

Konstruktive Qualitätssicherungsmechanismen werden demnach verwendet, um zu definieren, wie vorzugehen ist, um Artefakte zu produzieren, die der festgelegten Qualität entsprechen und analytische Qualitätssicherungsmechanismen werden verwendet, um festzustellen, ob die produzierten Artefakte der festgelegten Qualität entsprechen.

Bei einigen analytischen Qualitätssicherungsmechanismen werden gleichzeitig auch die Missstände, die zu einer Minderung der Qualität führen, festgestellt (z.B. Review). Bei Anderen wird lediglich ein Maß an Qualität festgestellt (z.B. Performanz-Analyse), ohne evtl. vorhandene Missstände festzustellen.

Um die Qualität eines Software-Produkts messen und beurteilen zu können werden Qualitätsmodelle definiert. In diesen werden (messbare/beurteilbare) Eigenschaften eines Software-Produkts zur Qualität des Software-Produkts in Beziehung gesetzt.

Definition 3 (Qualitätsmodell)
Ein Qualitätsmodell spezifiziert Qualität durch Verfeinerung in Qualitätsmerkmale und -teilmerkmale, denen dann Qualitätsindikatoren zugeordnet werden (in Anlehnung an [Bal98]).

Definition 4 (Qualitätsmerkmal)
Ein Qualitätsmerkmal ist ein Satz von Eigenschaften eines Software-Produkts, anhand dessen seine Qualität beschrieben und beurteilt wird. Ein Software-Qualitätsmerkmal kann über mehrere Stufen in Teilmerkmale verfeinert werden [Joi01][2].

In der vorliegenden Arbeit wird der Begriff „Qualitätsattribut“ synonym zum Begriff „Qualitätsmerkmal“ verwendet.

Definition 5 (Qualitätsindikator)
Ein Qualitätsindikator ist eine ausgewiesene Eigenschaft eines Softwareprodukts, die zu den Qualitätsmerkmalen in Beziehung gesetzt werden können, z.B. Pfadlänge, modularer Aufbau, Programmstruktur, Kommentare [Joi01][3].

Die Verfeinerung von Qualität in Merkmale und Teilmerkmale kann wie in Abb. 2.1 als gerichteter Graph dargestellt werden. Die Qualitätsindikatoren können jeweils mehreren Teilmerkmalen zugeordnet werden. Über Metriken werden Qualitätsindikatoren Werten zugeordnet, die objektiv aus den Software-Artefakten gewonnen werden können.

[2]Deutsche Übersetzung in Anlehnung an [Bal98]
[3]Deutsche Übersetzung in Anlehnung an [Bal98]

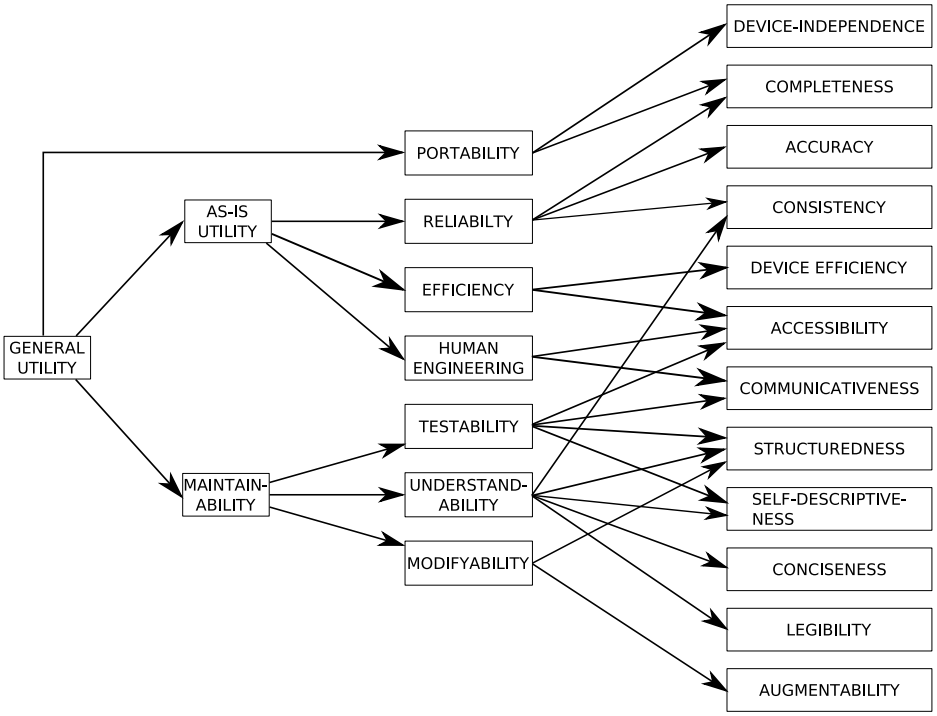

Abbildung 2.1.: Qualitätsbaum aus [BBK+78]

Definition 6 (Metrik)
Eine Metrik ist eine quantitative Skala und Methode, mit der der Wert bestimmt werden kann, den ein Qualitätsindikator für ein bestimmtes Software-Produkt aufweist [Joi01][4].

Durch die Untergliederung von Qualität in Qualitätsmerkmale, -teilmerkmale und -indikatoren und die Verwendung von Metriken, wird die Qualität eines Softwareprodukts im Entwicklungsprozess mess- und vergleichbar gemacht. Die Metriken und Indikatoren müssen für die jeweiligen Artefakte, die im Entwicklungsprozess von Softwareprodukten entstehen, angepasst verwendet werden. Für eine Architekturspezifikation können nicht die gleichen Qualitätsindikatoren und Metriken zur Beurteilung der Qualität herangezogen werden, wie für Quellcode.

Bei der Evaluation von Softwarearchitekturen handelt sich um ein Mittel der analytischen Qualitätssicherung. Dabei können Qualitätsindikatoren, soweit diese in der Modellierung der Softwarearchitektur ausreichend berücksichtigt wurden, für die Bestimmung eines Qualitätsmaß verwendet werden. Allerdings gibt

[4]Deutsche Übersetzung in Anlehnung an [Bal98]

es Qualitätsmerkmale, für die eine quantitative Bestimmung einer Ausprägung nur schwer durchzuführen bzw. mit geringer Aussagekraft behaftet ist (z.B. Wartbarkeit). Für diese Qualitätsmerkmale dient die analytische Qualitätssicherung der Softwarearchitektur eher der Aufdeckung (und späteren Korrektur) von Missständen.

2.2. Softwarearchitektur

An den zahlreichen Definitionen für Softwarearchitektur [SEI08] lässt sich erkennen, dass der Begriff „Softwarearchitektur“ nicht unumstritten ist und viele unterschiedliche Aspekte umfasst. Mit der IEEE 1471 [IEE00] wurde ein Versuch unternommen, die Aspekte von Architekturbeschreibungen und auch von Softwarearchitektur herauszugreifen und festzuschreiben, für die ein Konsens existiert [MEH01]. Obwohl die IEEE 1471 ein Standard für Architekturbeschreibungen ist, wird in ihr auch eine Definition für Softwarearchitektur gegeben.

Die IEEE 1471 [IEE00] definiert Softwarearchitektur[5] wie folgt:

> *Softwarearchitektur ist die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, deren Beziehungen zueinander und zur Umgebung sowie den Prinzipien, die den Entwurf und die Evolution des Systems bestimmen.*[IEE00][6]

Ein wichtiger Aspekt des Architekturmodells der IEEE 1471 ist, dass die Softwarearchitektur eine Eigenschaft eines Systems ist. Losgelöst von einem System kann es keine Softwarearchitektur geben. Auch wenn die Softwarearchitektur für ein noch zu erstellendes oder ein fiktives System erarbeitet wird, beschreibt diese die Eigenschaften des zu erstellenden oder fiktiven Systems. Die Prinzipien, die den Entwurf und die Evolution eines Systems bestimmen, sind jedoch nicht als Eigenschaft eines Systems zu sehen. Deshalb wird in diesem Dokument die folgende Definition von Softwarearchitektur aus [BCK03] bevorzugt.

> **Definition 7 (Softwarearchitektur)**
> *Die Softwarearchitektur eines Systems beschreibt dessen Software-Struktur respektive dessen -Strukturen, dessen Software-Bausteine sowie deren sichtbare Eigenschaften und Beziehungen zueinander. [BCK03]*[7]

[5] Eigentlich wird in der IEEE 1471 nicht Softwarearchitektur, sondern Architektur (im Sinne einer Systemarchitektur) definiert. Die Definition wird jedoch, wie in [Has06], auf Softwarearchitektur übertragen.

[6] Deutsche Übersetzung aus [Has06] überbernommen.

[7] Deutsche Übersetzung aus [VAC+05] überbernommen.

Die Definition von Softwarearchitekturen wird von *Bass et al.* auf die Struktur respektive die Strukturen eines Systems zurückgeführt. Durch die Verwendung des Plural wird deutlich, dass die Softwarearchitektur eines Systems mehrere Aspekte betreffen kann. Bei den in der Definition genannten Software-Bausteinen eines Systems kann es sich dabei sowohl um Module im Sinne von *Parnas* in [Par72], um Prozesse oder Leichtgewichtprozesse, um Datenbankmanagementsysteme, Programbibliotheken, Programmverzeichnisse oder um beliebige weitere Einheiten der Strukturierung handeln. Aus diesem Grund wird die Definition von *Bass et al.* abstrakt gehalten. Auch nennen *Bass et al.* in ihrer Definition Software-Bausteine statt Komponenten. Dies liegt daran, dass der Begriff der Komponente mit dem Verständnis einer austauschbaren Einheit der Kapselung von Programmlogik assoziiert wird. Es handelt sich bei den Einheiten der Strukturierung (Software-Bausteinen) jedoch nicht notwendigerweise immer um Komponenten.

Für die vorliegende Arbeit wird festgehalten, für jede Möglichkeit der Strukturierung gilt, dass sie aus einer Menge von Einheiten der Strukturierung, deren Topologie (Möglichkeit der Interaktion zwischen den Einheiten der Strukturierung) und Interaktionsform besteht. Die Einheiten der Strukturierung können wiederum selbst eine eigene Architektur haben. Je nach Abstraktionsgrad der Softwarearchitektur des Gesamtsystems wird die Softwarearchitektur einer Einheit der Strukturierung als Teil der Architektur des Gesamtsystems betrachtet oder abstrahiert.

(Software-)Bausteine werden im weiteren Verlauf der vorliegenden Arbeit auch als Elemente oder Einheiten einer (Software-)Architektur bezeichnet. Komponenten sind Einheiten der Strukturierung, die Programmlogik kapseln und bezüglich ihrer Systemumgebung so definiert sind, dass sie durch andere Komponenten austauschbar sind, ohne dass Kenntnisse über die innere Struktur ihrer Programmlogik notwendig sind.

Neben der Definition der Softwarearchitektur wird in [VAC+05] die Definition auch auf die Systemarchitektur ausgeweitet.

> **Definition 8 (Systemarchitektur)**
> *Die Systemarchitektur eines Systems beschreibt dessen Struktur respektive dessen Strukturen, dessen Bausteine (Software- und Hardware-Bausteine) sowie deren sichtbaren Eigenschaften und Beziehungen zueinander.[VAC+05]*

Zusätzlich zum Verständnis der Softwarearchitektur als Eigenschaft eines Systems gibt es noch das Verständnis der Softwarearchitektur als Disziplin (vgl. [VAC+05, S.48]). In der Disziplin der Softwarearchitektur – also der Teildisziplin

des Software Engineerings, bei der das Entwerfen und Gestalten von Softwarearchitekturen betrachtet wird – wird u.a. die hierarchische Zerlegung von Softwaresystemen betrachtet. Es existieren Regeln und Leitlinien, wie ein Softwaresystem zu bestimmten Zwecken zergliedert werden kann. Hauptanliegen der Zergliederung eines Softwaresystems ist die Reduktion der Komplexität, die zum einen durch die Aufteilung in mehrere Bestandteile und zum anderen durch Abstraktion von Details erlangt wird. Abgesehen von der Reduktion der Komplexität werden auch andere Aspekte durch die Zergliederung beeinflusst. Insbesondere Qualitätseigenschaften des resultierenden Systems können durch die Architektur des Systems maßgeblich beeinflusst werden.

Im Abschnitt 2.2.1 wird auf den Zusammenhang zwischen Softwarearchitektur und Softwarequalität eingegangen. Es wird gezeigt, auf welche Weise die Qualitätseigenschaften eines Softwaresystems durch die Architektur beeinflusst werden können.

Durch die oben gegebenen Definitionen wird deutlich, dass sowohl die Software- als auch die Hardwarearchitektur Bestandteil der Systemarchitektur sind. Gleichermaßen kann auch die Softwarearchitektur wiederum aus weiteren „Architekturen“ bestehen. Aus diesem Grund werden in der Definition von *Bass et al.* auch die „Strukturen“ im Plural genannt. Je nachdem, welcher Aspekt der Strukturierung eines Softwaresystems betrachtet wird (welcher Standpunkt eingenommen wird), entsteht eine entsprechende Sicht des Systems. Konzepte und Begriffe der Architekturbeschreibung wie z.B. „Standpunkt“ und „Sicht“ sowie die gängigsten Standpunkte werden in 2.2.2 erläutert.

2.2.1. Softwarequalität durch Softwarearchitektur beeinflussen

Um bewerten zu können, inwieweit eine Architekturspezifikation geeignet ist, spezifizierte Qualitätsmerkmale zu erfüllen bzw. zu begünstigen, ist ein Verständnis dafür notwendig, wie die Qualität eines Systems durch die Gestaltung dessen Softwarearchitektur beeinflusst werden kann. In diesem Abschnitt wird der Zusammenhang zwischen Softwarearchitektur und Softwarequalität betrachtet.

In der Arbeit von *Parnas* „On the Criteria to be Used in Decomposing Systems into Modules“ [Par72], die zu Anfang der vorliegenden Arbeit als Beispiel für die Beeinflussung von Softwarequalität durch die Strukturierung von Softwaresystemen genannt wurde, wird auf die Beeinflussung von Softwarequalität über die geeignete Strukturierung im Sinne einer Dekomposition eines Programms in Module eingegangen. Es wird beschrieben, wie die vom Programm zu erbringenden Funktionen geeignet in Module aufgeteilt werden bzw. wie das Programm geeignet „geschnitten“ werden kann, um das Qualitätsmerkmal Wartbarkeit positiv zu beeinflussen. Diese und weitere Arbeiten befassen sich mit dem Aspekt

der Architektur als hierarchische Dekomposition von definierter Funktionalität. So schreiben z.B. *Bengtsson und Bosch* bezüglich ihrer Methode zum „Reengineering“ von Softwarearchitekturen:

> *„Each transformation leads to a new version of the architecture that has the same functionality, but different values for its quality attributes.“* [BB98]

Diese Sichtweise, dass Architektur lediglich die Dekomposition von bereits definierter Funktionalität umfasst, ist für die vorliegende Arbeit nicht ausreichend. Um dies zu illustrieren wird demonstriert, wie durch Mittel der Architektur die Verfügbarkeit eines Systems beeinflusst werden kann.

Laut *Bass et al.* werden zur Steigerung der Verfügbarkeit eines Systems Taktiken[8] zur Erkennung, Behebung und Vermeidung von Fehlern verwendet (vgl. [BCK03, Kapitel 5] sowie Abschnitt A.1). Bei den Taktiken zur Erkennung von Fehlern wird Funktionalität ins System eingebracht, durch die Fehlzustände (z.B. der Ausfall einer Komponente) erkannt werden können. Bei den Taktiken zur Behebung von Fehlern werden Teile des Systems in einer Form der Redundanz mehrfach im System bereitgestellt und durch zusätzliche Funktionalität wird geregelt, wann welche Teile des Systems genutzt werden. Bei den Taktiken zur Vermeidung von Fehlern wird durch hinzufügen zusätzlicher Funktionalität vermieden, dass das System in Zustände gerät, in welchen eine Beeinträchtigung der Verfügbarkeit wahrscheinlich ist.

Aus dem Beispiel wird deutlich, dass es Fälle gibt, bei welchen die Gestaltung der Architektur eines Systems mit der Veränderung der Funktionalität des Systems oder von Teilsystemen einher geht. Im Falle des Qualitätsmerkmals Verfügbarkeit, verändert sich der Umfang der für den Nutzer nutzbaren Funktionalität nicht. Für die Gestaltung der Architektur eines Systems, für welches das Qualitätsmerkmal der Verfügbarkeit von Belang ist, ist es jedoch wichtig, die Dekomposition der für das System definierten Funktionalität in Kombination mit ggf. hinzuzunehmender Funktionalität für die Beeinflussung der Verfügbarkeit zu betrachten. Im Falle anderer Qualitätsmerkmale (z.B. Benutzbarkeit oder Funktions- und Informationssicherheit) wird das Hinzufügen zusätzlicher Funktionalität durch die Gestaltung der Architektur des Systems zusätzlich an der Nutzerschnittstelle sichtbar.

Im Rahmen der vorliegenden Arbeit gilt, dass es drei Möglichkeiten gibt, die Qualitätsmerkmale eines Systems durch die Gestaltung von dessen Softwarearchitektur zu beeinflussen:

[8] *Bass et al.* bezeichnen mit diesem Begriff sämtliche Möglichkeiten der Gestaltung der Architektur, die zur Beeinflussung der Qualitätsmerkmale des Systems verwendet werden können.

1. Durch Veränderung der Strukturierung des Systems. Ein Beispiel dafür ist die Kapselung von Informationen, um die Abhängigkeit zwischen Modulen zu mindern (vgl. Abschnitt A.2 zur Änderbarkeit auf Seite 172).

2. Durch Veränderung der Funktionalität des Systems (insbesondere durch das Hinzufügen neuer Funktionalität). Ein Beispiel dafür ist das Hinzufügen eines Identifikations- und Authentifikationsmechanismus in Kombination mit Komponenten zur Zugriffskontolle, um Vertraulichkeits- und Integritätsanforderungen zu erfüllen (vgl. Abschnitt A.4 zur Informationssicherheit auf Seite 175).

3. Durch eine Kombination von zusätzlicher Funktionalität und einer Veränderung der Strukturierung des Systems. Genau genommen geht das Hinzufügen von Funktionalität immer mit einer Änderung der Strukturierung des Systems einher. Allerdings kann zwischen zusätzlicher Funktionalität, die eine Veränderung an mehreren Modulen des Systems nach sich zieht, und Funktionalität, die lediglich an einem Teilbereich des Systems eingebunden wird und ohne eine notwendige Umstrukturierung mehrerer Entwurfskomponenten einher geht, unterschieden werden. Ein gutes Beispiel für eine Beeinflussung der Qualität eines Systems durch das Hinzufügen neuer Funktionalität in Kombination mit der Veränderung der Struktur eines Systems, ist das Hinzufügen der Möglichkeit, eine Operation rückgängig zu machen (vgl. Ausführungen zur Benutzbarkeit in Abschnitt A.6 auf den Seiten 177f). Um diese Funktionalität zu ermöglichen, muss sie in mehreren Komponenten des Systems vorgesehen werden. So muss z.B. die Benutzeroberfläche entsprechend angepasst werden, die Informationen über den ursprünglichen Zustand des Systems müssen gespeichert werden, die Komponenten, die auf den Zuständen (bzw. Datenbeständen) des Systems operieren, müssen entsprechende Operationen zur Verfügung stellen und evtl. muss zusätzliche Programmlogik für erlaubte und unerlaubte Zustandsänderungen eingebaut werden.

In [BCK03, Kapitel 5] zeigen *Bass et al.*, wie durch die Gestaltung der Architektur Qualitätsmerkmale eines Systems beeinflusst werden können. Es werden Taktiken zur Beeinflussung der Qualitätsmerkmale Verfügbarkeit, Änderbarkeit, Performanz, Informationssicherheit, Testbarkeit und Benutzbarkeit erläutert. *Bass et al.* befassen sich mit diesen Qualitätsmerkmalen, weil es sich dabei nach ihrer Auffassung um die wichtigsten und gebräuchlichsten Qualitätsmerkmale von Softwaresystemen handelt (vgl. [BCK03, S.79]). Unabhängig davon, ob dies stimmt, bieten die Ausführungen von *Bass et al.* zu diesen Qualitätsmerkmalen eine gute Basis, für die weiteren Ausführungen in dieser Arbeit. Eine Zusammenfassung der Taktiken, wie sie durch *Bass et al.* beschrieben werden, wird in Anhang A gegeben. Im weiteren Verlauf der vorliegenden Arbeit wird eine

Ontologie als Grundlage für die erarbeitete Methode zur Architekturevaluation erstellt (siehe Abschnitt 4.2.1 auf S.76). Aus diesem Grund werden an dieser Stelle einige Anmerkungen zu den Taktiken von *Bass et al.* zusammengefasst. Die hier genannten Anmerkungen werden in die Ontologie eingearbeitet und teilweise ergänzt.

- Die von *Bass et al.* vorgenommene Kategorisierung der Taktiken geschieht nach dem Zweck, der durch den Einsatz der Taktiken verfolgt wird. Das heißt, dass eine Kategorie nach *Bass et al.* auch bereits als Taktik gesehen werden kann. Die Taktiken, die unter einer Kategorie geführt werden, sind Verfeinerungen der Taktik, die durch die Kategorie repräsentiert wird.
- Manche der von *Bass et al.* genannten Taktiken sind auch als Muster[9] bekannt (vgl. Ausführungen zu Verfügbarkeit auf S.172). Z.B. Heartbeat [Han07, S. 101].
- In Mustern werden die Taktiken verwendet (zum Teil mehrere Taktiken gleichzeitig), um Qualitätsmerkmale zu beeinflussen (vgl. [BCK03, S.123]).
- Die Kategorien sind nicht disjunkt. Das heißt: Eine Taktik kann mehreren Kategorien zugeordnet werden. Ein Beispiel hierfür ist, Daten von Berechnungen redundant zu halten, um diese nicht mehrmals über ein Netzwerk übertragen zu müssen. Diese Taktik kann sowohl unter „Ressourcenbedarf“ als auch unter „Management von Ressourcen“ eingeordnet werden.
- Die Taktiken stehen teilweise zueinander im Konflikt, teilweise sind Taktiken voneinander abhängig. Dies wird insbesondere bei den Taktiken zur Sicherheit deutlich. Die Erhaltung von Integrität oder Vertraulichkeit ist von der Identifikation und Authentifikation abhängig. Ein Beispiel für Taktiken, die zueinander im Konflikt stehen sind die Taktiken der Benutzbarkeit, bei denen Modelle im Hintergrund gepflegt werden, und die Taktiken zur Performanz zur Reduktion von Berechnungen.

2.2.2. Beschreibungsmöglichkeiten von Softwarearchitekturen

Ein wichtiger Konsens in der Gemeinschaft der Forscher des Bereichs Softwarearchitektur, der im IEEE Standard 1471 festgeschrieben wurde, ist, dass jede Software eine Softwarearchitektur hat (vgl. [MEH01]), auch wenn diese nicht explizit dokumentiert ist. Die Softwarearchitektur eines Softwaresystems ist nicht das Dokument, welches die Softwarearchitektur spezifiziert oder beschreibt, sondern eine komplexe Eigenschaft eines Systems. Obwohl jedes Softwaresystem eine Softwarearchitektur hat, ist diese nicht per se sichtbar. Die Softwarearchitektur

[9] Die Bedeutung des Begriffs „Muster“ wird in Abschnitt 2.3 auf Seite 24 diskutiert.

eines Systems spiegelt sich in diversen Artefakten, die bei der Softwareentwicklung entstehen, wider. Dazu gehören z.B. sowohl Quell- als auch Objektcode sowie Bibliotheken, Laufzeitumgebungen oder gar das Verhalten des Systems zur Laufzeit. Da die Architektur einer Software in Artefakten aufgrund anderer in den Artefakten enthaltenen Informationen nur schwer zu erkennen ist, ist es sinnvoll die Softwarearchitektur in gesonderten Artefakten zu beschreiben. Da Softwarearchitektur in all ihren umfassenden Aspekten komplex ist, ist es zudem auch noch sinnvoll für einzelne Aspekte eigene Beschreibungen anzufertigen.

Entsprechend dieser Überlegungen wurde der IEEE Standard 1471 konzipiert. In Abbildung 2.2 ist das Konzeptmodell der IEEE 1471 abgebildet. Die wichtigsten Zusammenhänge werden hier zusammenfassend wieder gegeben. Für eine detailliertere Beschreibung wird auf den IEEE Standard 1471 selbst verwiesen [IEE00].

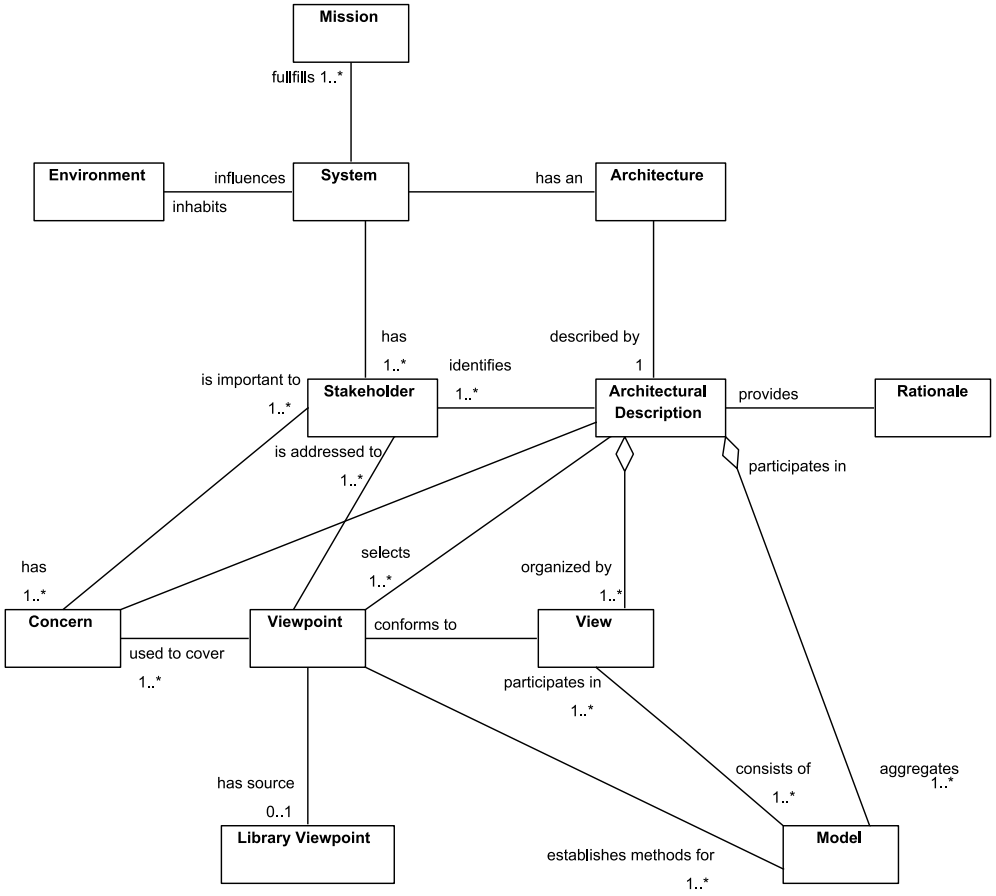

Abbildung 2.2.: Konzeptmodell für Architekturbeschreibungen ([IEE00])

Wie bereits erwähnt, ist die Softwarearchitektur eine Eigenschaft eines Systems. Dies wird im Konzeptmodell der IEEE 1471 durch die Relation zwischen den Entitäten „System“ und „Architecture“ dargestellt. Die Architektur wird durch eine Architekturbeschreibung beschrieben (Relation zwischen „Architecture“ und „Architectural Description“). Die Architekturbeschreibung umfasst mehrere Sichten (Relation zwischen „Architectural Description“ und „View“). Jede Sicht ist konform zu einem Standpunkt (Relation zwischen „View“ und „Viewpoint“). Das heißt, dass die Sicht die Aspekte der Architektur eines Systems beleuchtet, die durch den Standpunkt vorgeschrieben werden. Die Sicht entspricht dabei den Regeln und Richtlinien, die im Standpunkt definiert sind. Ein Standpunkt stellt einen (oder mehrere zusammenhängende) Belang(e) (engl. *Concern*) in den Vordergrund (Relation zwischen „Viewpoint“ und „Concern“). Die Belange kommen von Stakeholdern, die ein Interesse am System und an Architekturbeschreibungen (genauer Sichten der Architekturbeschreibung), die die Architektur des Systems bezüglich ihrer Belange beschreiben, haben (Relationen zwischen „System“ und „Stakeholder“, zwischen „Stakeholder“ und „Concern“ und zwischen „Stakeholder“ und „Viewpoint“). Für die Beschreibung aller weiteren Entitäten und Beziehungen sei auf die IEEE 1471 verwiesen [IEE00].

Für das vorliegende Dokument wird festgehalten, dass eine Sicht auch immer eine Abstraktion der gesamten Architektur des Systems ist, da sie lediglich die Informationen der Architektur zu Belangen des entsprechenden Standpunkts dokumentiert. Aspekte der Architektur, die nicht den Belangen des Standpunkts der Sicht entsprechen, werden in der Sicht abstrahiert.

Des Weiteren werden für das vorliegende Dokument die folgenden beiden Definitionen aus der IEEE 1471 übernommen:

Definition 9 (Standpunkt)
Ein Standpunkt ist eine Spezifikation der Richtlinien für die Konstruktion und Nutzung einer Sicht. Ein Standpunkt enthält eine Anleitung oder Vorlage, in welcher die Zwecke und die Zielgruppe für eine Sicht und die Techniken zu dessen Erzeugung und Analyse beschrieben werden, um daraus einzelne Sichten entwickeln zu können [IEE00].[10]

Definition 10 (Sicht)
Eine Sicht ist eine Repräsentation eines gesamten Systems aus der Perspektive einer Menge von verwandten Belangen [IEE00].[11]

[10] Leicht abgewandelte Übersetzung durch den Autor der vorliegenden Arbeit.
[11] Übersetzung des Autors der vorliegenden Arbeit.

Sichtenmodelle

Um die Architektur eines Systems umfassend zu beschreiben, wurden verschiedene Sichtenmodelle erzeugt. Die bekanntesten sind die Sichtenmodelle von *Kruchten* [Kru95], *Hofmeister et al.* [HNS99, SNH95] und von *Clements et al.* [CBB+02].

Sichtenmodell nach Kruchten *Kruchten* schlägt zur umfassenden Beschreibung eines Systems die Sichten[12] „Logische Sicht“ (engl. *Logical View*), „Prozesssicht“ (engl. *Process View*), „Entwicklungssicht“ (engl. *Development View*), „Physische Sicht“ (engl. *Physical View*) und „Szenarien“ (engl. *Scenarios*) vor. Dabei wird in der logischen Sicht die Dekomposition des Systems nach seinen zentralen funktionalen Anforderungen dargestellt. In der Prozesssicht werden die Aspekte der Ausführung des Systems zur Laufzeit dargestellt. Die Organisation der Programme als Quellcode in Module oder Bibliotheken wird in der Entwicklungssicht dargestellt. Die Verteilung der Programme oder Prozesse auf Hardwareeinheiten wird in der physischen Sicht dargestellt. Die fünfte Sicht ist zu den anderen Sichten redundant und stellt die zentralen Use-Cases des Systems dar. Sie dient zum einen der Entdeckung und Definition der zentralen Architekturentitäten und zum anderen der Validierung der anderen Sichten. Für alle Sichten gibt *Kruchten* Vorschläge für geeignete Notationen.

Sichtenmodell nach Hofmeister et al. *Hofmeister et al.* beschreiben in ihrem Sichtenmodell die vier Sichten „Konzeptuelle Sicht“ (engl. *Conceptual Architecture View*), „Modulsicht“ (engl. *Module Architecture View*), „Ausführungssicht“ (engl. *Execution Architecture View*) und „Codesicht“ (engl. *Code Architecture View*). Die von *Hofmeister et al.* beschriebenen Sichten korrespondieren in Teilen mit den Sichten von *Kruchten*. So entspricht die konzeptuelle Sicht nach *Hofmeister et al.* von den in dieser Sicht dargestellten Aspekten der logischen Sicht nach *Kruchten*, die Ausführungssicht nach *Hofmeister et al.* von den in dieser Sicht dargestellten Aspekten der Prozesssicht nach *Kruchten* und die Modulsicht und Codesicht nach *Hofmeister et al.* von den in dieser Sicht dargestellten Aspekten der Entwicklungssicht nach *Kruchten*. Für die physische Sicht sowie für die Szenarien nach *Kruchten* gibt es im Sichtenmodell von *Hofmeister et al.* keine entsprechende Sichten. Auch *Hofmeister et al.* stellen die von ihnen vorgeschlagenen Sichten mit einer eigenen Notation dar. Diese Notation basiert auf der UML.

[12] Nach der Nomenklatur der IEEE 1471 handelt es sich nicht um Sichten, sondern um Standpunkte.

Sichtenmodell nach Clements et al. *Clements et al.* geben keine Sichten, sondern Sichtenarten (engl. *Viewtypes*) vor. Dies entspricht dem Gedanken, der auch bei der Definition des IEEE Standards 1471 verfolgt wurde, dass eine Sicht nach den Bedürfnissen derjenigen (Stakeholder) gestaltet werden soll, die die Sicht verwenden. Also die Informationen in der Sicht enthalten sein sollen, die den Belangen der Stakeholder entsprechen. Da diese Belange von Softwareentwicklungsprojekt zu Softwareentwicklungsprojekt sehr unterschiedlich sein können, ist es nicht sinnvoll eine feste Menge an Sichten mit genormten Notationen in einem Sichtenmodell festzulegen und für alle Entwicklungen gleichermaßen zu verwenden. Die drei von *Clements et al.* vorgeschlagenen Sichtenarten „Modul Sichtart“ (engl. *Module Viewtype*), „Komponenten und Konnektoren Sichtart“ (engl. *Component-and-Connector Viewtype*) und „Zuweisungssichtart“ (engl. *Allocation Viewtype*) unterteilen die möglichen Sichten in Kategorien. Sichten der Modul Sichtart beschreiben die statischen Anteile eines Systems (z.B. die Modul- und Codesicht nach *Hofmeister et al.* oder die Entwicklungssicht nach *Kruchten*). Sichten, die der Komponenten und Konnektoren Sichtart zugerodnet werden können, beschreiben die dynamischen Anteile eines Systems (z.B. die Ausführungssicht nach *Hofmeister et al.* oder die Prozesssicht nach *Kruchten*). Unter die Zuweisungssichtart fallen alle Sichten, die Anteile des Systems aus den statischen oder dynamischen Anteilen auf weitere Strukturen abbilden (z.B. auf Hardware; z.B. die physische Sicht nach *Kruchten*).

Architektur begründen

Die bisherigen Konzepte zur Darstellung der Architektur, dienen der expliziten Repräsentation eines Soll- oder Ist-Stands der Eigenschaft eines Systems. Da der Prozess des Entwurfs eines Systems aber immer die Entscheidung zwischen Alternativen impliziert, sollte eine Architekturspezifikation oder -dokumentation zusätzlich zur Darstellung der geforderten oder vorhandenen Ausprägung der Eigenschaft „Architektur“ auch immer diesen Entscheidungsprozess dokumentieren. Damit wird auch der Bezug zu den Anforderungen expliziert. Auch im Standard der IEEE 1471 wird eine Dokumentation der Begründungen für Architekturentscheidungen gefordert.

2.3. Muster und Architekturmuster

Während in anderen Ingenieursdisziplinen Sammlungen von Lösungen zu bekannten Problemen der Disziplin weit verbreitet und auch in der Ausbildung stark vertreten sind (siehe z.B. [SG06, NWH05, NW03, NW86]), stehen Werke dieser Art im Bereich des Software Engineerings und insbesondere des Entwurfs

von Software- und Systemarchitekturen noch in den Anfängen. Die Dokumentation von Mustern, die in den letzten zehn bis 15 Jahren stattgefunden hat, beginnt, diese Lücke zu füllen.

Christopher Alexander, ein Professor der (Gebäude-)Architektur, definierte in einer Serie von fünf Büchern ([Ale79, AIS+77, ASA+75, ADMC85, Ale81]) eine Theorie über Muster und deren Nutzung zusammen mit einer Reihe von Mustern für den Entwurf von Gebäuden und Städten. Die Theorien zu Mustern wurden erstmals von Kent Beck und Ward Cunningham in den Bereich des Softareentwurfs übertragen [BC87]. Große Bekanntheit und Beliebtheit erlangten Muster mit dem Buch „Design Patterns“ [GHJV95] von der „Gang of four“ und danach auch mit der Bücherserie „Pattern Oriented Software Architecture“ (POSA), von denen es inzwischen fünf gibt ([BMR+96, SRSB00, KJ04, BHS07b, BHS07c]).

Beispiele. Ein bekanntes Architekturmuster[13] ist MVC (Model-View-Controller) [BMR+96], bei welchem ein Softwaresystem, das eine Benutzerschnittstelle für einen menschlichen Benutzer hat, in drei Teile aufgeteilt wird. Ein Teil, in dem die Modellierung des durch das System betrachtete Problem und entsprechende Berechnungen durchgeführt werden (Model), ein Teil, mit dem die Darstellung der im Modell gespeicherten Informationen durchgeführt wird und ein Teil (View), mit welchem die Aktionen des Benutzers angenommen werden und die notwendigen Berechnungen im Model initiiert werden (Controller). Detailliertere Beschreibungen sind aus [BMR+96] zu entnehmen.

Ein weiteres Beispiel für ein Architekturmuster ist Heartbeat [Han07]. Bei diesem Muster werden Teile eines Systems „überwacht“, indem von den zu überwachenden Teilen des Systems in regelmäßigen Zeitabständen ein Signal (Heartbeat) an eine Kontrollinstanz ausgeht, welches darauf hindeutet, dass diese Systemteile ihre Funktion noch erfüllen bzw. nicht ausgefallen sind. Werden von einem Systemteil keine Heartbeat-Signale mehr empfangen, können durch die Kontrollinstanz Maßnahmen eingeleitet werden, um die korrekte Funktionsweise der betroffenen Systemteile wiederherzustellen. Dieses Muster ist ein gutes Beispiel für eine Lösung für ein wiederkehrend auftretendes Problem im Bereich der Softwareentwicklung.

Weitere Bekannte Beispiele für Architekturmuster sind die Muster Layers und Pipes and Filters [BMR+96].

Heutzutage sind bereits über zweitausend Muster in der Literatur dokumentiert [Boo07]. Es existieren Konferenzen, die eigens zur Diskussion und Dokumentation von Mustern ins Leben gerufen wurden. Die bekanntesten davon sind die

[13] Definitionen für die Begriffe Muster und Architekturmuster folgen auf Seite 27

PLoP (**P**attern **L**anguages **o**f **P**rograms) Konferenzen und ihre Entsprechungen in anderen Ländern und Kontinenten.

Muster werden im Bereich des Softwareentwurfs oft als bereits bestehende, fertige Lösung für ein „Standardproblem“ verstanden. In welcher Hinsicht dies nicht ganz zutrifft wird in 2.3.1 erörtert, wo auch eine Definition des Begriffs erfolgt. Im Anschluss (Abschnitt 2.3.2) wird dargelegt, wie Muster beschrieben werden und welche Klassifikationen von Mustern existieren. Muster können zu anderen Mustern in Beziehung stehen. Welche Beziehungen zwischen Mustern existieren wird in 2.3.3 erläutert.

2.3.1. Kernideen und Definition

Buschmann et al. definieren in [BMR+96, S.8] (POSA 1) ein Muster wie folgt:

> *A* pattern for software architecture *describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities and relationships, and the ways in which they collaborate.*
>
> *Ein* Muster für Softwarearchitektur *beschreibt ein bestimmtes wiederkehrendes Entwurfsproblem, welches in spezifischen Entwurfskontexten auftaucht und enthält ein etabliertes generisches Schema zu dessen Lösung. Das Lösungsschema wird durch die Beschreibung der Komponenten, aus denen es besteht, ihrer Verantwortlichkeiten und Beziehungen und die Art ihrer Interaktion spezifiziert.*[14]

Diese Definition ist in der Domäne der Muster häufig zitiert. Sie enthält wichtige Aspekte von Mustern:

- Muster adressieren wiederkehrende Probleme. Einmalige Probleme werden durch spezifische Lösungen, nicht durch Muster, adressiert.
- Muster bieten Lösungen zu Problemen in einem Kontext. Tritt das Problem in einem anderen Kontext auf, ist es evtl. durch das Lösungsschema des Musters nicht mehr lösbar.
- Muster sind etabliert. Die Forschungsgemeinschaft, die sich im Rahmen des Softwareentwurfs mit Mustern beschäftigt (fortan als Pattern-Community bezeichnet), sieht eine generische Lösung erst dann als Muster an, wenn sie sich im realen Einsatz bewährt hat.

[14] Übersetzung des Autors der vorliegenden Arbeit.

- Muster bieten generische Schemata zur Lösung von Problemen. Muster sind keine spezifischen Lösungen.

Da sich die obige Definition im speziellen auf Muster für Softwarearchitekturen bezieht, wird in ihr noch der Bezug zur Softwarearchitektur hergestellt, indem definiert wird, dass das Lösungsschema aus einer Beschreibung von Komponenten, Verantwortungen, Beziehungen und Interaktionen besteht.

Kritik an der Definition von Mustern nach POSA

Die Definition von Mustern von *Buschmann et al.* ist für die Verwendung in der vorliegenden Arbeit nicht ausreichend. Im Folgenden wird erörtert, welche Punkte an der Definition verbessert werden können und im Anschluss eine eigene Definition gegeben.

- In der Definition von *Buschmann et al.* wird der Kontext, in dem ein Problem auftritt, explizit in die Definition aufgenommen. Der Kontext stellt jedoch lediglich eine Möglichkeit dar, das Problem, welches durch das Muster gelöst wird, genauer zu spezifizieren. Die Verwendung eines Kontextes zur genaueren Beschreibung der Anwendbarkeit des Musters ist bei der Beschreibung von Mustern üblich (siehe Abschnitt 2.3.2) und sehr hilfreich. Es ist aber nicht Kern dessen, was ein Muster ausmacht.
- Ein Muster nach der Definition von *Buschmann et al.* beschreibt ein wiederkehrendes Problem. Diese Aussage ist nicht genau genug. Würde es sich um ein wiederkehrendes Problem handeln, wäre der Einsatz der gleichen Lösung ausreichend. Stattdessen werden in Mustern generische Lösungsschemata beschrieben. Also handelt es sich um eine wiederkehrende Problemklasse von Problemen gleicher Art.
- Was die Bedeutung des Begriffs „Muster“ im normalen Sprachgebrauch ausmacht, ist der wiederkehrende Charakter einer Eigenschaft in einer Menge von betrachteten, unterschiedlichen Elementen. Bei den Mustern in der Disziplin des Softwareentwurfs liegt der wiederkehrende Charakter in der Gestaltung der Lösung des Problems. Da ein Muster jedoch ein generisches Lösungsschema bietet, gibt es auch Anteile eines Musters, die bei jedem Einsatz des Musters variieren. Eine Aufnahme der wiederkehrenden sowie variierdenden Anteile eines Musters in die Definition präzisiert den Musterbegriff. Die in diesem Dokument erarbeitete Definition, hebt deshalb hervor, dass ein Muster immer aus wiederkehrenden sowie aus variierenden Anteilen besteht.
- Der Pattern-Community ist es wichtig, dass ein Muster eine etablierte (engl. well-proven) Lösung zu einem Problem bietet. Deshalb können nach

Auffassung der Pattern-Community generische Lösungsschemata erst dann als Muster bezeichnet werden, wenn diese sich im realen Einsatz bewährt haben. Für die vorliegende Arbeit steht diese strikte Erprobung der Lösung nicht im Vordergrund. Dennoch wird ein Muster als eine Form angesehen, wie Expertenwissen strukturiert beschrieben und verwendet werden kann.

- Wie dieses Expertenwissen in geeigneter Weise strukturiert wird, hängt von der Art der Verwendung ab, für die die Strukturierung des Expertenwissens vorgesehen ist. Deshalb wird die Form der Strukturierung in der Definition des vorliegenden Dokuments explizit offen gelassen. Dass das Expertenwissen geeignet strukturiert werden muss, wird jedoch von der Definition vorgeschrieben.

Aus diesen Gründen wird ein Muster für die vorliegende Arbeit wie folgt definiert:

Definition 11 (Muster)
Ein Muster kapselt Expertenwissen, welches zur Lösung von Problemen einer Problemklasse genutzt werden kann, in einer für die Anwendung des Musters geeignet strukturierten Form. Der Lösungsraum, der durch das Muster aufgespannt wird, besteht sowohl aus wiederkehrenden als auch aus variierenden Anteilen.

Da sich die vorliegende Arbeit mit Softwarearchitektur und Mustern aus der Domäne des Softwareentwurfs befasst, wird auf der Definition des Musters aufbauend eine Definition von Architekturmustern gegeben.

Definition 12 (Architekturmuster)
Ein Architekturmuster ist ein Muster, bei welchem die wiederkehrenden sowie die variierenden Anteile die Architektur eines Systems (also die Einheiten der Stukturierung, die Topologie dieser Einheiten und die Interaktionsformen zwischen diesen Einheiten) oder eines Teils eines Systems bestimmen.

Dass Architekturmuster die Architektur eines Systems oder eines Teils eines Systems bestimmen, muss nicht bedeuten, dass durch die Architekturmuster die Architektur vollständig definiert wird. Es kann auch bedeuten, dass Einschränkungen bezüglich der Bestandteile der Architektur durch das Architekturmuster definiert werden. Beispielsweise kann durch ein Architekturmuster definiert werden, dass Einheiten vorliegen müssen, die definierte Funktionen erfüllen müssen und offen lassen, durch welche oder wie viele Einheiten diese Funktionen erfüllt werden müssen. In gängigen Beschreibungen von Architekturmustern wird diesbezüglich von Verantwortlichkeiten oder Rollen, die in Einheiten der Implementierung zu hinterlegen sind, gesprochen.

Als Entwurfsmuster (engl. *Design Pattern*) werden in der vorliegenden Arbeit Muster verstanden, die nur für ein Paradigma (z.B. Objektorientierte Softwareentwicklung) Verwendung finden können (z.B. das Singelton-Muster aus [GHJV95]). Diese Unterscheidung von Architektur- und Entwurfsmustern entspricht nicht der Unterscheidung aus der gängigen Literatur (z.B. [BHS07c, RH06]), in welcher die Unterscheidung davon abhängt, zu welchem Grad das jeweilige Muster die Strukturierung des Gesamtsystems oder nur eines Teilsystems regelt. Da ein Teilsystem aber wiederum ein Gesamtsystem sein kann, wird diese Unterscheidung in der vorliegenden Arbeit nicht verwendet. Programmiersprachenspezifische Muster (auch als Idiome bezeichnet) werden in der vorliegenden Arbeit nicht benötigt und werden daher auch nicht weiter betrachtet.

Begriffe, die aus der Neudefinition von Mustern hervorgehen

Mit der Aufnahme der Unterscheidung von wiederkehrenden und variierenden Anteilen in die Definition des Musterbegriffs ergibt sich als Konsequenz, dass in einem Muster auch beschrieben wird, wie sich die variierenden Anteile eines Musters unterscheiden können und welche Auswirkungen dies auf die konkrete Lösung hat. Ergo, wie man beim Einsatz eines Musters im Lösungsraum, der vom Muster abgedeckt wird, navigieren kann und dadurch die Probleme der durch das Muster adressierten Problemklasse ausschöpfend behandeln kann. Nachfolgend werden zu diesem Zweck die Begriffe „Variationsmöglichkeit“, „Belegung“ und „Konfiguration“ definiert.

> **Definition 13 (Variationsmöglichkeit eines Musters)**
> *Als Variationsmöglichkeit eines Musters wird jeder Teil der Lösung eines Musters bezeichnet, der bei jedem Einsatz des Musters unterschiedlich realisiert werden kann.*

> **Definition 14 (Belegung einer Variationsmöglichkeit)**
> *Als Belegung einer Variationsmöglichkeit wird eine konkrete Realisierung der Variationsmöglichkeit bezeichnet. Die Menge der gültigen Belegungen einer Variationsmöglichkeit kann in der Beschreibung eines Musters definiert werden.*

> **Definition 15 (Musterkonfiguration)**
> *Sei P ein Problem aus der Problemklasse der durch Muster M adressierten Probleme. Eine Lösung, die alle wiederkehrenden Anteile von Muster M enthält und die alle Variationsmöglichkeiten des Musters M gültig belegt, heißt eine gültige Konfiguration des Musters M.*

Die genannten Definitionen werden im Folgenden am bereits erwähnten Beispiel des Musters Heartbeat erläutert. Die Einheiten der Strukturierung des Musters

Heartbeat sind die zu überwachenden Systemteile und die Kontrollinstanz. Interaktionsmöglichkeiten bestehen jeweils zwischen den zu überwachenden Systemteilen und der Kontrollinstanz. Weitere Interaktionsmöglichkeiten und Interaktionsformen (z.B. zwischen den zu überwachenden Systemteilen) werden nicht eingeschränkt. Die Form der Interaktion wurde bei der Erläuterung des Musters Heartbeat in Abschnitt 2.3 auf Seite 25 erläutert.

Aus den Beschreibungen des Musters Heartbeat [Han07] gehen unmittelbar drei Variationsmöglichkeiten hervor:

1. Die Anzahl der durch eine Kontrollinstanz zu überwachenden Komponenten kann variieren.
2. Der Zeitintervall zwischen den Heartbeat-Signalen kann variiert werden.
3. Das Heartbeat-Signal kann ohne externe Einwirkung vom zu überwachenden Systemteil ausgehen oder durch eine Anfrage durch die Kontrollinstanz initiiert werden.

Unter der Annahme, dass es für das Muster Heartbeat keine weiteren Vatiationsmöglichkeiten gibt, ist ein System bei welchem eine Kontrollinstanz zwei Systemteile überwacht (Belegung der ersten Variationsmöglichkeit), wobei die Kontrollinstanz alle drei Sekunden (Belegung der zweiten Variationsmöglichkeit) ein Heartbeat-Signal an beide zu überwachende Systemteile entsendet und die Systemteile auf die Nachricht antworten (Belegung der dritten Variationsmöglichkeit) eine gültige Konfiguration des Musters Heartbeat.

In der Definition von Mustern wird nicht gefordert, dass die wiederkehrenden und variablen Anteile explizit beschrieben werden. Es ist nur gefordert, dass ein Muster aus wiederkehrenden und variablen Anteile besteht. Die Strukturierung der in einem Muster enthaltenen Information wird in der Definition erwähnt (und somit hervorgehoben), ohne dass eine bestimmte Strukturierung vorgeschrieben wird. Im folgenden werden gängige Formen zur Strukturierung von Mustern betrachtet.

2.3.2. Klassifikations- und Beschreibungsformen

Die Klassifikation von Mustern spielt in der Praxis in Mustersammlungen eine Rolle und dient zum einen der schnellen Suche nach geeigneten Mustern durch einen Nutzer einer Mustersammlung und zum anderen der Schaffung eines Rahmens für das bessere Verständnis der Muster einer Klassifikation.

Bei genauerer Betrachtung der Definition von *Buschmann et al.* im vorangegangenen Abschnitt fällt auf, dass nicht der Begriff „Muster“, sondern der Ausdruck „Muster für Softwarearchitektur“ definiert wird. Dies liegt daran, dass

Buschmann et al. drei Arten von Muster unterscheiden: „Idiome“ (engl. *Idioms*), „Entwurfsmuster“ (engl. *Design Patterns*) und „Architekturmuster“ (engl. *Architectural Patterns*). Idiome sind programmiersprachen- oder programmierparadigmenspezifische Muster. Mit Idiomen werden Probleme adressiert, die nur in der jeweiligen Programmiersprache oder dem jeweiligen Programmierparadigma auftreten. Entwurfsmuster hingegen sind von einer Programmiersprache oder einem Programmierparadigma unabhängig für Probleme des Entwurfs anwendbar. Sie sind für den Entwurf lokal begrenzter Probleme, dessen Auswirkungen sich nicht auf das Gesamtsystem erstrecken, anwendbar. Zu den Architekturmustern gehören Muster, die sich auf den Entwurf des Gesamtsystems beziehen. Die Unterscheidung zwischen Entwurfs- und Architekturmustern ist nicht sehr präzise gefasst und vom Auge des Betrachters abhängig[15]. Es existiert jedoch in der Pattern-Community ein gemeinsames Verständnis zwischen Mustern, die der einen oder der anderen Kategorie angehören.

Zusätzlich zu dieser Klassifikation, die in der Patten-Community Verbreitung und Akzeptanz gefunden hat (siehe z.B. [CS95, VCK96, MRB97]), werden noch weitere Formen der Klassifikation verwendet. Es existieren Mustersammlungen, die Muster aus einer Domäne betrachten (z.B. Sicherheit [SFBH+05], Verteilte Systeme [BHS07b]). Die Betrachtung von Mustern aus nur einer Domäne stellt bereits eine Klassifikation dar. Zusätzlich können weitere (domänenspezifische) Klassifikationen vorgenommen werden. Beispielsweise werden die Muster für die Domäne der Sicherheit aus [SFBH+05] weiter in „Enterprise Security and Risk Management“, „Identification and Authentication (I&A)“, „Access Control Models“, „System Access Control Architecture“, „Operating System Access Control Patterns“, „Accounting“, „Firewall Architecture“, „Secure Internet Applications“ und „Cryptographic Key Management“ unterteilt, was einer Klassifizierung nach dem Zweck der Muster bzw. der durch die Muster adressierten Probleme entspricht.

Eine Klassifikation von Mustern nach den durch die Muster adressierten Qualitätsmerkmalen ist in der Praxis nur üblich, wenn der einzige Zweck des Musters in der Erfüllung eines Qualitätsmerkmals liegt. Die Klassifikation nach dem Zweck des Musters ist die üblichste Form der Klassifikation. Diese geschieht durch den Autor einer Mustersammlung nach Intuition und wird üblicherweise nicht genau abgegrenzt, sondern ggf. erläutert. Durch die mangelnde Abgrenzung der Klassifikationen und welche Muster in die jeweiligen Klassen fallen, ist es manchmal möglich, ein Muster mehreren Klassen zuzuordnen.

[15] Eine detailliertere Diskussion zur Klassifikation von Mustern nach Abstraktionsniveau wird unter [BHS07c, S.213 ff] geführt.

Typische Beschreibungselemente von Mustern

Um die wesentlichen Aspekte von Mustern zu vermitteln, werden in den meisten Mustersammlungen Abschnitte zur Beschreibung für jedes Muster aus der Sammlung vorgeschrieben. Obwohl sich die Form der Beschreibung von Mustersammlung zu Mustersammlung unterscheidet, gibt es einige Abschnitte, die in vielen Mustersammlungen verwendet werden. Die wichtigsten dieser Abschnitte werden nachfolgend kurz erläutert:

Name In den meisten bekannten Mustersammlungen wird jedem Muster ein aussagekräftiger Name gegeben.

Problem In diesem Abschnitt wird das Problem beschrieben, das durch den Einsatz des beschriebenen Musters gelöst werden kann.

Kontext Der Kontext (engl. *Context* oder *Applicability*) eines Musters ist eine Verfeinerung der Problembeschreibung. Es wird dargestellt, welche Bedingungen gelten müssen, damit das Muster zur Lösung des Problems eingesetzt werden kann. Der Kontext kann also als eine Art Vorbedingung für den Einsatz des Musters gesehen werden.

Kräfte In der ursprünglichen Mustertheorie des Professors der (Gebäude-)Architektur, *Christopher Alexander* [AIS+77], waren die Kräfte (engl. *Forces* in manchen Musterbeschreibungen auch *Motivation*) auch tatsächlich physikalische Kräfte, die bei der Anwendung des Musters in Betracht zu ziehen und durch das (durch das Muster entstehende) Bauwerk auszugleichen waren. Die Bezeichnung wurde bei der Beschreibung von Mustern für den Entwurf von Software übernommen und bezeichnet generell Aspekte, die bei der Gestaltung der Lösung zu berücksichtigen sein können. Meist sind dies mögliche Anforderungen an die Lösung. Bei einem Einsatz des Musters ist festzustellen, welche dieser Anforderungen wie gewichtet wird und die Lösung entsprechend zu gestalten. Oft handelt es sich bei den Kräften um Qualitätsanforderungen. Als solche stehen Kräfte oft auch im Widerspruch zueinander. In manchen Beschreibungen sind die Kräfte ein Teil der Problembeschreibung.

Lösung In diesem Abschnitt wird beschrieben, wie das Problem gelöst werden kann. Üblicherweise umfasst die Lösung (engl. Solution) eine Beschreibung, welche Mittel verwendet werden können, um welche Kräfte wie auszugleichen. Dies ist insbesondere dann wichtig, wenn die Kräfte zueinander im Widerspruch stehen.

Struktur Der Abschnitt der Struktur (engl. Stucture) kann als eine Verfeinerung des Abschnitts der Lösung gesehen werden. In diesem Abschnitt werden die Komponenten, die beim Einsatz des Musters Verwendung finden,

ihre Verantwortlichkeiten bezüglich der Lösung und ihre Beziehungen untereinander dargestellt. Oft ist in diesem Abschnitt auch eine graphische Darstellung vorzufinden.

Dynamik Analog zum Abschnitt der Struktur kann auch der Abschnitt der Dynamik (engl. *Dynamics* oder *Collaboration*) als Verfeinerung des Abschnitts der Lösung gesehen werden. Der Schwerpunkt liegt bei diesem Abschnitt darin zu zeigen, wie die Komponenten des Musters zur Laufzeit interagieren, um das Problem zu lösen. Auch in diesem Abschnitt ist oft eine graphische Darstellung vorzufinden.

Konsequenzen Im Abschnitt der Konsequenzen (engl. *Consequences*) wird beschrieben, welche Resultate aus dem Einsatz des Musters zu erwarten sind und warum diese das Problem lösen. Zudem wird in diesem Abschnitt auch beschrieben, welche Nachteile sich aus dem Einsatz eines Musters ergeben können. In manchen Beschreibungen wird der Teil, in dem beschrieben wird, weshalb der Einsatz des Musters das Problem löst, als Argumentation (engl. *Rationale*) bezeichnet.

Bekannte Einsätze Da in der Pattern-Community ein Muster erst als Muster bezeichnet werden kann, wenn es sich in der Praxis bewährt hat (siehe Diskussion in Abschnitt 2.3.1), gibt es in den meisten bekannten Mustersammlungen einen Abschnitt „Bekannte Einsätze“ (engl. *Known Uses*), in dem beschrieben wird, in welchen realen Anwendungen das Muster bereits erfolgreich eingesetzt wurde.

Siehe auch In diesem Abschnitt (engl. *See also* oder *Related Patterns*) wird auf weitere Muster verwiesen, die in irgendeiner Weise für die Anwendung des beschriebenen Musters von Bedeutung sein könnten. Der Zusammenhang zum beschriebenen Muster wird vom Autor des Musters anhand seiner Erfahrungen in informellen Beschreibungen hergestellt.

Zusätzlich zu den genannten Abschnitten sind die folgenden Abschnitte zur Beschreibung von Mustern üblich: Beispiel (engl. *Example*), Implementierung (engl. *Implementation* oder *Sample Code*), Auch bekannt als (engl. *Also Known As*), Gelöstes Beispiel (engl. *Example Resolved*) und Varianten (engl. *Variants*). Diese Abschnitte werden in dieser Arbeit nicht weiter vertieft. Es sei an dieser Stelle auf die Erläuterungen aus den entsprechenden Mustersammlungen verwiesen [GHJV95, BMR+96]

2.3.3. Beziehungen zwischen Mustern

Bereits 1995 hat *Zimmer* Beziehungen zwischen den Mustern von *Gamma et al.* kategorisiert [Zim95]. Inzwischen haben *Buschmann et al.* Beziehungen zwi-

schen Mustern weitergehend betrachtet. *Buschmann et al.* unterscheiden in [BHS07a, BHS07c] u.a. komplementäre Muster (engl. *pattern complements*), zusammengesetzte Muster (engl. *pattern compounds*), Mustersequenzen (engl. *pattern sequences*) und Mustersprachen (engl. *pattern languages*).

Zu den komplementären Mustern zählen *Buschmann et al.* Muster, die sich gegenseitig ergänzen (ergänzendes Komplement) sowie Muster, die Alternativen zueinander bilden (alternatives Komplement). *Buschmann et al.* begründen diese Kategoriesierung zweier unterschiedlicher Musterbeziehungen unter derselben Kategorie damit, dass alternative Muster oft keine sich gegenseitig ausschließende Alternativen bilden, sondern auch gemeinsam verwendet werden können, um ein Problem komplexerer Natur zu adressieren. Dadurch sind nach Auffassung von *Buschmann et al.* die beiden Musterbeziehungstypen nicht unterschiedlicher Natur (vgl. [BHS07c, Abschnitt 5.4, S.159]).

Unter zusammengesetzten Mustern verstehen *Buschmann et al.* bekannte Muster, die häufig in einer wiederkehrenden Kombination gebraucht werden, um ein wiederkehrendes Entwurfsproblem (Probleme aus der gleichen Problemklasse) zu adressieren. Die wiederkehrende Kombination von Mustern wird selbst wieder als ein Muster angesehen bzw. beschrieben.

Eine Mustersequenz besteht aus einer Menge von Mustern, die nacheinander für den Entwurf eingesetzt jeweils einen Kontext bilden, der wiederum die Voraussetzung für den Einsatz eines weiteren Musters aus der Mustersequenz schafft. Die Paare einer Mustersequenz sind somit jeweils ergänzende Komplemente.

Auch eine Mustersprache besteht aus einer Menge von Mustern, die in einem Zusammenhang stehen. Es handelt sich dabei um Muster, die gemeinsam für Lösungen von Problemen aus einer Problemdomäne verwendet werden. Ziel einer Mustersprache ist, dass für die Problemdomäne der Mustersprache der Entwurf eines Systems durch Verwendung der Mustersprache gesteuert werden kann. Deshalb enthält eine Mustersprache nicht nur die Muster, die beim Entwurf eines Systems der jeweiligen Problemdomäne von Relevanz sein könnten, sondern auch eine Menge von Beschreibungen, die den Prozess des Entwurfs eines Systems begleiten. Dabei werden von Mustern ausgehend die Möglichkeiten des weiteren Entwurfs mit weiteren Mustern der Mustersprache beschrieben. Betrachtet man eine Mustersprache als gerichteten Graphen, bei dem jeder Knoten ein Muster und jede Kante eine mögliche Weiterführung eines Entwurfs darstellt, so stellt ein Pfad durch den Graphen (eine Mustersequenz) einen möglichen Entwurf in der Problemdomäne dar. Die Auswahl des Pfads durch den Graphen wird durch die für den jeweiligen Entwurf relevanten Anforderungen bestimmt.

Mustersprachen wurden u.a. bereits für die Domänen der verteilten Systeme [BHS07b], der Informationssicherheit [SFBH+05], spezielle Bereiche eingebette-

ter Systeme [Pon01] sowie einige weitere Domänen (siehe [BHS07a, BHS07c]) erstellt.

Die hier kurz vorgestellten Beziehungen werden in Teilen für diese Arbeit übernommen. Welche Aspekte der hier vorgestellten Beziehungen in dieser Arbeit wie Verwendung finden wird beim Aufbau einer Ontologie für die in dieser Arbeit vorgestellte Architekturbewertungsmethode in Abschnitt 4.2.1 auf Seite 76 beschrieben.

KAPITEL 3

Stand der Verfahren zur Architekturbewertung

Mit SAAM und später auch ATAM und CBAM hat das SEI die ersten und inzwischen bekanntesten qualitativen Vorgehen zur Analyse von Softwarearchitekturen entwickelt.

Das folgende Kapitel beginnt mit einem Überblick über die existierenden Methoden zur Analyse von Softwarearchitekturen. Im Anschluss werden die bekanntesten Methoden erläutert. Anschließend werden die Methoden in eine Taxonomie eingeordnet und kritisch betrachtet.

Abschließend werden zwei weitere verwandte Arbeiten kurz vorgestellt. Es handelt sich um das Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen von *Malich* und um die Ontologie zur Unterstützung der Evaluation von Softwarearchitekturen von *Erfanian und Aliee*.

Inhalt

3.1. Arten der Architekturbewertung

Die Architekturbewertung ist eine Form der analytischen Qualitätssicherung, bei der die Architektur eines bereits existierenden oder noch zu entwickelnden Systems bewertet wird. Da die Softwarearchitektur eine Eigenschaft eines Systems ist, welche nicht per se sichtbar ist (siehe Beschreibung zur Definition von Softwarearchitektur Abschnitt 2.2, S. 15), werden bei einer Architekturbewertung nicht die Architektur selbst, sondern immer nur geeignete Repräsentationen der Architektur verwendet. Findet die Architekturbewertung vor der Implementierung des zu entwickelnden Systems statt, handelt es sich bei der „geeigneten Repräsentation“ der Architektur um die Architekturspezifikation des Systems. Findet die Bewertung nach erfolgter Implementierung statt, wird die Architekturdokumentation als Bewertungsgrundlage verwendet. In diesem Dokument wird hauptsächlich der Begriff „Architekturbeschreibung“ verwendet. Je nach Kontext handelt es sich bei der Architekturbeschreibung um eine Architekturdokumentation oder um eine Architekturspezifikation. Für die in dieser Arbeit präsentierte Methode wird angenommen, dass die Architekturbewertung vor der Implementierung des Systems durchgeführt wird. Wie für eine Architekturbewertung nach der Implementierung des Systems vorzugehen ist, kann aus dem beschriebenen Vorgehen zur Architekturbewertung abgeleitet werden. Hierzu muss zum einen eine Architekturdokumentation des zu evaluierenden Systems vorliegen und zum anderen sicher gestellt werden, dass die in der Architekturdokumentation beschriebene Architektur auch tatsächlich zu der im System vorliegenden Architektur konsistent ist. In [Kos05] beschreibt Koschke den Stand der Technik zur Rekonstruktion von Softwarearchitekturen aus einem bereits vorliegenden System. Synonym zum Begriff „Architekturbewertung“ werden die Begriffe „Architekturevaluation“ sowie „Architekturanalyse“ verwendet.

In [RH06] wird die folgende Definition von Architekturbewertung gegeben, die auch hier verwendet wird:

> **Definition 16 (Architekturbewertung)**
> *Die Architekturbewertung umfasst alle Aktivitäten zur qualitativen oder quantitativen Bestimmung der Qualitätseigenschaften einer Architekturspezifikation. Somit dient die Architekturbewertung der Überprüfung der Qualität der Architekturspezifikation.[RH06]*

Die Qualität der Architekturspezifikation beinhaltet insbesondere die Qualität der Architektur, die spezifiziert wird, sowie die Qualität der Spezifikation selbst. Das heißt: Bei der Überprüfung der Qualität der Architekturspezifikation wird sowohl geprüft, inwieweit die Anforderungen an das zu entwickelnde System durch die in der Spezifikation repräsentierte Architektur erfüllt werden als auch

inwieweit die festgelegten oder vorausgesetzten Erfordernisse an eine Architekturspezifikation durch die vorliegende Architekturspezifikation erfüllt werden (siehe Definition von Qualität auf S. 12).

Es werden *quantitative* und *qualitative* Methoden zur Architekturbewertung unterschieden [RH06, PBG07, ABC+97]. In [RH06] werden die Methoden wie folgt weiter untergliedert:

- Quantitative Techniken
  - Attribut- und kompositionsbasierte Techniken
  - Messende Techniken
  - Simulationsbasierte Techniken
- Qualitative Techniken
  - Szenariobasierte Techniken
  - Reviews
  - Fragebögen
  - Checklisten

Bei der Verwendung *quantitativer Methoden* werden konkrete Ausprägungen eines Qualitätsmaßes erzeugt. Dazu werden Annahmen über Eigenschaften der Komponenten eines Systems als Bewertungsgrundlage herangezogen und über die Regeln, die die Methode bereitstellt, miteinander kombiniert. Für die Annahmen werden Expertenschätzungen, Erfahrungswerte oder Messungen herangezogen.

Im Gegensatz zu den quantitativen Methoden wird bei der Verwendung *qualitativer Methoden* keine konkrete Ausprägung eines Werts für ein Maß eines Qualitätsmerkmals ermittelt [RH06]. Stattdessen dienen die qualitativen Methoden der Feststellung, inwieweit die untersuchte Architektur die Erlangung eines Qualitätsmerkmals in einer definierten Ausprägung überhaupt ermöglicht, ob gängige Fehler enthalten sind oder welche Alternativen zu wählen sind.

3.2. Architecture Tradeoff Analysis Method (ATAM)

Die bekannteste szenariobasierte Methode zur Evaluation von Softwarearchitekturen ist ATAM (**A**rchitecture **T**radeoff **A**nalysis **M**ethod) [KKB+98, KKC00, CKK02, BCK03], welche von Forschern des Software Engineering Institute (SEI)

der Carnigie Mellon University (CMU) entwickelt wurde. An der Benennung erkennt man, worauf bei der Entwicklung dieser Methode besonderer Wert gelegt wurde. Bei der Durchführung einer ATAM Analyse werden nicht lediglich die geforderten Qualitätsmerkmale in der Architektur identifiziert, sondern auch aufgezeigt, inwieweit die Qualitätsmerkmale interagieren. Das heißt, inwieweit die Komponenten der Architektur einen (u.U. gegenläufigen) Einfluss auf zwei oder mehrere Qualitätsmerkmale haben können.

Die Schritte der ATAM

Die ATAM gliedert sich in vier Phasen (siehe Abbildung 3.1). Den Kern der ATAM bilden die Phasen eins und zwei. In diesen Phasen findet die Bewertung an sich statt. Sie bestehen aus den folgenden neun Schritten der ATAM, die in diesem Abschnitt näher betrachtet werden. In den Phasen null und drei werden Vor- und Nachbearbeitung der eigentlichen Bewertung betrachtet. Diese Phasen werden hier nicht betrachtet.

1. Vorstellung der ATAM (engl. *Present the ATAM*)
2. Vorstellung der Geschäftsziele (engl. *Present the business drivers*)
3. Vorstellung der Architektur (engl. *Present the architecture*)
4. Identifikation der Architekturansätze (engl. *Identify the architectural approaches*)
5. Erzeugung des Qualitätsmerkmal-Baums (engl. *Generate the quality attribute utility tree*)
6. Analyse der Architekturansätze (engl. *Analyze the architectural approaches*)
7. Brainstorming und Priorisierung von Szenarien (engl. *Brainstorm and prioritize scenarios*)
8. Analyse der Architekturansätze (engl. *Analyze the architectural approaches*)
9. Vorstellung der Resultate (engl. *Present the results*)

Die Schritte sind wiederum vier Gruppen zugeordnet. Schritte eins bis drei gehören der Gruppe Präsentation an, Schritte vier bis sechs der Gruppe Investigation und Analyse, Schritte sieben und acht der Gruppe Testen und Schritt neun der Gruppe Berichten. Im Folgenden wird kurz auf die einzelnen Schritte eingegangen.

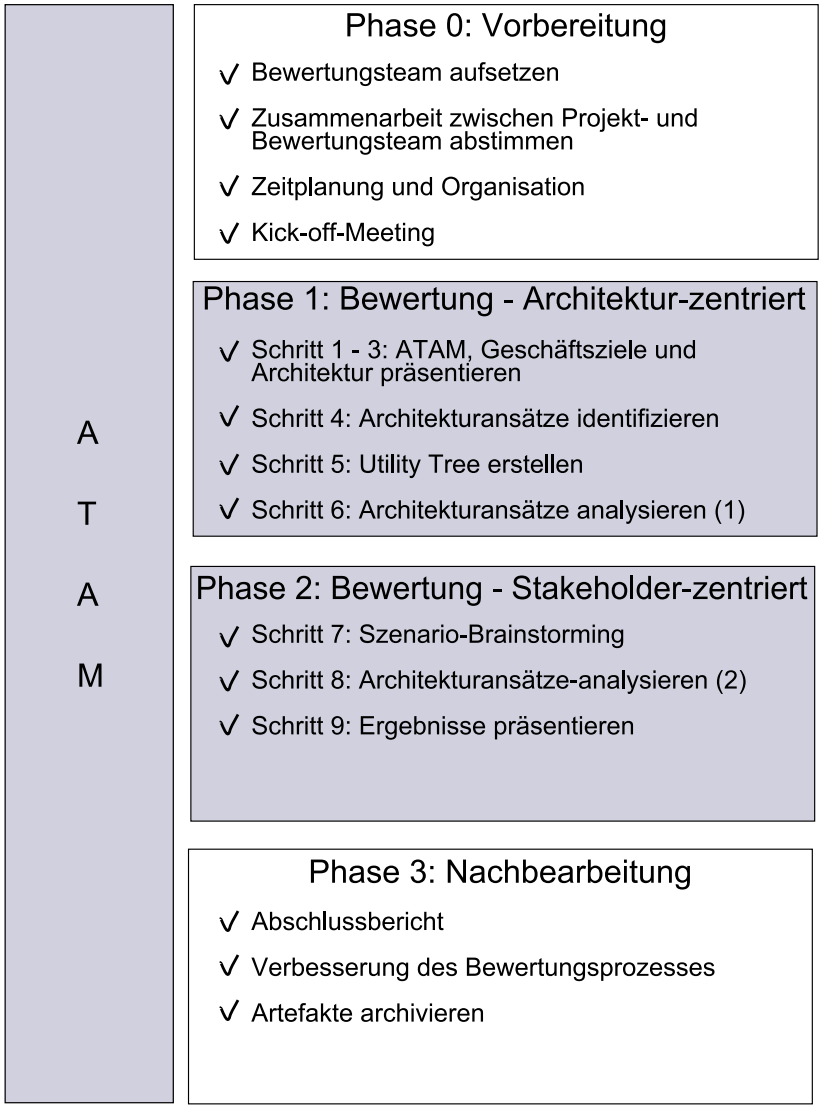

Abbildung 3.1.: Überblick über ATAM ([PBG07])

Schritt 1: Vorstellung der ATAM. Die ATAM ist eine Methode, bei der zum einen viele Stakeholder beteiligt sind und zum anderen ein hohes Maß an Interaktion zwischen Stakeholdern und an der Entwicklung beteiligten Personen erforderlich ist. Daher ist es notwendig, dass alle Beteiligten wissen, was sie erwartet und wie sie in die Methode einbezogen werden. Der erste Schritt der ATAM schreibt deshalb vor, dass allen Beteiligten zu Beginn einer ATAM erläutert wird, was ATAM ist und wie ATAM funktioniert.

Schritt 2: Vorstellung der Geschäftsziele. Im zweiten Schritt werden die wichtigsten Funktionen des zu evaluierenden Systems dargelegt. In diesem Schritt soll klar werden, zu welchem Zweck das System entwickelt wird oder wurde. Auch werden in diesem Schritt die wichtigsten nichtfunktionalen Anforderungen an das System dargelegt.

Schritt 3: Vorstellung der Architektur. Analog zu den ersten zwei Schritten erfolgt die Vorstellung der Architektur des Systems formlos durch den Architekten. Allerdings wird in [CKK02] darauf hingewiesen, dass unterschiedliche Sichten zur Beschreibung der Architektur verwendet werden sollten. Auch wird in [CKK02] beschrieben, welche Sichten typischerweise von Nutzen sein können. Weiterhin sollte der Architekt erläutern,

- wie die wichtigsten Qualitätsanforderungen durch die Architektur auf welche Weise adressiert werden,
- welche Rahmenbedingungen die Architektur erfüllt und
- mit welchen anderen Systemen das zu evaluierende System interagiert.

Das Team, das die Evaluation durchführt, versucht bereits in diesem Schritt, in Vorbereitung für den nächsten Schritt, Architekturansätze zu identifizieren und zu dokumentieren.

Schritt 4: Identifikation der Architekturansätze. Das Evaluationsteam versucht im vierten Schritt der ATAM die im zu evaluierenden System verwendeten Architekturansätze zu identifizieren und zu dokumentieren. Zum einen wird hierzu der Architekt des Systems befragt, zum anderen dokumentiert das Evaluationsteam Architekturansätze, die es durch persönliche Erfahrung identifizieren kann. Weiter gehende methodische Anleitung, wie die Ansätze identifiziert werden können, wird durch die ATAM nicht gegeben. In diesem Schritt wird noch keine Wertung vorgenommen. Die identifizierten Ansätze werden lediglich dokumentiert.

Schritt 5: Erzeugung des Qualitätsmerkmal-Baums. Um die Eignung der Architektur feststellen zu können, werden in diesem Schritt Anforderungen gesammelt, gegen die die Eignung der Architektur geprüft werden kann. Die Anforderungen werden in ATAM als Szenarien erhoben. Szenarien in ATAM sind mögliche Interaktionsabfolgen, die Stakeholder mit dem zu evaluierenden System durchführen und die daraus erwarteten Resultate. Eine genauere Erläuterung von Szenarien findet sich im weiteren Verlauf dieses Abschnitts auf Seite 44.

Die identifizierten Szenarien werden in einen Qualitätsmerkmal-Baum eingeordnet. Die Wurzel des Baumes heißt Nutzen. Die Knoten erster Ebene bezeichnen die Qualitätsmerkmale. Diese werden wiederum auf der nächsten Ebene in Teilmerkmale verfeinert. Die erhobenen Szenarien werden den Teilmerkmalen zugeordnet.

Die Szenarien werden von den Teilnehmern der ATAM nach Wichtigkeit priorisiert. Der Architekt priorisiert die Szenarien nach dem geschätzten Lösungsaufwand.

Schritt 6: Analyse der Architekturansätze. Der sechste Schritt stellt den Hauptteil der ATAM. In diesem werden zum einen die in Schritt vier dokumentierten Architekturansätze erweitert und verfeinert. Zum anderen werden die in Schritt fünf als Szenarien erhoben Anforderungen mit den Architekturansätzen in Verbindung gebracht.

Die Analyse beginnt mit den am höchsten priorisierten Szenarien aus Schritt fünf. Für jedes dieser Szenarien wird festgehalten, durch welche Architekturansätze sie beeinflusst werden. Anschließend stellt das Evaluationsteam einen Satz von Analysefragen (engl. *analysis questions*). Diese Fragen sind vom einzelnen Problem unabhängig. Sie gelten universell bei dem Einsatz eines bestimmten Architekturansatzes oder für ein bestimmtes Qualitätsmerkmal. In [KKC00, S. 10f] sind Beispiele für solche problemunabhängige Analysefragen aufgeführt. Einige dieser Fragen für das Qualitätsmerkmal Performanz werden im Folgenden aufgeführt:

- Welche Komponenten sind an dem Vorgang beteiligt?
- Wie sind die Ausführungszeiten dieser Komponenten?
- Befinden sich die Komponenten auf denselben oder auf unterschiedlichen Prozessoren?
- Was geschieht, wenn mehrere Anfragen sukzessiv in kurzer Zeit auftreten?

Durch die Diskussion über und die Beantwortung der Analysefragen gewinnt das Evaluationsteam ein besseres Verständnis der Architektur und identifiziert dabei auch Risiken, Nicht-Risiken (engl. *Non-Risks*), Sensitivity Points und Tradeoff Points. *Sensitivity Points* sind Eigenschaften von Bestandteilen der Architektur (diese können einzelne Komponenten oder mehrere Komponenten und deren Beziehungen untereinander sein), die einen bedeutenden Einfluss auf ein Qualitätsmerkmal haben. Ein Beispiel für ein Sensitivity Point (aus [KKC00, S. 22]) ist: „Die Latenzzeit, um eine wichtige Nachricht zu verarbeiten, ist abhängig von der Priorität des am niedrigsten priorisierten Prozesses, der an der Verarbeitung der Nachricht beteiligt ist.“ Führt der Wert eines Sensitivity Points

potentiell zu einer Verletzung einer Qualitätsanforderung, handelt es sich um ein *Risiko*. Von einem *Nicht-Risiko* ist die Rede, wenn der Wert eines Sensitivity Points kein Risiko ist. Hat ein Sensitivity Point einen Einfluss auf mehr als ein Qualitätsmerkmal, handelt es sich um ein *Tradeoff Point*.

Schritt 7: Brainstorming und Priorisierung von Szenarien. Mit den Schritten sieben und acht werden die Ergebnisse der Schritte fünf und sechs validiert und verfeinert. Deshalb sind Schritte sieben und acht leicht veränderte Reiterationen der Schritte fünf und sechs. Der Unterschied besteht darin, dass in Schritt sieben eine größere Menge an Stakeholdern an der Erhebung und Priorisierung der Szenarien beteiligt ist und eine größere Menge an Szenarien erhoben wird. Insbesondere werden die Stakeholder dazu aufgefordert, zusätzlich zu den bereits vorhandenen Szenarien speziell auch Wachstums- sowie Erkundungsszenarien zu entwickeln. Erkundungsszenarien beschreiben Anforderungen, von denen zu erwarten ist, dass sie durch die Architektur nicht erfüllt werden können. Durch die Nutzung dieser Szenarien bei der Analyse können die Schwachpunkte der vorliegenden Architektur besser identifiziert werden. Zusätzlich helfen Erkundunsszenarien in der späteren Analyse bei der Identifikation von weiteren Sensitivity Points.

Schritt 8: Analyse der Architekturansätze. Mit den neuen Szenarien aus Schritt sieben werden in Schritt acht die gleichen Aktivitäten wie in Schritt sechs durchgeführt.

Schritt 9: Vorstellung der Resultate. Der letzte Schritt schreibt vor, dass die Resultate aufbereitet und vorgestellt werden. Zusätzlich werden identifizierte Risiken zu Themen zusammengefasst. Beispielsweise kann festgestellt werden, dass es mehrere Risiken gibt, die darauf hinweisen, dass die zu evaluierende Architektur unzureichend auf Hardwareausfälle reagiert.

Die wichtigsten Konzepte der ATAM

Szenarien. Mit der Hilfe von Szenarien werden in ATAM, sowie in anderen Methoden des SEI, Qualitätseigenschaften eines Systems beschrieben. Die Grundannahme beim Konzept von Szenarien ist, dass ein Qualitätsmerkmal eines Systems immer als Verhältnis zwischen einer Aktion, die auf das System einwirkt, und einer Reaktion des Systems dargestellt werden kann. Ein Szenario, so wie es von den Forschern des SEI definiert wird, besteht aus *Quelle*, *Reiz*, *Umgebung*, *Artefakt*, *Antwort*, *Antwortmetrik* (engl.: *Source of stimulus*, *Stimulus*, *Environment*, *Artifact*, *Response*, *Response Measure*). Die *Quelle* ist eine Entität, die

auf das System einwirkt. Die Quelle kann ein Mensch, aber auch ein weiteres System oder ein beliebiges anderes Objekt, das auf das System wirken kann, sein. Der *Reiz* beschreibt, wie diese Quelle auf das System einwirkt. In der *Umgebung* werden die Rahmenbedingungen der Umwelt oder des Systems selbst beschrieben, die für die Reaktion des Systems von Bedeutung sind. Der Reiz kann auf das System als ganzes oder auf einen Teil des Systems wirken. Worauf der Reiz wirkt, wird als *Artefakt* bezeichnet. Die *Antwort* beschreibt die Reaktion des Systems auf den entsprechenden Reiz und die *Antwortmetrik* beschreibt wie die Antwort gemessen werden kann. Abbildung 3.2 zeigt ein Beispielszenario aus [BCK03].

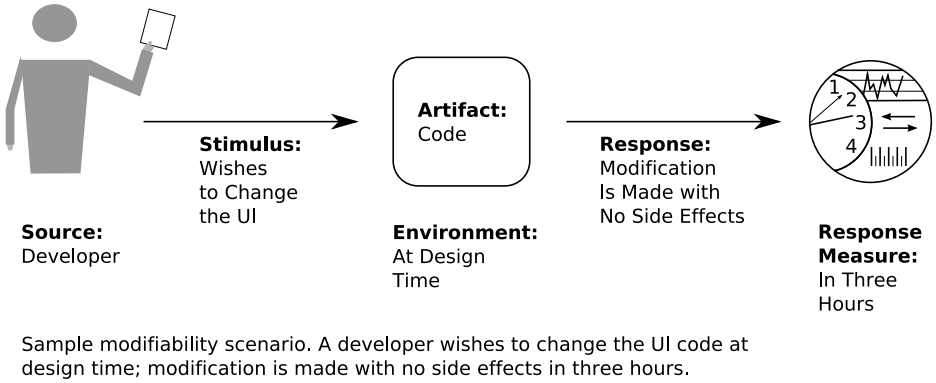

Abbildung 3.2.: Ein beispielhaftes Szenario für das Qualitätsmerkmal Wartbarkeit (aus [BCK03, S. 77])

In ATAM werden drei Arten von Szenarien für die Erhebung von Qualitätsanforderungen verwendet. In *Use-case-Szenarien* werden Anforderugen erhoben, die bei der normalen Nutzung des Systems auftreten. *Wachstumsszenarien* (engl. *Growth szenarios*) werden verwendet, um Anforderungen an zu erwartende Änderungen oder Erweiterungen festzuhalten. In *Erkundungsszenarien* (engl. *exploratory scenarios*) werden bewusst übertriebene Qualitätsanforderungen an das System gestellt, um die Grenzen der zu evaluierenden Architektur zu eruieren und die Schwachstellen der Architektur zu lokalisieren.

Attribute Based Architectural Styles und Analysis Questions. Um bessere Überlegungen über die Qualitätseigenschaften von Architekturstilen durchführen zu können, wurde am SEI das Konstrukt der *Attribute Based Architectural Styles (ABAS)* entwickelt [KK99, KKB+99]. Hierbei wird ein Architekturstil (im Sinne eines Architekturmusters nach dem Verständnis in diesem Dokument) hinsichtlich eines (Qualitäts-)Attributs betrachtet. Für diese Betrachtung wird ein geeignetes Qualitätsmodell aufgebaut und auf den Architekturstil angewendet.

Um beispielsweise aus dem Simplex Architekturmuster [SRG95] ein verfügbarkeitsbasiertes Simplex ABAS zu erstellen, müsste ein Verfügbarkeits-Qualitätsmodell erarbeitet werden und anhand dieses Qualitätsmodells erläutert werden, wie z.B. das Qualitätsmerkmal Verfügbarkeit durch den Einsatz und die verschiedenen Ausprägungen von Simplex beeinflusst würde. In [KKB+99] wird genau dieses Beispiel verwendet. Als Qualitätsmodell werden hier Markov-Ketten verwendet. Jeder Knoten der Markov-Kette repräsentiert einen Zustand des Systems. Jede Kante repräsentiert einen Übergang des Systems in einen anderen Zustand. Jede Kante wird mit der Wahrscheinlichkeit beschriftet, dass das System vom Zustand im ausgehenden Knoten der Kante zum Zustand im Zielknoten der Kante übergeht. Mit diesem Qualitätsmodell lassen sich Aussagen darüber treffen, wie sich Änderungen am System auf die Verfügbarkeit auswirken. Beispielsweise, welche Änderungen an der Verfügbarkeit des Gesamtsystems zu erwarten sind, wenn mehrere oder wenigere redundante Backups zur Verfügung stehen oder wenn die Verfügbarkeit des Leaders verändert wird.

Ein ATAM Evaluationsteam kann seine Kenntnisse über ein ABAS verwenden, um Auswirkungen auf die Qualitätsmerkmale einer zu evaluierenden Architektur vorherzusehen.

Sind die Erläuterungen des Architekten über die bestehende Architektur unzureichend, um Aussagen über Qualitätsmerkmale treffen zu können, kommen die *Analysis Questions* zum Einsatz. Dabei handelt es sich um Fragen, mit denen das Evaluierungsteam sich ein besseres Verständnis über die vorliegende Architektur verschafft. Die Fragen werden vom Evaluationsteam so gestellt, dass dadurch die Schwachstellen und die Sensitivity und Tradeoff Points der Architektur zum Vorschein kommen. Zusätzlich zur eigenen Erfahrung der Mitglieder des Evaluationsteams dienen diesbezüglich auch die ABAS sowie u.U. Checklisten.

Sensitivity Points, Risks, Nonrisks, Tradeoff Points. Eines der wichtigsten Ergebnisse einer ATAM-Analyse ist eine Liste der Architekturbestandteile, die einen bedeutenden Einfluss auf Qualitätsmerkmale haben. Die identifizierten Architekturbestandteile werden, je nachdem ob diese einen Einfluss auf ein einzelnes oder auf mehr als ein Qualitätsmerkmal haben, in *Sensitivity Points* und *Tradeoff Points* kategorisiert und als *Risiken* oder *Nicht-Risiken* klassifiziert. Entscheidend für die Klassifikation als Risiko oder Nicht-Risiko ist, ob ein als Sensitivity oder Tradeoff Point identifizierter Architekturbestandteil potentiell zu einer Verletzung einer Qualitätsanforderung führt.

Weiterführende Literatur

Die Übersicht über ATAM, die in der vorliegenden Arbeit gegeben wird, beschränkt sich auf die wesentlichen Schritte. Viele Angaben zu ATAM bezie-

hen sich auf die „soft-skills“ des Software Engineerings (z.B. Angaben zu den Phasen, in denen ATAM durchzuführen ist, Angaben zu Vertragsfragen oder Angaben zur Einbeziehung des Managements). Für detailliertere Erläuterungen diesbezüglich sei auf die ATAM-Literatur [KKB+98, KKC00, CKK02, BCK03] verwiesen.

Zusätzliche Informationen sind auch in den zahlreichen Publikationen über Erweiterungen, Anpassungen und Fallstudien zu ATAM zu finden (z.B.:[BFJK99, KBK+99, BW99, WB99, LC05]).

3.3. Weitere Methoden

Zusätzlich zur bekanntesten Methode zur Evaluation von Softwarearchitekturen – der ATAM – gibt es noch weitere verbreitete Methoden. Vorläufer der ATAM war die SAAM, die ursprünglich für den Vergleich mehrerer alternativer Architekturen entwickelt wurde und ihren Schwerpunkt auf der Evaluation der Wartbarkeit hat. Schwerpunkt der CBAM ist die Evaluation der Kosten, die mit Architekturentscheidungen in Zusammenhang gebracht werden können. Für die Evaluation von ersten Überlegungen zu Architekturen wurde ARID entwickelt. Eine Methode, die je nach Zweck der Evaluation unterschiedliche Ausprägungen hat, ist ALMA. Die genannten Methoden werden im Folgenden vorgestellt. Abschließend wird auf den SARA Report eingegangen, in welchem die Erfahrungen aus Evaluationen aus der Industrie zusammengetragen wurden.

3.3.1. SAAM

In ihrer Ursprünglichen Fassung war die SAAM (**S**oftware **A**rchitecture **A**nalysis **M**ethod) für den Vergleich von alternativen Systemen gedacht ([KBWA94]). Inzwischen hat die Methode eine Entwicklung hinter sich und ist auch in der Bewertung von Softwarearchitekturen einzelner Systeme von Nutzen. Der Kern der Methode, Szenarien auf die Komponenten der Architektur abzubilden ist jedoch immer gleich geblieben.

Die Schritte der SAAM

Abbildung 3.3 zeigt die sechs Schritte der SAAM aus [CKK02]. Die Pfeile in der Abbildung zeigen Abhängigkeiten zwischen den einzelnen Schritten. Das heißt, dass die Ergebnisse des Schrittes, an dem der Pfeil beginnt, im Schritt, zu dem der Pfeil zeigt, verwendet werden. Nachfolgend wird auf die einzelnen Schritte näher eingegangen.

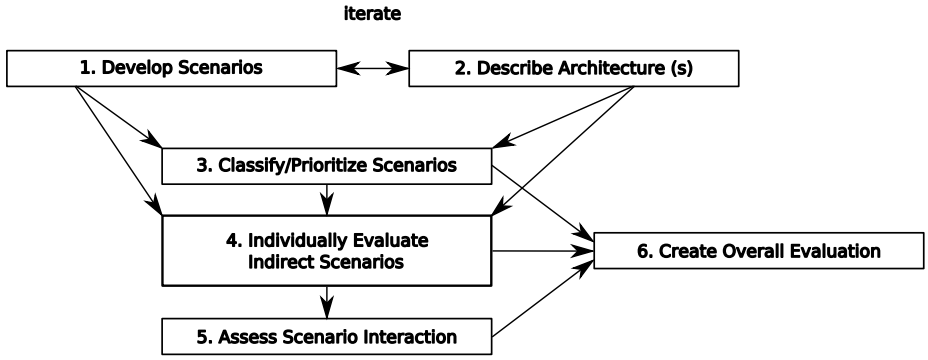

Abbildung 3.3.: Aktivitäten mit Abhängigkeiten der SAAM (aus [CKK02])

1. Erzeugen von Szenarien (Develop Scenarios)
2. Beschreiben der Architektur (Describe Architecture(s))
3. Klassifizieren und Priorisieren der Szenarien (Classify/Prioritize Scenarios)
4. Evaluieren der indirekten Szenarien (Individually Evaluate Indirect Scenarios)
5. Erheben der Szenario-Interaktion (Assess Scenario Interaction)
6. Erzeugen der Gesamtevaluation (Create Overall Evaluation)

Schritt 1 und 2: Erzeugen von Szenarien und Beschreiben der Architektur. Die ersten beiden Schritte der SAAM erfolgen entweder gemeinsam oder über mehrere Iterationen. Die Szenarien, welche von den beteiligten Stakeholdern in einem Brainstorming erhoben werden, können Lücken in der Architekturbeschreibung aufdecken. Analog können bei der Beschreibung der Architektur weitere Szenarien sichtbar werden. Die Szenarien und die Architektur des Systems werden in den ersten beiden Schritten dokumentiert.

Schritt 3: Klassifizieren und Priorisieren der Szenarien. Im dritten Schritt der SAAM werden die Szenarien in direkte und indirekte Szenarien klassifiziert. Als direkte Szenarien werden solche Szenarien bezeichnet, die durch die vorliegende Architektur bereits unterstützt werden. Das heißt, dass die vorliegende Architektur die Anforderung, die durch das Szenario repräsentiert wird, bereits erfüllen würde. Dazu wird das Szenario in Gedanken simuliert. Es wird in einem Gedankenexperiment verfolgt, wie die Architektur auf den durch das Szenario definierten Reiz reagieren würde, welche Komponenten auf welche Weise interagieren würden und welche Antwortmetrik zu erwarten wäre.

Analog werden indirekte Szenarien klassifiziert. Als indirekte Szenarien werden solche Szenarien bezeichnet, die durch die vorliegende Architektur noch nicht unterstützt werden. Das heißt, dass die vorliegende Architektur verändert werden müsste, damit die durch das Szenario repräsentierte Anforderung erfüllt werden könnte.

In einem Abstimmungsverfahren werden die Szenarien von den Stakeholdern priorisiert. Die Priorisierung dient dazu, im weiteren Verlauf der Evaluation nur Szenarien zu betrachten, die den Stakeholdern auch wichtig sind.

Schritt 4: Evaluieren der indirekten Szenarien. Bei der Evaluation der indirekten Szenarien wird dokumentiert, welche Änderungen an der vorliegenden Architektur durchzuführen sind, um die Unterstützung der Szenarien durch die Architektur zu ermöglichen. Zusätzlich werden die Kosten geschätzt, die bei der Umsetzung dieser Änderungen zu erwarten wären.

Schritt 5: Erheben der Szenario-Interaktion. Laut der SAAM liegt eine Szenario-Interaktion dann vor, wenn zwei oder mehr indirekte Szenarien eine Änderung an derselben Komponente der Architektur benötigen, um von der Architektur unterstützt zu werden. Im fünften Schritt der SAAM werden für die in Schritt 3 ausgewählten Szenarien alle Szenario-Interaktionen dokumentiert. Dazu werden die Informationen aus Schritt 4 verwendet.

Schritt 6: Erzeugen der Gesamtevaluation. Aufgrund der Wichtigkeit der Szenarien (durch die Stakeholder festgelegt) und der Anzahl direkter und indirekter Szenarien sowie notwendigen Anpassungen, um ein indirektes Szenario zu unterstützen, kann im letzten Schritt die Gesamtevaluation erzeugt werden.

Die wichtigsten Konzepte der SAAM

Die Klassifikation in direkte und indirekte Szenarien dient dem Vergleich von alternativen Architekturen. Dadurch können die Alternativen direkt miteinander verglichen werden.

Durch das Abbilden von Szenarien auf die Komponenten der Architektur (Schritte vier und fünf) wird deutlich, welche Komponenten der Architektur welche Funktionalität erbringen. Dadurch werden wiederum „ungünstige“ Zerlegungen ersichtlich, bei denen beispielsweise mehrere Funktionen von einer Komponente erbracht werden, die in mehreren Komponenten besser aufgeteilt wären. Die Interaktion mehrerer Szenarien in einer Komponente kann aber auch bedeuten,

dass die Dokumentation der Architektur nicht ausreichend detailliert vorliegt, da die Komponente evtl. in weitere Komponenten zergliedert werden kann, so dass keine Interaktion zwischen den Szenarien vorliegt.

3.3.2. CBAM

Die **C**ost **B**enefit **A**nalysis **M**ethod wurde nach ihrer Vorstellung auf der ICSE 2001 [KAK01] und einem im selben Jahr folgenden technischen Bericht [AKK01] im Jahr 2003 in einer weiterentwickelten Fassung in [BCK03, Kapitel 12] veröffentlicht.

Bei der CBAM werden die Kosten und der Nutzen von Alternativen bei der Gestaltung der Architektur eines zu erstellenden oder zu verändernden Systems gegenübergestellt. Der Grundgedanke der CBAM basiert darauf, dass Architekturentscheidungen – Entscheidungen über alternative Architekturen für ein zu erstellendes oder zu veränderndes System – sowohl technische als auch wirtschaftliche Konsequenzen haben. Technische Konsequenzen insofern, als dass die Architekturentscheidungen die Qualitätsmerkmale des Systems beeinflussen. Wirtschaftliche Konsequenzen zum einen, über die Kosten, die die Umsetzung einer Architekturentscheidung mit sich bringt, zum anderen, über die Ausprägung der Qualitätsmerkmale.

Wesentliche Schritte der CBAM

In der aktuellsten Fassung besteht die CBAM aus neun Schritten. Da diese jedoch einige Details beinhalten, die für die vorliegende Arbeit nicht relevant sind, werden hier nur die Schritte von zentraler Bedeutung kurz erläutert:

- Szenarien und Architekturstrategien wählen
- Nutzen der Szenarien abschätzen
- Quantifizieren der Nutzen der Architekturstrategien
- Quantifizieren der Kosten der Architekturstrategien
- Rentabilität berechnen

Szenarien und Architekturstrategien wählen. Die in der ATAM-Analyse erarbeiteten Szenarien werden in CBAM in mehreren Iterationen[1] neu priorisiert. Für die CBAM sind nur die Szenarien von Bedeutung, für die eine Änderung der

[1] Genauere Angaben zu den Verfahren finden sich unter [BCK03, Kapitel 12]

Architektur des betrachteten Systems erwünscht ist. Das heißt, ein oder mehrere Qualitätsmerkmale, die mit diesem Szenario verbunden sind, werden durch den Ist-Stand der Architektur nicht zufrieden stellend berücksichtigt.

Für diese Szenarien stellt der Architekt mögliche Änderungen an der Architektur vor, die zu einer besseren Berücksichtigung des entsprechenden Qualitätsmerkmals führen würde. In CBAM werden diese Vorschläge als Architekturstrategie (AS) bezeichnet. Für die Verbesserung der Qualitätsmerkmale eines Szenarios können ein oder mehrere Architekturstrategien vorgeschlagen werden.

Nutzen der Szenarien abschätzen. Den priorisierten Szenarien wird ein Nutzen W (weight) zugeordnet. Das am höchsten priorisierte Szenario erhält den Nutzen $W = 1$, alle weiteren Szenarien werden bezüglich dieses Szenarios mit Werten zwischen 0 und 1 bewertet. Die Bewertung erfolgt durch ein Abstimmungsverfahren.

Quantifizieren der Nutzen der Architekturstrategien. Der Nutzen B_i (benefit), der einer Architekturstrategie i zugewiesen wird, berechnet sich über die Summe der Nutzen, den die jeweilige Architekturstrategie über ein Szenario j erzielt. Dabei wird bei der Summe die Gewichtung der Szenarien berücksichtigt. Somit ergibt sich für den Nutzen einer Architekturstrategie folgende Formel:

$$B_i = \sum_j (b_{i,j} \cdot W_j)$$

Wobei $b_{i,j}$ den Einfluss auf den Nutzen der Architekturstrategie i bezogen auf das Szenario j bezeichnet. Da sich die Antwortmetrik eines Szenarios durch die Umsetzung einer Architekturstrategie auch verschlechtern kann, ist es möglich, dass der Wert von $b_{i,j}$ negativ ist.

Quantifizieren der Kosten der Architekturstrategien. Im Anschluss werden für jedes Architekturszenario i die entsprechenden Kosten für dessen Umsetzung C_i geschätzt. Die Entwickler der CBAM beziehen sich in diesem Schritt auf die Arbeiten von Boehm [Boe81] und Jones [Jon98].

Rentabilität berechnen. Die Rentabilität einer Architekturstrategie i berechnet sich aus dem Kosten-Nutzen-Verhältnis:

$$R_i = \frac{B_i}{C_i}$$

Nach diesem Verhältnis können die Architekturstrategien anschließend sortiert und für die Umsetzung eingeplant werden. Allerdings müssen weitere Faktoren berücksichtigt werden, auf die in der CBAM hingewiesen wird. So muss beispielsweise auf Abhängigkeiten zwischen den Architekturstrategien geachtet werden, da die Umsetzung einer Architekturstrategie den Nutzen einer anderen zunichte machen oder aber die Kosten für die Umsetzung einer anderen Architekturstrategie verringern könnte. Des Weiteren sollten nicht nur die Kosten zur Umsetzung einer Architekturstrategie, sondern auch deren Einfluss auf die Planung in Betracht gezogen werden.

Die wichtigsten Konzepte der CBAM

Genau wie die ATAM und die SAAM basiert die CBAM auf dem Konzept der Szenarien und der Interaktion der Stakeholder bei der Bewertung dieser Szenarien. Zusätzlich sehen die Entwickler der CBAM in der Schätzung der Nutzen einer Architekturstrategie sowie in der expliziten Modellierung von Unsicherheiten bei Schätzungen einen Mehrwert der CBAM.

Schätzungen von Kosten und Nutzen. In [KAK01] weisen Kazman et al. darauf hin, dass sich die Modellierung ökonomischer Aspekte im Software Engineering in der Vergangenheit primär auf die Modellierung von Kosten bezogen hat. In der CBAM wird deshalb auch der Nutzen einer Architekturstrategie modelliert und den jeweiligen Kosten gegenüber gestellt. Die Modellierung des Nutzen einer Architekturstrategie geschieht über die geschätzte Änderung in der Antwortmetrik eines Szenarios. Dazu werden zu jedem Szenario die best-case und worst-case Antwortverhalten erhoben. Diesen Antwortverhalten werden die Werte 100 und 0 zugewiesen. Anschließend werden die Antwortverhalten des Ist-Stands und des Soll-Stands mit entsprechenden Werten belegt. Über diese Referenzwerte kann der Nutzen einer Architekturstrategie durch die Differenz des Werts des erwarteten Antwortverhaltens zum Wert des Antwortverhaltens des Ist-Stands ausgedrückt werden.

Unsicherheit bei Schätzungen einbeziehen. Da die Werte für die Kosten und die Nutzen der Umsetzung der Architekturstrategien auf Schätzungen basieren, werden die dadurch entstehenden Unsicherheiten bei CBAM dokumentiert und bei der Entscheidung zur Umsetzung von Architekturstrategien berücksichtigt. Abbildung 3.4 zeigt die Kosten und Nutzenschätzwerte verschiedener Architekturstrategien aus einer Fallstudie der CBAM [KAK01]. Die Unsicherheiten bei den Schätzungen werden, abhängig davon wie unterschiedlich die beteiligten

Stakeholder die Kosten und Nutzen eingeschätzt haben, entsprechend durch die Breite und Höhe der Ellipsen dargestellt.

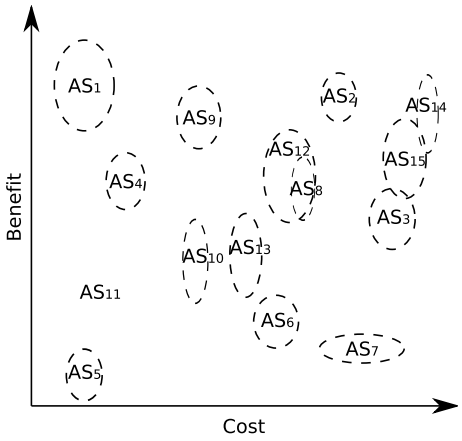

Abbildung 3.4.: Kosten und Nutzen von Architekturstrategien (angelehnt an [KAK01, S. 301])

3.3.3. ARID

Zentraler Fokus von ARID (**A**ctive **R**eviews for **I**ntermediate **D**esigns) [Cle00, CKK02] ist die Evaluierung von Architekturteilen bzw. von noch nicht fertig gestellten Architekturen. Dies hat den Vorteil, dass bereits frühzeitig mit der Qualitätssicherung begonnen werden kann, auch wenn die gesamte Architektur des Systems noch nicht fertig gestellt wurde.

Die Methode basiert auf den von Parnas und Weiss vorgestellten **A**ctive **D**esign **R**eviews (ADRs) [PW85] und erweitert diese um das Konzept der Szenarien. Bei ADRs werden die Reviewer dazu gebracht, sich mit dem zu beurteilenden Material auseinanderzusetzen, indem die Reviewer Fragen beantworten müssen, die so formuliert werden, dass eine Antwort nur möglich ist, wenn der Reviewer sich mit dem Material auseinander setzt.

Die Szenarien werden in ARID analog zur ATAM und zur SAAM verwendet, um die Anforderungen an das zu evaluierende Teilsystem festzulegen. In diesem Fall beschreiben die Szenarien hauptsächlich die Interaktion des zu evaluierenden Teilsystems mit dem restlichen System (der Umgebung des Teilsystems).

Analog zu den anderen Methoden des SEI werden auch bei ARID die Szenarien über Abstimmungsprozesse priorisiert und es werden mehrere Schritte zur Vor- und Nachbereitung durchgeführt. Auf diese wird in diesem Abschnitt nicht mehr eingegangen. Kern der Methode ist, dass die Reviewer dazu aufgefordert werden,

für die erarbeiteten Szenarien Implementierungen erarbeiten, die die Szenarien mit Hilfe der zu evaluierenden Teilsysteme durchführen. Die Implementierungen müssen nicht ausführbar sein. Ziel ist, dass der Architekt des Teilsystems erkennt, wie sich andere Entwickler die Interaktion mit dem zu evaluierenden Teilsystem vorstellen.

3.3.4. ALMA

Die **A**rchitecture-**L**evel **M**odifiability **A**nalysis [BLBvV00, LBvVB02, BLBvV04] ist eine Weiterentwicklung der SAAM, bei der die Änderbarkeit eines Systems evaluiert wird. Die ALMA unterscheidet sich von der SAAM u.a. in der Bestimmung eines Ziels für die Analyse. Die Analyse kann dadurch zielgerichteter durchgeführt werden. Für die Analyse der Änderbarkeit definiert die ALMA drei mögliche Ziele: Risikoabschätzung, Wartungskostenvorhersage und Softwarearchitekturvergleich. Je nach Ziel der Analyse werden unterschiedliche Techniken bei der Durchführung der Analyse verwendet. So werden z.B. für das Ziel der Risikoabschätzung Änderungsszenarien gesucht, die bei der zu analysierenden Architektur nur schwer durchzuführen sind, während bei dem Ziel des Softwarearchitekturvergleichs solche Änderungsszenarien gesucht werden, die in den beiden zu vergleichenden Architekturen unterschiedliche Folgen nach sich ziehen, wohingegen bei dem Ziel der Wartungskostenvorhersage nach Änderungsszenarien, die mit hoher Wahrscheinlichkeit im betrachteten Zeitraum eintreten, gesucht wird.

3.3.5. SARA

Der SARA Report [OKK+02] (**S**oftware **A**rchitecture **R**eview and **A**ssessment Report) ist keine eigene Methode zur Architekturbewertung. Stattdessen werden im SARA Report die Erfahrungen und Prinzipien von Experten aus der Industrie zusammengetragen, wie bei Architekturbewertungen vorzugehen ist. Es werden methodische Anleitungen gegeben, Techniken gegenüber gestellt und Vorlagen zur Dokumentation zur Verfügung gestellt. Einige für die vorliegende Arbeit interessante Aspekte des SARA Reports werden im Folgenden hervorgehoben.

Im SARA Report wird der Zweck einer Architekturbewertung wie folgt verstanden:

> *The purpose of an architecture review is to understand the impact of every architecturally significant decision (ASD) on every architecturally significant requirement (ASR) [OKK+02, S.11].*

Die zwei Begriffe, die hier explizit und in anderen Methoden implizit Verwendung finden, sind „Architecturally Significant Requirements (ASR)“ und „Architecturally Significant Decisions (ASD)“. Diese Begriffe wurden im SARA Report aus [JRvdL00] übernommen. Die explizite Nennung und Nutzung dieser Begriffe schafft in der Architekturevaluation einen Mehrwert. Deshalb werden die Begriffe auch für die vorliegende Arbeit übernommen. Sie werden nachfolgend definiert.

> **Definition 17 (Architekturrelevante Entscheidung)**
> *Als architekturrelevante Entscheidung wird die bewusste Entscheidung für die Gestaltung eines Anteils einer Architektur bezeichnet, für den eine alternative Gestaltung der Architektur möglich wäre.*

Aufbauend auf der Definition der architekturrelevanten Entscheidung lässt sich der Begriff der architekturrelevanten Anforderung definieren:

> **Definition 18 (Architekturrelevante Anforderung)**
> *Eine Anforderungen wird als architekturrelevant bezeichnet, wenn die Umsetzung der Anforderung es erfordert, dass diese Anforderung beim Entwurf der Architektur zu berücksichtigen ist. Das heißt, dass die Umsetzung der Anforderung einen Einfluss auf eine (oder mehrere) architekturrelevante Entscheidung(en) hat.*

Jazayeri et al. nennen Anhaltspunkte, wann eine Anforderung als architekturrelevant gilt (vgl. [JRvdL00, S.11]):

- Anforderungen, die nicht durch eine (oder eine kleine Menge von) Systemkomponente(n) ohne Abhängigkeit zum restlichen System umgesetzt werden können.
- Anforderungen, die Eigenschaften aus unterschiedlichen Kategorien von Komponenten adressieren.
- Anforderungen, die die Manipulation mehrerer Komponenten nach sich ziehen.

Analog zu den meisten qualitativen Verfahren der Architekturbewertung wird auch im SARA Report davon ausgegangen, dass die architekturrelevanten Anforderungen sowie die Architektur und die architekturrelevanten Entscheidungen zu Beginn der Architekturevaluation nicht explizit dokumentiert vorliegen und während des Evaluationsprozesses erhoben werden müssen.

3.4. Die Methoden im Vergleich

In einigen Publikationen wurden die hier vorgestellten Methoden zur Architekturbewertung gegenüber gestellt. Eine entsprechende Gegenüberstellung in Anlehnung an eine dieser Publikationen geschieht in Abschnitt 3.4.1.

Da die im vorliegenden Dokument erarbeitete Methode vor allem relativ zur Methode ATAM positioniert wird, werden in 3.4.2 die Vor- und Nachteile von ATAM hervorgehoben.

3.4.1. Klassifikationen in der Literatur

In [RH06, DN02, EHM07, BZJ04, CKK02] werden die Methoden zur Architekturbewertung anhand diverser Kriterien klassifiziert und untereinander verglichen. Da *Eicker et al.* die Klassifikationen von *Dobrica und Niemelä*, *Babar et al.* und *Clements et al.* in ihrer Arbeit bereits berücksichtigt haben und ihre Klassifikation dadurch von den hier genannten Arbeiten die umfassendste und sorgfältigste ist, wird hier auf der Klassifikation von *Eicker et al.* aufgebaut. In der Klassifikation von *Eicker et al.* werden Methoden in den Vergleich einbezogen, die im vorliegenden Dokument nicht erläutert werden bzw. für das vorliegende Dokument nicht relevant sind. Diese Methoden werden aus dem Vergleich entnommen. Die Methode ALMA wird im Vergleich von *Eicker et al.* nicht aufgeführt, da es sich bei dieser um eine Spezialisierung von SAAM handelt.

Für die Gegenüberstellung der Methoden wurden von *Eicker et al.* die folgenden Kriterien hinzugezogen (für detailliertere Erläuterungen sei auf [EHM07] verwiesen):

Qualitätsmerkmal. Das Qualitätsmerkmal, welches durch den Einsatz der Methode evaluiert werden soll.

Berücksichtigung von Beziehungen. Die Methoden werden bei diesem Kriterium danach unterschieden, ob die jeweilige Methode die wechselseitigen Beziehungen zwischen den Qualitätsmerkmalen (genauer zwischen den Architekturmechanismen, die zur Erlangung eines Qualitätsmerkmals eingesetzt wurden) berücksichtigt, oder ob die Qualitätsmerkmale unabhängig voneinander bewertet werden.

Voraussetzungen für die Anwendung. Bei diesem Kriterium wird danach unterschieden, welche Vorbedingungen gegeben sein müssen, damit die Methode angewendet werden kann.

Reifegrad / Validierung. Bei diesem Kriterium werden die Methoden danach verglichen, in wie vielen Veröffentlichungen über den Einsatz der Methode berichtet wird, wann die Methode erstmalig vorgestellt wurde und wie plausibel die Wirksamkeit der Methode ist.

Detaillierungsgrad der Prozessbeschreibung. Anhand dieses Kriteriums werden die Methoden danach unterschieden, wie genau die beim Einsatz der Methode durchzuführenden Aktivitäten beschrieben sind.

Ziel der Methode. Die zu vergleichenden Methoden unterscheiden sich in der Art der Aussage über die Qualität der zu evaluierenden Architektur. Bei diesem Kriterium werden die Methoden nach der Art der Aussage unterschieden, die über die Qualität der zu evaluierenden Architektur getroffen werden soll.

Projektphase. Die Methoden zur Evaluation von Architekturen unterscheiden sich nach dem Zeitpunkt des Einsatzes innerhalb der Projektphase.

Bewertungsansatz. Bei diesem Kriterium werden die Methoden danach unterschieden, wie – mit welchen Techniken – eine Aussage über die Qualität der zu evaluierenden Architektur herbeigeführt wird.

Beteiligung Stakeholder. Die Methoden unterscheiden sich darin, ob alle Stakeholder am Evaluationsprozess beteiligt sind.

Erfahrung und Kenntnisse des Evaluationsteams. Je nach Evaluationsmethode ist mehr oder weniger Erfahrung der Mitglieder des Evaluationsteams nötig, um eine aussagekräftige Evaluation zu gewährleisten.

Die Klassifikation von *Eicker et al.* wird (in reduzierter Form) in Tablle 3.1 wieder gegeben.

In Abschnitt 4.6 wird die in diesem Dokument erarbeitete Methode POSAAM in die Klassifikation von *Eicker et al.* eingeordnet.

3.4.2. Stellungnahme zu ATAM

Die im vorliegenden Dokument erarbeitete Methode positioniert sich in Bezug zur Methode ATAM. Deshalb werden im Folgenden die Schritte von ATAM

Tabelle 3.1.: Klassifikation von Methoden zur Architekturbewertung (reduzierte Fassung von [EHM07])

	SAAM + Spez.	**ATAM**	**ARID**	**CBAM**
Qualitätsmerkmale	Modifizierbarkeit	Verschiedene	Entwurfstauglichkeit	Kosten und Nutzen
Berücksichtigung von Beziehungen	Nein	Ja	Nein	Nein
Voraussetzungen für die Anwendung	Keine besonderen Anforderungen, leicht anzuwendende Methode	Durch die Analyse- und Befragungsphase hoher Ressourcenbedarf	Keine besonderen Anforderungen	Teilweise Durchführung einer ATAM Analyse
Reifegrad / Validierung	Ausgereift, erstmals 1996 beschrieben, Anwendung in vielen Bereichen	Sehr ausgereift, mittlerweile in 2. Version, Fallstudien und Trainings, validiert in vielen Projekten	Keine Angaben, verwendet jedoch validierte Techniken in Kombination (ADR + ATAM)	Ausgereift, umfangreiche Weiterentwicklung.
Detaillierungsgrad der Prozessbeschreibung	Ausführlich, inklusive Fallstudie	Ausführlich, inklusive Fallstudie	Ausführlich, inklusive Fallstudie	Ausführlich, inklusive Fallstudie
Ziele der Methode	Tauglichkeit (Suitability), Risiko	Sensitivity, Tradeoff, Risiko	Entwurfstauglichkeit	Bestimmung der wirtschaftlichen Auswirkungen der Entwurfsentscheidungen, ROI
Projektphase	Früh	Früh	Früh, Entwurf der Komponenten	Früh
Bewertungsansatz	Szenarios	Utility-Baum, Szenarios	Szenarios + ADR	Wert des Szenarios (Utility), Kostenmodell, ROI
Beteiligung Stakeholder	Ja	Ja	Ja	Ja
Erfahrung und Kenntnisse des Evaluationsteams	Leicht anzuwenden, kaum Erfahrung nötig	Hohe Anforderung an das Evaluationsteam wegen Aufstellen des Utility-Baums und der Identifizierung der Architekturansätze	Hohe Anforderungen an das Evaluationsteam, Erfahrung zur Bewertung der Ansätze nötig	Vergleichsweise hoch durch Bestimmung der Utilities

bezüglich deren methodischer Anleitung bezüglich Architekturbewertung betrachtet.

Die Schritte eins bis drei von ATAM (*Vorstellung der ATAM*, *Vorstellung der Geschäftsziele* und *Vorstellung der Architektur*) sind Präsentationen des Ist-Stands. Sie enthalten keinerlei methodische Anleitung bezüglich Architekturbewertung. Es werden lediglich Richtlinien gegeben, welche Inhalte auf welche Weise präsentiert und welche Stakeholder an den Präsentationen teilnehmen sollen.

Analog verhält es sich bei Schritt neun (*Vorstellung der Resultate*).

In Schritt vier (*Identifikation der Architekturansätze*) werden die verwendeten Architekturansätze identifiziert und dokumentiert. Dazu wird entweder der Architekt befragt oder die Erfahrung des Evaluationsteams genutzt, das die Architekturansätze anhand der Architekturbeschreibung aus Schritt drei intuitiv identifizieren kann. Weitere methodische Anleitung zur Vorgehensweise wird nicht gegeben.

Schritt acht (*Analyse der Architekturansätze*) ist eine Reiteration von Schritt sechs (*Analyse der Architekturansätze*). Der einzige Unterschied zwischen den Schritten besteht darin, dass in Schritt acht neue Szenarien in die Betrachtung einfließen. Diese neuen Szenarien werden in Schritt sieben (*Brainstorming und Priorisierung von Szenarien*) gewonnen. Es handelt sich bei Schritt sieben also lediglich um eine Verfeinerung von Schritt fünf (*Erzeugung des Qualitätsmerkmal-Baums*).

Die methodische Anleitung von ATAM beschränkt sich also auf die Schritte fünf und sechs.

In Schritt fünf (*Erzeugung des Qualitätsmerkmal-Baums*) werden Anforderungen erhoben, die für den Rest der ATAM-Evaluierung verwendet werden. In der ATAM-Literatur werden Anweisungen gegeben, welche Arten von Szenarien (Use-case-Szenarien, Wachstumsszenarien und Erkundungsszenarien) wie zu erheben und zu priorisieren sind. Die Szenarien werden bezüglich ihrer Wichtigkeit und bezüglich des geschätzten Realisierungsaufwands priorisiert. Die methodische Anleitung in diesem Schritt ist stark auf die Anforderungserhebung ausgerichtet. Lediglich die Differenzierung der Szenarien dient im weiteren Verlauf von ATAM dem Aufdecken von Schwachstellen in der Architektur.

Schritt sechs (*Analyse der Architekturansätze*) stellt den Kern der Architekturbewertung nach ATAM dar. In diesem Schritt werden die Architekturansätze zu den Anforderungen in Beziehung gebracht. Dazu ist primär die Erfahrung des Evaluationsteams gefragt. Zusätzlich werden Analysefragen verwendet, um Gedankenexperimente zu motivieren, durch die die Experten des Evaluationsteams ein besseres Verständnis der Details der Architektur gewinnen und Risiken, Nicht-Risiken, Sensitivity und Tradeoff Points identifizieren können. Die methodische Anleitung zur Gewinnung der Analysefragen beschränkt sich darauf, die in einer Durchführung einer ATAM verwendeten Analysefragen für weitere ATAM-Evaluationen zu dokumentieren. Weitere Analysefragen können laut ATAM aus Expertenwissen oder fachlicher Literatur gewonnen werden.

Zusammenfassend ist eine Konzentration auf Themen des Requirements und Social Engineerings in ATAM erkennbar. Reduziert man die Betrachtung von

ATAM auf die Schritte, in denen der Schwerpunkt auf der Methodik der Architekturbewertung gelegt wird, erhält man lediglich Schritt sechs. In diesem Schritt ist methodische Anleitung zur Architekturbewertung vorhanden. Viel ist jedoch vom Expertenwissen des Evaluationsteams abhängig.

3.5. Weitere verwandte Arbeiten

Grundlage für die neu vorgeschlagene Methode POSAAM ist das strukturierte Ablegen von Mustern und Musterrelationen in geeigneter Repräsentation sowie die explizite Repräsentation von in den Mustern implizit gespeicherten Informationen. In diesem Zusammenhang existieren bereits Arbeiten. Insbesondere die Arbeit über ein Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen von *Malich*.

Des Weiteren haben *Erfanian und Aliee* eine Ontologie zur Unterstützung der Evaluation von Softwarearchitekturen geschaffen. Eine ähnliche Ontologie wird auch in der vorliegenden Arbeit definiert.

Die beiden Arbeiten werden in diesem Abschnitt kurz vorgestellt, um im Kapitel, welches POSAAM vorstellt, auf diese Arbeiten Bezug nehmen und davon abgrenzen zu können.

3.5.1. Wissensmodell von Malich

Wie auch in der vorliegenden Arbeit, wird in [Mal08] das in Mustern gekapselte Expertenwissen für den Entwurf und die Bewertung von Softwarearchitekturen verwendet. Kern der Arbeit von *Malich* ist die Zuordnung von Mustern zu Qualitätsmerkmalen in Form einer Matrix, wie in Abbildung 3.5 dargestellt.

In dieser Matrix werden Informationen über die Muster selbst (in den Vorspalten) und über deren Zusammenhang zu Qualitätsmerkmalen (in den Zellen) gespeichert. Die Form, in welcher Einträge der Matrix vorgenommen werden können, ist in einer in Backus-Naur-Form definierten Grammatik vorgegeben (siehe [Mal08, Anhang]). In den nachfolgenden Abschnitten wird auf die in der Martix gespeicherten Informationen über die Muster und über deren Zusammenhang zu Qualitätsmerkmalen eingegangen.

Informationen über Muster

In den Vorspalten der Matrix werden Informationen über das jeweilige Muster (ein Muster pro Zeile) hinterlegt. Diese Informationen beinhalten Referenzen auf

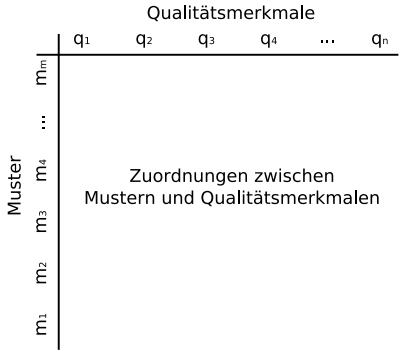

Abbildung 3.5.: Zuordnung von Mustern zu Qualitätsmerkmalen nach Malich (in Anlehnung an [Mal08, Abb.5-3; S.142])

veröffentlichte Werke, in welchen die jeweiligen Muster beschrieben sind, für welche Szenarien die Muster anwendbar sind und Informationen über Beziehungen zu weiteren Mustern. Im folgenden wird auf die Bedeutung der anwendbaren Szenarien und der Beziehungen zu anderen Mustern eingegangen und beschrieben, wie die Informationen hinterlegt sind.

Anwendbare Szenarien. In [ZBJ04, Bab04] haben *Zhu et al.* gezeigt, wie Informationen über die Szenarien (nach *Bass et al.* [BCK03]), für die ein Muster anwendbar ist, aus der Musterbeschreibung extrahiert werden können. Durch eine explizite Repräsentation dieser Informationen ist es leichter die Anwendbarkeit von Mustern für Anforderungen, die in Form von Szenarien vorliegen, zu überprüfen. *Malich* hat die explizite Repräsentation von anwendbaren Szenarien von Mustern in seine Arbeiten aufgenommen. Die Szenarien werden jedoch nach der von *Malich* definierten Grammatik lediglich in Form von natürlicher Sprache hinterlegt und nicht in die von *Bass et al.* vorgeschlagenen Anteile untergliedert (siehe [Mal08, S.152]).

Beziehungen zu anderen Mustern. In den Vorspalten werden Beziehungen der Muster zueinander hinterlegt. Zu jedem Muster können Beziehugnen zu anderen Mustern hinterlegt werden. Die Beziehungen gehören zu einem der fünf Beziehungstypen „uses“, „is used by“, „generalizes“, „specialises“ und „is alternative to“ (siehe Abbildung 3.6). Die Nutzungs- sowie die Vererbungsbeziehung werden bei *Malich* jeweils redundant hinterlegt, um ein Navigieren von jedem Muster aus möglich zu machen. Ergo existieren nach *Malich* insgesamt drei Formen der Beziehungen zwischen Mustern. Zusätzlich wird bei der Nutzungsbeziehung angeben, ob ein Muster die Verwendung eines anderen Musters impliziert (requires) oder lediglich eine Nutzung möglich ist (optional).

```
     Relationships  ::=  Relationship*

      Relationship  ::=  " - " RelationshipType " (" Cardinality "): "
                         PatternName PageReferences SectionReferences

  RelationshipType  ::=  "uses"| "is used by"|
                         "generalises"| "specialises"|
                         "is alternative to"
       Cardinality  ::=  "optional"| "required"
```

Abbildung 3.6.: Produktionsregeln für Beziehungen zwischen Mustern aus der Grammatik von Malich (aus [Mal08, Abb. 5-8; S.154])

Zusammenhang zwischen Mustern und Qualitätsmerkmalen

In den Zellen der Matrix liefern Einträge genauere Informationen über den Zusammenhang zwischen Muster und Qualitätsmerkmal der entsprechenden Zeile und Spalte. Die Einträge in den Zellen entsprechen in ihrer Form einer in Backus-Naur-Form definierten Grammatik. Nach dieser besteht der Zusammenhang zwischen Muster und Qualitätsmerkmal aus mehreren Auswirkungen des Musters auf das Qualitätsmerkmal. Jede dieser Auswirkungen fällt in eine der vier Kategorien „PositiveResponse“, „Sensitivitypoint“, „Tradeoffpoint“ oder „NegativeResponse“ (siehe Abbildung 3.7). Eine Zelle kann jedoch auch leer gelassen werden, was so zu interpretieren ist, dass das Muster der entsprechenden Zeile keinen Einfluss auf das Qualitätsmerkmal der entsprechenden Spalte hat (siehe [Mal08, S.202]).

```
EffectOnQualityCharacteristic  ::=  ( PositiveResponse )*
                                    ( Sensitivitypoint )*
                                    ( Tradeoffpoint )*
                                    ( NegativeResponse )*
```

Abbildung 3.7.: Ausschnitt der Grammatik für eine Zelle (aus [Mal08, Abb. 5-9; S.157])

Jeder Eintrag, der einer der Kategorien entspricht, enthält einen Verweis auf die Argumentation in einer bereits veröffentlichten Musterbeschreibung, in welcher dargelegt wird, weshalb ein entsprechender Einfluss auf das Qualitätsmerkmal existiert. Dies gilt für Einträge aus allen vier Kategorien gleichermaßen. Exemplarisch wird hier die Produktionsregel für Einträge aus der Kategorie „PositiveResponse“ gegeben (siehe Abbildung 3.8). Das Symbol „PositiveResponseDescription“ besteht aus beliebigen Zeichenfolgen und ist eine informelle Beschreibung der Argumentation bezüglich des Einflusses des Musters auf das Qualitätsmerkmal in Kurzform. Mit den Symbolen „PageReferences“ und „SectionRefe-

rences“ wird auf veröffentlichte Musterbeschreibungen verwiesen. Für genauere Erläuterungen bezüglich der Grammatik sei auf [Mal08] selbst verwiesen.

```
PositiveResponse  ::=  " - Positive response: "
                       PositiveResponseDescription
                       PageReferences SectionReferences
```

Abbildung 3.8.: Produktionsregel für das Element „PositiveResponse“ (aus [Mal08, Abb. 5-11; S.159])

Für Einträge der Kategorien „Sesitivitypoint“ und „Tradeoffpoint“ werden noch zusätzliche Informationen angegeben. Die zusätzlichen Informationen betreffen insbesondere Angaben, ob ein positiver oder ein negativer Einfluss auf das entsprechende Qualitätsmerkmal durch den Sensitivity oder Tradeoff Point[2] gegeben ist und, bei Tradeoff Points, welche weiteren Qualitätsmerkmale durch den Tradeoff Point betroffen sind.

3.5.2. Eine Ontologie für die Architekturbewertung

In [EA08] haben *Erfanian* und *Aliee* eine Ontologie von Begriffen, die in ihren Augen bei der Bewertung von Softwarearchitekturen von Bedeutung sind, erarbeitet. Bei der Konstruktion ihrer Ontologie haben sich *Erfanian* und *Aliee* auf die ABAS aus [KK99, KKB+99] sowie auf die Konzepte, die in ATAM Verwendung finden, gestützt. Eine grafische Repräsentation der Ontologie von *Erfanian* und *Aliee* in UML Notation wird in den Abbildungen 3.9 und 3.10 wieder gegeben. Für genauere Erläuterungen der gesamten Ontologie sei auf [EA08] verwiesen. An dieser Stelle werden nur die für die vorliegende Arbeit wichtigsten Aspekte hervorgehoben.

Die zentralen Elemente der Ontologie wurden in Abbildung 3.9 (abweichend vom Original) grau hinterlegt. Es handelt sich um die Elemente „Scenario“, „Architectural Decision“, „Quality Attribute“ und „Architectural Style“. Die Beziehungen des Elements „Architectural Style“ werden gesondert in Abbildung 3.10 verfeinert. Das heißt, es werden zusätzliche Beziehungen zu zusätzlichen Elementen dargestellt.

Wichtig für die Bewertung und deshalb in Abbildung 3.9 auch grau hinterlegt ist die Darstellung von Risiken, Nicht-Risiken sowie Sensitivity und Tradeoff Points. Diese werden als Eigenschaft der Beziehung zwischen Architekturentscheidungen und Szenarios dargestellt. Dies entspricht der Definition von Qualität als Erfüllung festgelegter Erfordernisse (siehe Definition und Erläuterungen

[2] Zur Erläuterung von Sensitivity und Tradeoff Points siehe die Beschreibung in Abschnitt 3.2, Schritt 6 auf Seite 43.

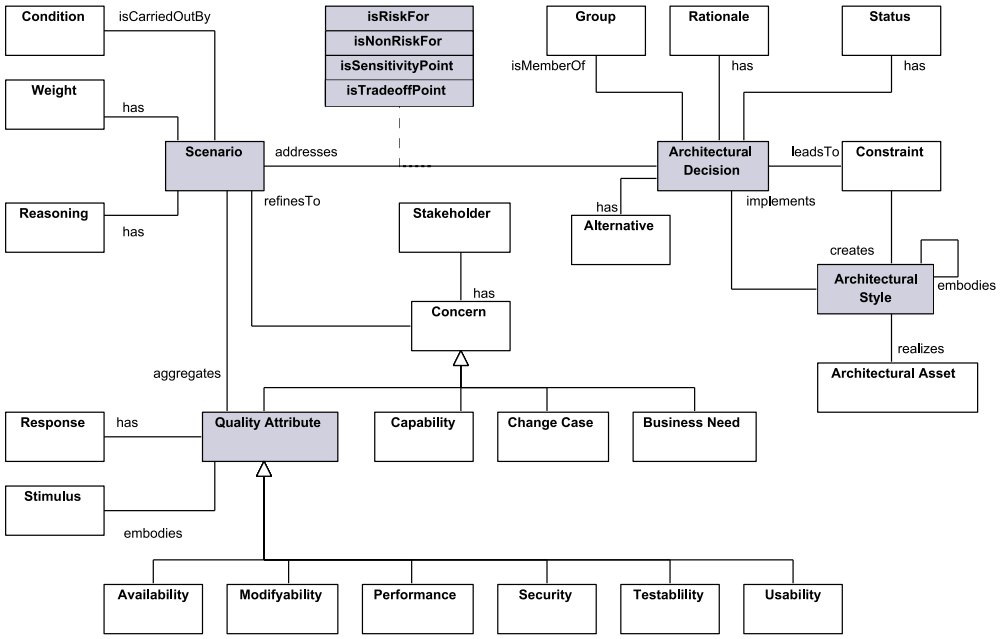

Abbildung 3.9.: Ontologie für die Architekturbewertung nach *Erfanian* und *Aliee* ([EA08])

zu Qualität in Abschnitt 2.1 auf Seite 12). Die festgelegten Erfordernisse werden in den Szenarios geliefert. Diese beinhalten einen Bezug (mit einer Ausprägung) zu einem Qualitätsmerkmal. Die Architekturentscheidung kann einen Einfluss auf ein oder mehrere der Ausprägungen eines Qualitätsmerkmals des resultierenden Systems haben. Dieser Einfluss kann nach der Ontologie von *Erfanian* und *Aliee* ein Risiko, Nicht-Risiko, Sensitivity oder Tradeoff Point sein.

Ein Architekturstil[3] umfasst nach *Erfanian* und *Aliee* mehrere Architekturentscheidungen. In Abbildung 3.10 wird zudem deutlich, dass ein Architekturstil Taktiken (nach *Bass et al.*; siehe Ausführungen in Abschnitt 2.2.1 auf Seite 17) realisiert.

Im weiteren Verlauf der vorliegenden Arbeit wird eine Ontologie für die Architekturbewertung erarbeitet. Dabei wird auf die hier erläuterte Ontologie Bezug genommen.

[3] Die Bezeichnung „Architekturstil“ (engl. *Architectural Style*) wird oft als alternative Bezeichnung für Architekturmuster verwendet.

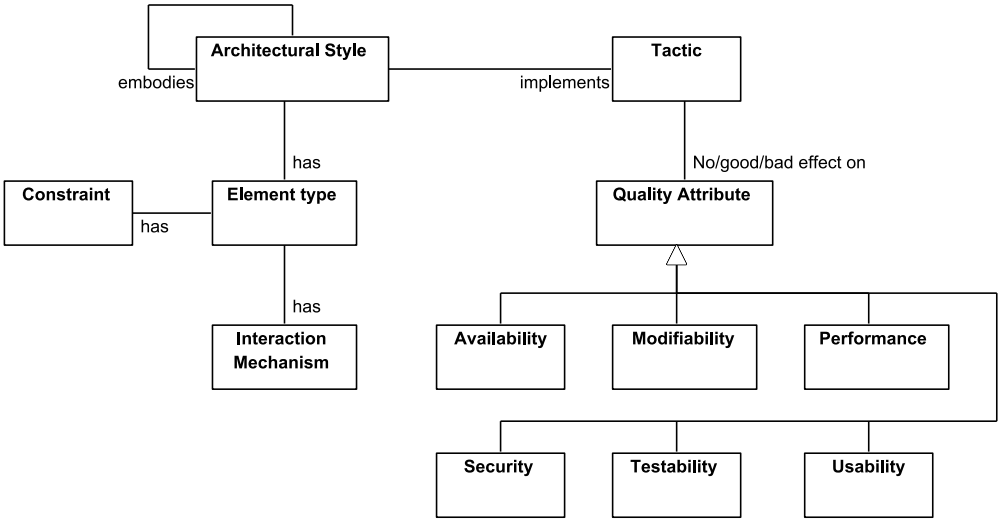

Abbildung 3.10.: Ontologie (Teil über Muster) für die Architekturbewertung ([EA08])

KAPITEL 4

Methodik zur Architekturbewertung

In diesem Kapitel wird die im Rahmen dieser Arbeit entwickelte Methode zur Architekturbewertung, POSAAM, vorgestellt.

Der erste Abschnitt gibt einen Überblick über die wichtigsten Konzepte von POSAAM. Diese Konzepte werden im weiteren Verlauf des Kapitels vertieft. Die Struktur des Kapitels wird anhand der im ersten Abschnitt präsentierten Konzepte genauer erläutert.

Inhalt

4.1. Übersicht über die Methode

Die neue Methode, deren Entwicklung in diesem Kapitel beschrieben wird, heißt POSAAM (**P**attern **O**riented **S**oftware **A**rchitecture **A**nalysis **M**ethod). Wie der Name bereits andeutet, nutzt die Methode das in Mustern gekapselte Expertenwissen für die Architekturbewertung. Bevor in Abschnitt 4.1.2 POSAAM in seinen zentralen Zügen erläutert wird, werden in 4.1.1 die Ziele, die die Entwicklung von POSAAM getrieben haben, kurz dargelegt. Nach der Erläuterung von POSAAM auf abstraktem Niveau kann gezeigt werden, wie das Kapitel aufgebaut ist, um konkretere Aspekte von POSAAM zu detaillieren. Der weitere Aufbau dieses Kapitels wird deshalb in 4.1.3 dargelegt.

4.1.1. Ziele und Herausforderungen an eine neue Methode zur Architekturbewertung

Aus den Mängeln, die bei der Analyse der qualitativen Architekturbewertungsmethoden aus Kapitel 3 festgestellt wurden, ergeben sich Ziele für die in dieser Arbeit vorgestellten Methode. Die zu erarbeitende Methode hat zum Ziel,

Z1 die Abhängigkeit von Experten bei der Architekturevaluation zu vermindern,

Z2 die Systematik und damit die Wiederhol- und Nachvollziehbarkeit bei der Architekturevaluation zu erhöhen,

Z3 die Integration der Architekturevaluation in den Entwicklungsprozess zu verbessern,

Z4 das aus einer Evaluation gewonnene Wissen systematischer wiederzuverwenden sowie

Z5 den Schwerpunkt der Evaluation von der Anforderungserhebung, der nachträglichen Architekturdokumentation und dem Social-Engineering weg und hin zur eigentlichen Architekturevaluation - der Feststellung wie die Anforderungen in der Architektur berücksichtigt werden - zu bewegen.

Die Nummerierung der Ziele dient der späteren Identifikation und hat keinerlei Bedeutung hinsichtlich einer möglichen Priorisierung. Im Folgenden wird aufgezeigt, wie POSAAM funktioniert und wie den hier aufgelisteten Zielen durch die Methode Rechnung getragen wird.

4.1.2. Die zentralen Konzepte von POSAAM

Um die Ziele aus 4.1.1 zu erreichen, wurden in POSAAM Konzepte eingearbeitet. Bereits vor der Evaluation müssen Anforderungen und Architektur spe-

zifiziert sein (Z6). Die weitere Verwendung von Evaluationsergebnissen, ist in POSAAM explizit vorgesehen (Z3). Die Evaluation folgt einer Systematik, die auf der Verwendung von Mustern und Prinzipien basiert (Z1 und Z2). Das zu einem Muster gespeicherte Expertenwissen beinhaltet Informationen über die Verwendung quantitativer Bewertungsmethoden (Z4). Und die Verbesserung des Evaluationsprozesses ist in POSAAM explizit vorgesehen (Z5). Diese Konzepte werden nachfolgend in der genannten Reihenfolge vertieft.

Um POSAAM zu erläutern werden einige Diagramme verwendet. Diese und im weiteren Verlauf des Kapitels folgende Abbildungen zur Erläuterung der Abläufe in POSAAM orientieren sich an den Aktivitätsdiagrammen der UML 2.0 [Obj07b, Obj07a], entsprechen aber in einigen Punkten nicht genau der UML 2.0. Insbesondere wird auf die PIN-Notation verzichtet. Des Weiteren werden in den Abbildungen nicht alle Relationen zwischen Elementen dargestellt, da dies der Übersichtlichkeit der Abbildungen schaden würde. Beispielsweise hat in POSAAM jede Aktionen immer einen Einfluss auf mindestens ein Artefakt (Objektknoten). Diese Einflüsse jedoch alle in der Abbildung darzustellen, würde die Übersichtlichkeit der Abbildungen stark beeinträchtigen. Der genaue Ablauf der Methode wird deshalb jeweils im Text erläutert. Die Abbildungen dienen lediglich dem Überblick und besseren Verständnis.

Eingaben für eine Evaluation

In Abbildung 4.1 ist eine Übersicht über POSAAM gegeben. Um den Schwerpunkt von der Anforderungserhebung und der nachträglichen Architekturdokumentation weg zu bewegen (Z6), werden in POSAAM eine Beschreibung der Anforderungen sowie der Architektur vorausgesetzt. Auch wenn die Anforderungen nicht im Rahmen des Bewertungsprozesses erhoben werden, sind sie für die Durchführung einer Evaluation erforderlich, da nach dem Qualitätsverständnis, welches dieser Arbeit zugrunde liegt, sich Qualität auf die Erfüllung festgelegter oder vorausgesetzter Erfordernisse bezieht (vgl. Definition von Qualität auf S. 12). Ohne eine Anforderungsspezifikation existieren auch keine Erfordernisse deren Erfüllung oder Begünstigung bzw. Gefährdung geprüft werden könnten. Architektur- sowie Anforderungsspezifikation sind Eingaben für den Evaluationsprozess in POSAAM (vgl. die Eingaben der Aktivität in Abbildung 4.1).

Um sicherzustellen, dass eine Architekturbewertung nach POSAAM mit den vorhandenen Beschreibungen durchführbar ist, werden sowohl die Architekturspezifikation als auch die Spezifikation der architekturrelevanten Anforderungen einer Prüfung unterzogen (die ersten beiden Aktionen in der Aktivität aus Abbildung 4.1). Der genaue Ablauf der Prüfungen wird im weiteren Verlauf des vorliegenden Dokuments noch detaillierter beschrieben. Die Ergebnisse der Prüfungen

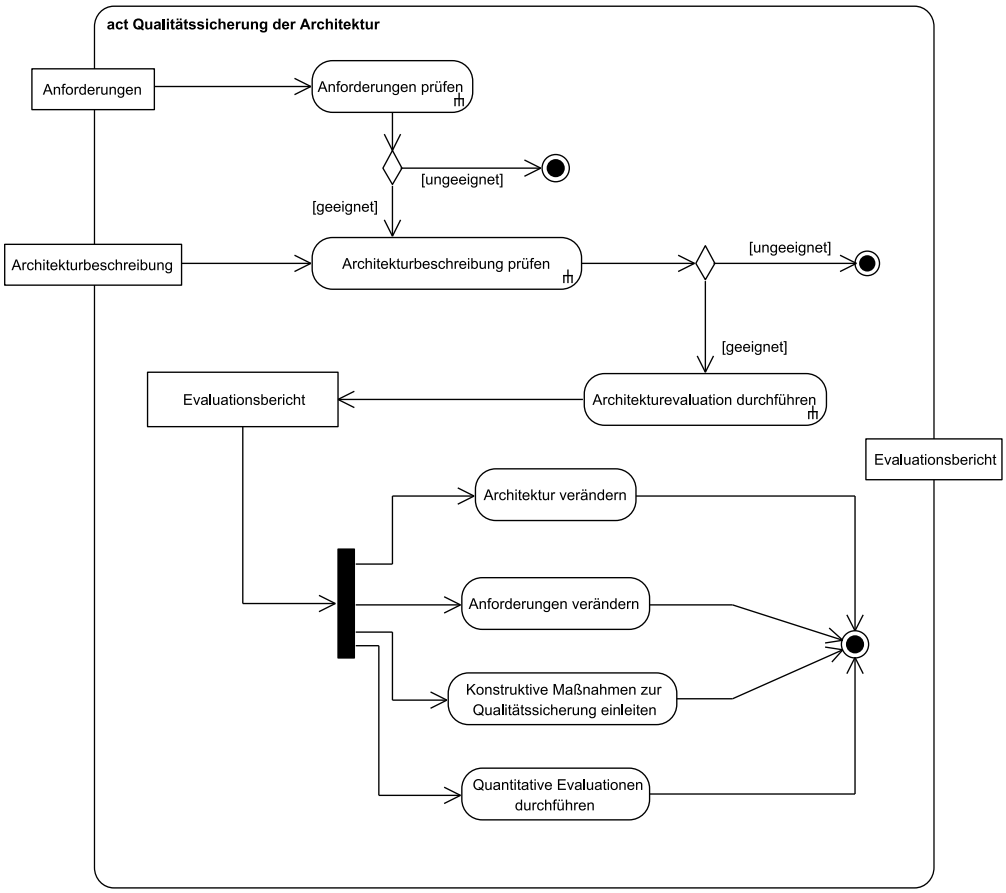

Abbildung 4.1.: Überblick über POSAAM

werden im Evaluationsbericht festgehalten. Ist eine der Prüfungen nicht erfolgreich, so kann eine Architekturbewertung nach POSAAM nicht stattfinden. In diesem Fall wird POSAAM abgebrochen. Eine erneute POSAAM Evaluation macht nur Sinn, wenn die im Evaluationsbericht dokumentierten Mängel an der Anforderungs- und/oder Architekturdokumentation behoben wurden.

Verwendung der Evaluationsergebnisse

Bei erfolgreicher Prüfung der Anforderungs- und Architekturspezifikation kann mit dem Kern des Evaluationsprozesses begonnen werden. Ergebnis der Evaluation ist ein Bericht, der für den weiteren Verlauf des Entwicklungsprozesses genutzt wird. Die weitere Verwendung von Ergebnissen der Evaluation im Entwicklungsprozess stellt eines der Ziele für POSAAM dar (Z3). In erster Linie

kann eine Architekturevaluation zwei Ausgänge haben. Das Resultat der Evaluation kann entweder sein, dass die zu evaluierende Architektur den spezifizierten Anforderungen gerecht wird oder dass dies nicht der Fall ist. Im letzteren Fall wurden im Verlauf des Evaluationsprozesses Mängel festgestellt und im Evaluationsbericht dokumentiert. Aus den im Evaluationsbericht dokumentierten Mängeln lässt sich schließen, ob die Architektur für die spezifizierten Anforderungen verbessert und deshalb einer Änderung unterzogen werden kann oder ob zwar die Architektur den Anforderungen nicht gerecht wird, eine zufrieden stellende Verbesserung der Architektur jedoch nicht ersichtlich ist. In diesem Fall ist eine Änderung der Anforderungen nötig. Ist das Resultat einer Evaluation, dass die zu evaluierende Architektur den spezifizierten Anforderungen entspricht, können Informationen aus dem Evaluationsbericht genutzt werden, um für den weiteren Entwicklungsprozess weiter gehende Qualitätssicherungsmaßnahmen zu definieren.

Expertenunabhängigkeit durch den Einsatz von Mustern

Um das Ziel der höheren Unabhängigkeit von Experten (Z1) zu erreichen, nutzt POSAAM das in Mustern gekapselte Expertenwissen. Ohne eine geeignete Strukturierung des in Mustern gekapselten Expertenwissens und eine systematische Vorgehensweise, über die diese Strukturierung genutzt wird, ist es jedoch nur möglich das in Mustern gekapselte Wissen für eine Evaluation zu nutzen, wenn dem Evaluationsteam das Wissen aus den Mustern bereits bekannt ist. Ergo, wenn das Evaluationsteam bereits aus Experten besteht. Um diesem Problem entgegen zu wirken, werden in POSAAM die Muster mit zusätzlichen Informationen versehen und geeignet strukturiert. Zusätzlich bietet POSAAM eine Systematik, um diese Informationen und Strukturen für eine geeignete Suche zu verwenden. In Anlehnung an [Mal08] wird die Form für die strukturierte Sammlung von Mustern und von Informationen, die zu den Mustern in Beziehung gesetzt werden, von nun an als Wissensmodell bezeichnet. Informationen über Muster, die bereits in der durch das Wissensmodell definierten Form vorliegen, werden von nun an als Wissensbasis bezeichnet.

Gemäß dem SARA Report (siehe Abschnitt 3.3.5 auf Seite 54) liegt der Zweck einer Architekturevaluation darin, den Einfluss jeder architekturrelevanten Entscheidung auf jede architekturrelevante Anforderung zu verstehen [OKK+02, S.11]. Um diesen Zweck zu erfüllen liegen mindestens zwei systematische Vorgehensweisen auf der Hand. Zum einen können die architekturrelevanten Entscheidungen nacheinander durchgegangen werden und für jede dieser Entscheidungen festgehalten werden, welchen Einfluss sie auf welche architekturrelevanten Anforderungen hat. Zum anderen bietet sich auch der umgekehrte Weg an, bei dem die architekturrelevanten Anforderungen durchgegangen werden und für jede

dieser Anforderungen festgehalten wird, welche architekturrelevanten Entscheidungen einen Einfluss auf die aktuell betrachtete Anforderung haben. Um das in Mustern gekapselte Expertenwissen bei der Evaluation einsetzen zu können, eignet sich der zweite Weg besser. Muster bieten für definierte Probleme geeignete Lösungen. Bei Architekturmustern beinhaltet die Lösung Hilfestellungen zu architekturrelevanten Entscheidungen. Somit bilden Architekturmuster die Brücke zwischen architekturrelevanten Anforderungen und architekturrelevanten Entscheidungen. Während es jedoch schwierig ist, zu einer architekturrelevanten Entscheidung entsprechende Muster zu finden, die diese Entscheidung als Lösung beinhalten, ist die Suche nach Mustern, die architekturrelevante Anforderungen lösen, einfacher. Dies liegt an der Natur der Gestaltung von Mustern. Sie sind vom Problem zur Lösung ausgelegt. Dennoch ist die Suche nach Mustern zu einer Anforderung über die Problembeschreibungen der Muster noch immer nicht sehr effizient.

Muster werden in POSAAM an Qualitätsmerkmale und Qualitätsteilmerkmale gekoppelt. Dadurch ist eine effizientere Suche nach Mustern für Qualitätsmerkmale möglich. In POSAAM wird gefordert, dass die spezifizierten architekturrelevanten Anforderungen auch einem Qualitätsmerkmal oder Qualitätsteilmerkmal zugeordnet sind. Durch die Einordnung sowohl der Muster als auch der architekturrelevanten Anforderungen in einen Qualitätsbaum, können die Muster zu den Anforderungen in Beziehung gebracht werden. Dadurch kann zu einer architekturrelevanten Anforderung eine Menge von Mustern identifiziert werden, die für eine Beeinflussung des Qualitätsmerkmals, das in der architekturrelevanten Anforderung gefordert wird, verwendet werden könnten. Mit diesem Schritt beginnt auch der Evaluationsprozess von POSAAM (vgl. Abbildung 4.2).

Im folgenden Schritt geschieht die Identifikation eines der Muster aus der Menge der im ersten Schritt identifizierten Muster in der vorliegenden Architekturspezifikation. Wird ein Muster in der vorliegenden Architekturspezifikation identifiziert, wird die Wissensbasis über Muster genutzt, um Prüfungen bezüglich des eingesetzten Musters durchzuführen (letzte Aktion in Abbildung 4.2). Dazu gehören Prüfungen, ob

1. das „richtige“ Muster verwendet wurde (d.h., dass keines der anderen zur Verfügung stehenden Muster eine bessere Alternative dargestellt hätte).
2. das Muster in der Architekturspezifikation korrekt eingesetzt wurde.
3. das Muster entsprechend der Anforderungsspezifikation korrekt konfiguriert wurde.
4. das Muster in der richtigen Kombination mit anderen Mustern verwendet wurde.

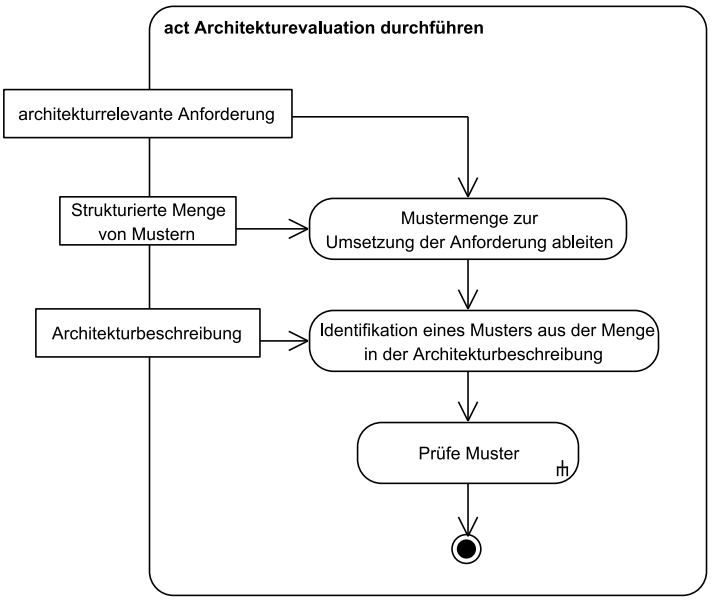

Abbildung 4.2.: Drei zentrale Schritte bei der Evaluation nach POSAAM

Einsatz von Prinzipien

Die soeben erläuterte Form der Prüfung funktioniert jedoch nicht in allen Fällen. Folgende Probleme können bei einer Prüfung mit der Hilfe von Mustern auftreten:

- Für eine gegebene Anforderung existiert kein Muster, welches für die Lösung der Anforderung einzusetzen ist. Dies kann z.B. dann der Fall sein, wenn es sich um eine Anforderung handelt, die in einem sehr speziellen Kontext auftritt und somit eine gänzlich neue Problemlösung erfordert.
- In der Wissensbasis von Mustern, die für die Evaluation bereit steht, existiert kein Muster, welches für die Lösung der Anforderung geeignet wäre.
- Obwohl Muster für eine Anforderung existieren, präsentiert der Einsatz dieses Musters nicht die beste Lösung für die gegebene Anforderung. Dies kann dann der Fall sein, wenn das Muster nicht ausgereift ist oder die Anwendbarkeit des Musters nicht klar abgegrenzt ist.
- Keines der Muster, die nach der vorliegenden Wissensbasis für eine Anforderung geeignet wären, ist in der Architekturbeschreibung auffindbar. Dafür kann es zwei Gründe geben: Das Muster ist in der Architekturbeschreibung nicht vorhanden oder das Muster wird in der Architekturbe-

schreibung nicht explizit genug beschrieben, als dass es durch das Evaluationsteam identifiziert werden könnte.

Um trotz dieser Probleme die Systematik und Nachvollziehbarkeit zu erhalten, werden in POSAAM Prinzipien genutzt. Ähnlich zu den Taktiken nach *Bass et al.* handelt es sich bei Prinzipien um allgemeine Regeln, Richtlinien oder Heuristiken, die bei einer Problemlösung verwendet werden können. Die Bezeichnung Taktiken wird vermieden, da dieser Begriff durch *Bass et al.* vorbelegt ist und insbesondere auch konkretere Problemlösungen in der Art von Mustern umfasst (siehe Diskussion der Taktiken nach *Bass et al.* in Abschnitt 2.2.1 auf den Seiten 17ff.).

Auch Prinzipien können Qualitätsmerkmalen oder -teilmerkmalen zugeordnet werden. Dadurch können analog zu den Mustern auch Prinzipien zu den Anforderungen in Beziehung gesetzt werden. Tritt im Rahmen einer Evaluation eines der oben genannten Probleme auf, wird nach alternativen Architekturmechanismen gesucht, die den Prinzipien gerecht werden. Durch dieses Verfahren ist die Systematik und Nachvollziehbarkeit (Z2) auch noch gewährleistet, wenn die Suche nach Mustern nicht erfolgreich ist.

Verbesserung des Evaluationsprozesses

Grundlage für die Durchführung einer Evaluation nach POSAAM ist eine Wissensbasis von Mustern, die einem Wissensmodell, wie es POSAAM vorgibt, entspricht. Strukturen, die im Wissensmodell vorgegeben werden, sind

- Informationen, die implizit in gängigen Musterbeschreibungen vorhanden sind, werden in der Wissensbasis expliziert.
- Zusätzliche Informationen zu Mustern, die in gängigen Musterbeschreibungen nicht vorhanden sind, werden in der Wissensbasis abgelegt.
- Informationen über die Zusammenhänge zwischen Mustern werden in der Wissensbasis abgelegt. Manche dieser Zusammenhänge sind implizit in gänigen Musterbeschreibungen vorhanden (siehe Abschnitt 2.3.2 auf Seite 30). Andere (wie z.B. die Wahl zwischen alternativen Mustern für ein gegebenes Problem) sind nicht in den gängigen Musterbeschreibungen an sich abgelegt, jedoch aus anderen Quellen (z.B. in Vorgehensbeschreibungen in Mustersprachen) verfügbar.

Die Informationen, die in eine Wissensbasis, auf der eine Evaluation nach POSAAM basiert, aufgenommen werden, entstammen aus gängigen Musterbeschreibungen. Die Zahl der gängigen Musterbeschreibungen ist bereits umfangreich (laut [Boo07] existieren bereits über zweitausend dokumentierte Muster)

und wächst beständig. Zusätzlich werden die gängigen Musterbeschreibungen ständig aktualisiert und verbessert. Daraus folgt, dass die Wissensbasis von POSAAM einem ständigen Wandel unterzogen ist bzw. dass es bei einer POSAAM Evaluation immer der Fall sein kann, das die Wissensbasis nicht auf aktuellem Stand ist. Dies liegt daran, dass die Wissensbasis von POSAAM zunächst einen initialen Aufbauprozess durchlaufen und die Änderungen aus den Musterbeschreibungen nachvollziehen muss.

Aus diesem Grund existieren in POSAAM explizite Mechanismen zur Verbesserung der Wissensbasis. Beispielsweise ist in POSAAM vorgesehen eine Ergänzung der Wissensbasis vorzunehmen, wenn in der zu prüfenden Musterbeschreibung die Verwendung eines Musters festgestellt wird, welches in der Wissensbasis nicht enthalten ist. Die explizite Aufnahme von Aktivitäten dieser Art in POSAAM dient dem Erreichen des Ziel Z5.

4.1.3. Weiterer Aufbau des Kapitels

Voraussetzung für die Durchführung einer Architekturbewertung nach POSAAM ist das Vorliegen einer Wissensbasis. Der Aufbau der Wissensbasis für eine Evaluation nach POSAAM wird durch das Wissensmodell, welches durch POSAAM vorgegeben wird, festgelegt. Die Erarbeitung des Wissensmodells richtet sich nach der Ontologie, die im Rahmen der vorliegenden Arbeit für POSAAM geschaffen wurde. Sowohl die Ontologie als auch das Wissensmodell für eine Evaluation nach POSAAM werden im folgenden Abschnitt 4.2 erläutert.

Die Beschreibung von POSAAM geschieht in den drei folgenden Abschnitten. Das Verfahren zur Prüfung der gelieferten Beschreibungen (Architekturbeschreibung als auch Beschreibung der Anforderungen) für eine Evaluation nach POSAAM wird in Abschnitt 4.3 erläutert. In Abschnitt 4.4 wird der Kern der Methode dargelegt. Es werden die Aktivitäten, die bei einer POSAAM-Evaluation durchzuführen sind erläutert. Im Anschluss daran wird in Abschnitt 4.5 dargelegt, welche Ergebnisse im Laufe der Evaluation entstanden sind und wie mit diesen weiter zu verfahren ist.

Nach der Beschreibung von POSAAM wird in Abschnitt 4.6 eine Einordnung in das Klassifikationsschema von *Eicker et al.* vorgenommen.

4.2. Eine Ontologie und ein Wissensmodell für POSAAM

Um eine Evaluation nach POSAAM durchzuführen, werden Informationen über Zusammenhänge und Eigenschaften von Mustern genutzt, die vorher gesammelt

und geeignet hinterlegt wurden. Welche Informationen über Zusammenhänge und Eigenschaften von Mustern auf welche Weise zu hinterlegen sind, um diese Informationen während der Evaluation abrufen zu können, wird im Wissensmodell in Abschnitt 4.2.2 erarbeitet.

Zuvor wird in Abschnitt 4.2.1 eine Ontologie für POSAAM erarbeitet. In der Ontologie werden wichtigsten Konzepte und deren Zusammenhänge aufgezeigt, die die Grundlage für die Evaulation nach POSAAM bilden. In der Ontologie finden sich auch Konzepte und Zusammenhänge, die das Wissensmodell betreffen, weshalb sie an dieser Stelle erarbeitet wird.

4.2.1. Eine Ontologie für POSAAM

Ausgangspunkt für die Ontologie für POSAAM sind die wichtigsten Konzepte, die bei einer Evaluation nach POSAAM eine Rolle spielen werden. Dies sind die Konzepte:

- Muster,
- Architekurmuster,
- Prinzip,
- Qualitätsmerkmal und
- Anforderung

Nachfolgend wird nun die Ontologie anhand dieser Konzepte schrittweise aufgebaut.

Muster und Architekturmuster betreffender Anteil der Ontologie

In Abschnitt 2.3.1 wurde der Begriff Muster diskutiert und definiert. Nach der dortigen Definition eines Musters besteht ein Muster immer aus einem wiederkehrenden sowie aus einem variierenden Anteil (vgl. Abbildung 4.3). Für das abstrakte Konzept des Musters (ohne Bezug zu einem Muster aus dem Bereich der Softwareentwicklung) ist diese Aufteilung ausreichend.

Wie in Abschnitt 2.3.3 beschrieben können Muster zueinander in Beziehung stehen. In der POSAAM-Ontologie existieren drei Beziehungen zwischen Mustern.

1. Ein Muster kann eine Verfeinerung eines anderen Musters sein. Dies ist immer dann der Fall, wenn ein Muster den Lösungsraum eines anderen Musters einschränkt.

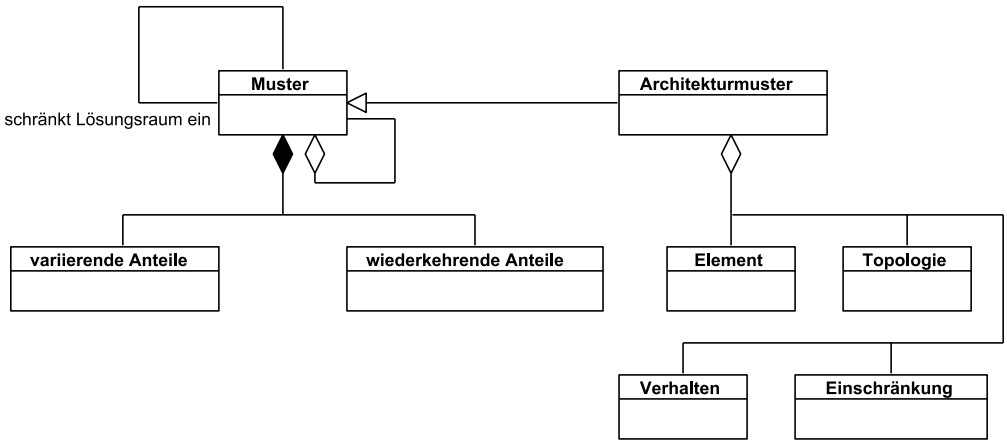

Abbildung 4.3.: Teil der POSAAM-Ontologie Muster betreffend.

2. Ein Muster kann aus einer Kombination anderer Muster bestehen.
3. Ein Muster kann andere Muster verwenden.

Die genannten Beziehungen sind in Abbildung 4.3 sichtbar.

Ein Architekturmuster ist ein Muster und hat somit wiederkehrende sowie variierende Anteile. Zusätzlich beschreibt ein Architekturmuster mögliche Gestaltungen für einen Teil einer Architektur eines Systems. In Anlehnung an [BCK03, S. 125] werden in der POSAAM-Ontologie in einem Architekturmuster Elemente, eine Topologie dieser Elemente (welche Elemente mit welchen anderen Elementen verbunden sind und somit interagieren können), die Art der Interaktion der Elemente und Einschränkungen bezüglich der Elemente, der Topologie und der Interaktionsmöglichkeiten beschrieben. Die wiederkehrenden sowie die variierenden Anteile eines Musters können in einem Architekturmuster beliebige Kombinationen von Elementen, Topologien, Interaktionen und Einschränkungen betreffen.

Anteile der Ontologie Prinzipien betreffend

Die Erarbeitung des Anteils der POSAAM-Ontologie, welcher die Prinzipien betrifft, basiert auf den Erkenntnissen aus Abschnitt 2.2.1.

Ein Prinzip ist eine Regel, Richtline oder Heuristik zur Gestaltung eines Teils einer Architektur eines Systems. Die Befolgung eines Prinzips hat einen bekannten Einfluss auf ein oder mehrere Qualitätsmerkmale (vgl. Abbildung 4.4). Dieser Einfluss kann sowohl positiv als auch negativ sein.

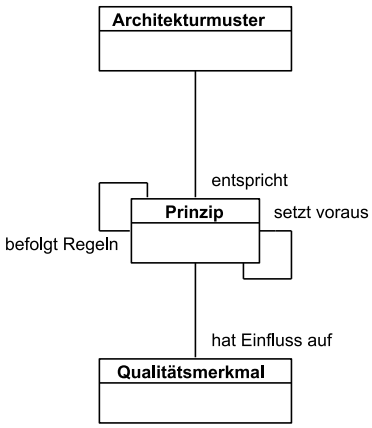

Abbildung 4.4.: Teil der POSAAM-Ontologie Prinzipien betreffend.

Genau wie Muster können auch Prinzipien zueinander in Beziehung stehen. Ein Prinzip kann eine Verfeinerung eines anderen Prinzips sein. Das bedeutet, dass das Befolgen der Regeln, die in einem Prinzip A definiert werden, welches eine Verfeinerung eines Prinzips B ist, immer impliziert, dass die Regeln von Prinzip B befolgt werden.

Eine weitere Beziehung, die zwischen Prinzipien existieren kann, ist, dass ein Prinzip den Einsatz eines anderen Prinzips voraussetzt (siehe Diskussion in Abschnitt 2.2.1).

Die Möglichkeit, dass Prinzipien, wie in den Schlussfolgerungen aus Abschnitt 2.2.1 festgestellt wurde, zueinander im Konflikt stehen oder sich gegenseitig verstärken können, ist durch den Einfluss des Prinzips auf ein Qualitätsmerkmal bereits in der Ontologie verankert.

In Abbildung 4.4 wird die Beziehung zwischen Architekturmustern und Prinzipien dargestellt. Architekturmuster können einem oder mehreren Prinzipien entsprechen. Dadurch ist es möglich, über den Einsatz eines Architekturmusters Qualitätsmerkmale zu beeinflussen.

Dieser Teil der POSAAM-Ontologie ähnelt einem Teil der Ontologie von *Erfanian* und *Aliee* (vgl. Abschnitt 3.5.2, Abbildung 3.10).

Anteile der Ontologie Anforderungen betreffend

In POSAAM werden Anforderungen durch Szenarien (wie sie durch [BCK03] definiert werden) dargestellt. Infolge dessen gibt es in der POSAAM-Ontologie

kein Element mit der Bezeichnung „Anforderung“, sondern nur das Element „Szenario“, welches eine Anforderung darstellt.

Jedes Szenario besteht, genau wie in [BCK03] definiert[1], aus den Elementen Quelle, Reiz, Umgebung, Artefakt, Antwort und Antwortmetrik (vgl. Abbildung 4.5). Ein Szenario definiert somit eine gewünschte Ausprägung eines Qualitätsmerkmals.

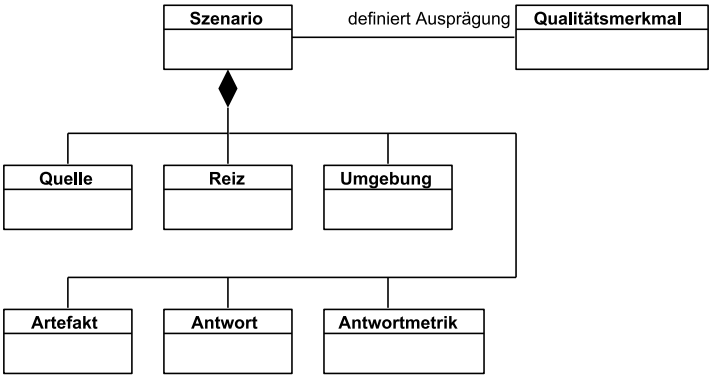

Abbildung 4.5.: Teil der POSAAM-Ontologie Anforderungen betreffend.

Gesamtbild der Ontologie

In Abbildung 4.6 wird ein Überblick über die POSAAM-Ontologie gegeben. Um eine bessere Übersichtlichkeit zu gewährleisten wurden einige Details, die in vorangegangenen Betrachtungen bereits erläutert wurden, in der Abbildung abstrahiert und tauchen deshalb nicht mehr auf. Die Zusammenhänge aus den vorangegangenen Betrachtungen besitzen jedoch nach wie vor Gültigkeit bzw. sind Bestandteil der POSAAM-Ontologie.

Bis auf das Element „Architekturentscheidung“ sind alle Elemente der POSAAM-Ontologie und ihre Zusammenhänge bereits in vorangegangenen Betrachtungen erläutert worden. Genau wie in der Ontologie von *Erfanian* und *Aliee* hat jede Architekturentscheidung ein oder mehrere Alternativen. Die Belegung einer Variationsmöglichkeit eines Architekturmusters ist auch eine Architekturentscheidung. Eine Architekturentscheidung kann die Ausprägung eines oder mehrerer Qualitätsmerkmale beeinflussen (sowohl positiv als auch negativ).

In Abbildung 4.6 werden einige Unterschiede zur Ontologie von *Erfanian* und *Aliee* ([EA08]) deutlich:

[1] Siehe auch Ausführungen zum Konzept „Szenario“ auf Seite 44.

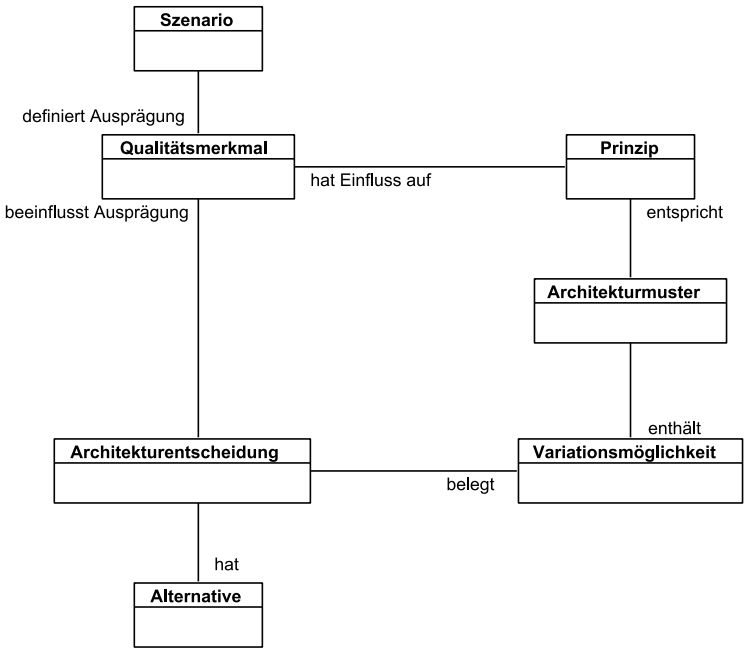

Abbildung 4.6.: Überblick über die POSAAM-Ontologie

- Einige Elemente, die für die Evaluation nach POSAAM unbedeutend sind, wurden nicht in die POSAAM-Ontologie aufgenommen. Z.B. sind in der POSAAM-Ontologie keine Stakeholder vorhanden, da Anforderungen in POSAAM vorausgesetzt werden. Es mag sinnvoll sein, die Assoziation zu Stakeholdern in Anforderungsdokumenten zu hinterlegen. Für POSAAM ist diese Information jedoch nicht relevant.

- Die Beziehung zwischen einer Architekturentscheidung und einem Architekturmuster ist bei der POSAAM-Ontologie über die Variatiosmöglichkeit des Architekturmusters gegeben. Dies liegt an der abweichenden Definition für Muster, die in der vorliegenden Arbeit gegeben wurde (vgl. Abschnitt 2.3.1).

- Ein Risiko ist in der POSAAM-Ontologie eine Diskrepanz zwischen der durch ein Szenario geforderten Ausprägung eines Qualitätsmerkmals und des Einflusses einer Architekturentscheidung auf dasselbe Qualitätsmerkmal.

Die vorliegende Ontologie kann noch mit einer Vielzahl von Konzepten erweitert werden. Um eine gute Übersichtlichkeit zu gewährleisten, wurden lediglich die für eine POSAAM-Evaluation wichtigsten Konzepte dargestellt.

4.2.2. Ein Wissensmodell für POSAAM

Die Evaluation nach POSAAM basiert auf der Nutzung von in Mustern gekapseltem Expertenwissen. Allerdings sind die Beschreibungen der existierenden Architekturmuster für die Nutzung in POSAAM nicht ohne weiteres geeignet. Das Expertenwissen ist in den Architekturmustern in einer Form hinterlegt, die eine effiziente Arbeit bei einer Evaluation nicht zulässt.

Viel für die Evaluation wichtiges Wissen, beispielsweise bezüglich des Zusammenhangs der Muster zu Qualitätsmerkmalen, ist in den Beschreibungen der Architekturmuster implizit hinterlegt. Dieses Wissen muss explizit hinterlegt werden. Zusätzliches, für die Evaluation einer Architektur nützliches Expertenwissen, welches in den existierenden Musterbeschreibungen nicht vorhanden ist, kann an die Architekturmuster gekoppelt werden (z.B. die Zuordnung von Architekturmustern zu Prinzipien).

Malich hat in [Mal08] ein Wissensmodell für die Nutzung von in Mustern gekapseltem Expertenwissen beim Entwurf und der Bewertung von Softwarearchitekturen entwickelt. Im Folgenden wird auf dieses Modell Bezug genommen, bevor ein Wissensmodell für POSAAM erarbeitet wird.

Bezug zum Wissensmodell von Malich

Malich hat in einer Dissertationsarbeit [Mal07], die auch als Buch verfügbar ist [Mal08], bereits eine strukturierte Hinterlegung von in Mustern gekapseltem Expertenwissen zum Zwecke des Entwurfs und der Bewertung von Softwarearchitekturen in Form eines Wissensmodells vorgeschlagen (siehe Abschnitt 3.5.1).

In der Arbeit von *Malich* werden in einer Matrix Informationen zu den Mustern (in den Vorspalten der Matrix) und Informationen zur Beziehung zwischen Mustern und Qualitätsmerkmalen (in den Zellen der Matrix) hinterlegt (siehe Erläuterungen in Abschnitt 3.5.1). Die Form, nach der die Informationen hinterlegt werden ist in einer BNF-Grammatik beschrieben. Da die meisten Einträge in natürlicher Sprache verfasst werden müssen, wird bei der Erarbeitung des Wissensmodells für POSAAM auf eine Definition in BNF-Form verzichtet. Stattdessen wird zu jedem Merkmal, über welches Informationen im Wissensmodell hinterlegt werden, beschrieben, welche Informationen hinterlegt werden und welche Beschreibungsformen für diese Informationen geeignet sind.

Zusätzlich unterscheidet sich das Wissensmodell von POSAAM vom Wissensmodell von *Malich* durch die im Wissensmodell hinterlegten Inhalte. Insbesondere werden im Wissensmodell von POSAAM die wiederkehrenden sowie die variierenden Anteile eines Musters expliziert. Infolgedessen unterscheidet sich auch die

Darstellung von Sensitivity und Tradeoff Points. Diese können im Wissensmodell von POSAAM nur an eine Variationsmöglichkeit geknüpft auftreten. Ein weiterer zentraler Aspekt bezüglich dessen sich die Wissensmodelle unterscheiden, ist die Hinterlegung des Bezugs von Mustern zu Prinzipien und von Prinzipien zu Qualitätsmerkmalen.

Explizieren von implizitem Wissen

Im Folgenden wird beschrieben, wie im Wissensmodell von POSAAM Merkmale eines Musters hinterlegt werden, die in gängigen Musterbeschreibungen implizit bereits enthalten sind. Zu jedem Merkmal wird in einer Tabelle beschrieben, welche Informationen hinterlegt werden und wie diese Information beschrieben wird bzw. welche Beschreibungsformen für diese Informationen geeignet sind.

Um einen Bezug zu den meist wesentlich umfangreicheren veröffentlichten Beschreibungen von Mustern aus dem POSAAM Wissensmodell zu haben, wird eine Referenz zu einer veröffentlichten Beschreibung im Wissensmodell abgelegt (siehe Tabelle 4.1).

Tabelle 4.1.: Im Wissensmodell hinterlegt: Referenz zu einer externen Quelle

Merkmal	Referenz zu Quelle des Musters
Information, die hinterlegt wird	Referenz zu (veröffentlichter) Musterbeschreibung
Form, in der die Information hinterlegt wird	Natürlicher Text, wie er bei der Angabe von Quellen in Sachtexten üblich ist.

Im Wissensmodell für POSAAM wird zu jedem Muster beschrieben, welche Anteile bei einer Instanz des Musters immer vorhanden sein müssen, damit es sich um eine Instanz des Musters handelt. In Tabelle 4.2 wird beschrieben, wie die wiederkehrenden Anteile eines Musters im POSAAM-Wissensmodell hinterlegt werden.

Das Gegenstück zu den wiederkehrenden Anteilen sind die variierenden Anteile eines Musters. Diese werden im POSAAM-Wissensmodell als Variationsmöglichkeiten (siehe Definitionen in Abschnitt 2.3.1 auf Seite 29) dargestellt. Wie die

Tabelle 4.2.: Im Wissensmodell hinterlegt: Wiederkehrende Anteile eines Musters

Merkmal	Wiederkehrende Anteile eines Musters
Information, die hinterlegt wird	Beschreibung der wiederkehrenden Anteile des Musters.
Form, in der die Information hinterlegt wird	Natürlicher Text, bei dem auf die Elemente, Topologie, Interaktion und Einschränkungen eines Musters Bezug genommen wird.

Variationsmöglichkeiten im POSAAM-Wissensmodell hinterlegt werden, ist in Tabelle 4.3 beschrieben.

Tabelle 4.3.: Im Wissensmodell hinterlegt: Variationsmöglichkeiten eines Musters

Merkmal	Variationsmöglichkeiten eines Musters
Information, die hinterlegt wird	Es wird hinterlegt, welche Variationsmöglichkeiten bei der Anwendung des Musters bestehen.
Form, in der die Information hinterlegt wird	Die Variationsmöglichkeiten werden als Liste hinterlegt. In jedem Listeneintrag wird auf eine mögliche Variationsmöglichkeit des Musters eingegangen. Dabei wird in Form von natürlichem Text auf die Elemente, Topologie, Interaktionsmöglichekeiten und Einschränkungen der wiederkehrenden Anteile des Musters Bezug genommen.

In den Arbeiten [ZBJ04, Mal08] wurde gezeigt, wie implizites Wissen, welches in gängigen Musterbeschreibungen hinterlegt ist, aus diesen extrahiert werden kann. Diese Arbeiten bezogen sich u.a. auf das implizit in Musterbeschreibungen hinterlegte Wissen bezüglich

- des Zusammenhangs zwischen Mustern und Qualitätsmerkmalen bzw. der Auswirkungen von Mustern auf Qualitätsmerkmale,
- der Beziehungen zwischen Mustern,
- der in einem Muster vorliegenden Sensitivity Points und deren Auswirkung auf Qualitätsmerkmale und
- der in einem Muster vorliegenden Tradeoff Points und deren Auswirkungen auf Qualitätsmerkmale.

Nachdem erläutert wurde, wie die wiederkehrenden und variierdenen Anteile eines Musters im Wissensmodell von POSAAM hinterlegt werden, ist es nun möglich zu zeigen, wie die soeben genannten Merkmale im POSAAM-Wissensmodell hinterlegt werden.

Im POSAAM-Wissensmodell existieren zwei Formen, wie der Zusammenhang zwischen Mustern und Qualitätsmerkmalen hinterlegt wird. In der einen Form, wird das Muster selbst zu einem oder mehreren Qualitätsmerkmalen in Bezug gesetzt. Dabei wird der Bezug auf das Qualitätsmerkmal auf den Einsatz des Musters unabhängig von möglichen Variationen betrachtet. Es handelt sich sozusagen um die grobe Tendenz des Musters bezüglich eines oder mehrerer Qualitätsmerkmale. Wie diese Form des Bezugs zwischen Mustern und Qualitätsmerkmalen im POSAAM-Wissensmodell hinterlegt wird, ist in Tabelle 4.4 beschrieben.

Tabelle 4.4.: Im Wissensmodell hinterlegt: Durch Muster beeinflusste Qualitätsmerkmale

Merkmal	Beeinflusste Qualitätsmerkmale
Information, die hinterlegt wird	Eine Liste der Qualitätsmerkmale die durch den Einsatz des Musters beeinflusst werden. (Es handelt sich dabei um die Qualitätsmerkmale, die typischerweise, unabhängig von den Variationsmöglichkeiten, durch das Muster beeinflusst werden.)
Form, in der die Information hinterlegt wird	Es wird eine Liste hinterlegt. Jeder Eintrag der Liste enthält einen Verweis auf das beeinflusste Qualitätsmerkmal, auf die Art des Einflusses (positiv oder negativ) und ggf. einen Verweis auf eine Variationsmöglichkeit, durch die die Ausprägung dieses Qualitätsmerkmals beeinflusst wird.

Eine andere Form, wie der Zusammenhang zwischen Mustern und Qualitätsmerkmalen im POSAAM-Wissensmodell hinterlegt wird, ist durch den Bezug der Variationsmöglichkeiten eines Musters. Wenn Variationsmöglichkeiten eines Musters einen Einfluss auf Qualitätsmerkmale haben, so werden diese immer entweder ein Sensitivity oder ein Tradeoff Point sein, da eine Veränderung der Belegung der Variationsmöglichkeit immer eine Auswirkung auf die Ausprägung des Qualitätsmerkmals hat. Wie der Zusammenhang von Variationsmöglichkeiten und Qualitätsmerkmalen im POSAAM-Wissensmodell hinterlegt wird, ist in Tabelle 4.5 beschrieben.

Tabelle 4.5.: Im Wissensmodell hinterlegt: Qualitätseinfluss von Variationsmöglichkeiten

Merkmal	Qualitätseinfluss von Variationsmöglichkeiten
Information, die hinterlegt wird	Für jede Variationsmöglichkeit wird hinterlegt, auf welche Qualitätsmerkmale die Variationsmöglichkeit einen Einfluss hat. Auch wird hinterlegt, wie die Ausprägung des Qualitätsmerkmals in Abhängigkeit der möglichen Belegungen der Variationsmöglichkeiten beeinflusst wird (positiv oder negativ). Wird mehr als ein Qualitätsmerkmal beeinflusst und ist der Einfluss auf die Ausprägung zweier Qualitätsmerkmale bei Veränderung der Belegungen der Variationsmöglichkeit gegenläufig, so handelt es sich bei der Variationsmöglichkeit um einen Tradeoff Point, sonst um ein Sensitivity Point.
Form, in der die Information hinterlegt wird	Für jede Variationsmöglichkeit wird zunächst mit einem der beiden Bezeichner „Tradeoff Point“ oder „Sensitivity Point“ hinterlegt, ob es sich bei dieser Variationsmöglichkeit um einen Tradeoff oder einen Sensitivity Point handelt. Zusätzlich wird eine Liste der Qualitätsmerkmale hinterlegt, die durch die Variationsmöglichkeit beeinflusst werden können. Zu jedem der Qualitätsmerkmale wird ein natürlicher Text hinterlegt, in dem erläutert wird, welche Auswirkungen die unterschiedlichen Möglichkeiten der Belegung auf die Ausprägung des Qualitätsmerkmals haben.

Muster stehen zueinander in Beziehung (vgl. Abschnitt 2.3.3 auf Seite 33). Für eine Evaluation nach POSAAM sind die Beziehungen zwischen Mustern wichtig. Sie werden deshalb im Wissensmodell von POSAAM hinterlegt (siehe Tabelle 4.6).

Tabelle 4.6.: Im Wissensmodell hinterlegt: Beziehungen zu anderen Mustern

Merkmal	Beziehungen zu anderen Mustern
Information, die hinterlegt wird	Es wird hinterlegt, zu welchen anderen Mustern das vorliegende Muster in Beziehung steht. Es gibt in POSAAM drei mögliche Formen der Beziehung zwischen Mustern (vgl. Abschnitt 4.2.1).
Form, in der die Information hinterlegt wird	Zu jedem Muster wird eine Liste von Einträgen hinterlegt, die jeweils aus der Kombination aus einem der Bezeichner „verfeinert“, „besteht aus“ oder „verwendet“ und der Referenz auf eine veröffentlichte Quelle einer Musterbeschreibung bestehen.

Hinzufügen von zusätzlichem Wissen

Zusätzlich zu den Informationen, die in gängigen Musterbeschreibungen bereits implizit enthalten sind, gibt es weitere Informationen über Muster, die für eine Evaluation von Nutzen sein können. Im Anschluss wird erläutert, welche Informationen über das Wissen über Muster im POSAAM-Wissensmodell in welcher Form hinterlegt werden.

Neben Mustern sind Prinzipien für eine Evaluation nach POSAAM bedeutend. Deshalb werden Prinzipien im POSAAM-Wissensmodell hinterlegt (siehe Tabelle 4.7).

Tabelle 4.7.: Im Wissensmodell hinterlegt: Prinzip

Merkmal	Prinzip
Information, die hinterlegt wird	Eine Beschreibung der Regeln, deren Befolgung eine bekannte (positive oder negative) Auswirkung auf ein Qualitätsmerkmal haben.
Form, in der die Information hinterlegt wird	Natürlicher Text. Im natürlichen Text wird die Regel erläutert. Analog zur Hinterlegung von Mustern kann auch einfach eine Referenz auf eine externe Beschreibung des Prinzips gegeben werden.

Analog zu den Anteilen von Mustern im Wissensmodell werden auch einige Merkmale, die an die Prinzipien gekoppelt sind, im Wissensmodell hinterlegt. Die Regeln, die durch die Prinzipien aufgestellt werden, schreiben die Strukturierung des Systems anhand von Eigenschaften der zu strukturierenden Anteile des Systems vor. Beispiele für diese Eigenschaften sind die Wahrscheinlichkeit, mit der Änderungen vorgenommen werden müssen (Wartbarkeit), die Häufigkeit, mit der eine Ressource verwendet wird (Performanz) oder der Geldwert einer gespeicherten Information (Informationssicherheit). Da die Kenntnis über diese Eigenschaften einen Einsatz eines Prinzips und eine Argumentation über den Einfluss von Prinzipien auf Qualitätsmerkmale erst ermöglichen, werden zu einem Prinzip die Eigenschaften hinterlegt, die durch die Regeln des Prinzips genutzt und beeinflusst werden (siehe Tabelle 4.8).

Damit die Prinzipien im Laufe der POSAAM-Evaluation verwendet werden können, muss der Zusammenhang der Prinzipien mit den Qualitätsmerkmalen expliziert werden. Es wird deshalb im POSAAM-Wissensmodell explizit hinterlegt, auf welche Qualitätsmerkmale sich ein Prinzip auswirkt (siehe Tabelle 4.9).

Tabelle 4.8.: Im Wissensmodell hinterlegt: Eigenschaften, die durch Prinzip geregelt werden

Merkmal	Eigenschaften, die durch Prinzip geregelt werden
Information, die hinterlegt wird	Jedes Prinzip regelt die Strukturierung der Architektur oder eines Teils der Architektur anhand von Eigenschaften der zu Regelnden Anteile der Architektur. Welche Eigenschaften durch die Regel wie geregelt werden, wird in diesem Merkmal hinterlegt.
Form, in der die Information hinterlegt wird	Für jedes Prinzip wird eine Liste von Eigenschaften hinterlegt, welche beim Einsatz der Regeln des Prinzips beachtet oder beeinflusst werden. Zu jeder Eigenschaft kann eine Erläuterung in Form von natürlicher Sprache hinterlegt werden.

Tabelle 4.9.: Im Wissensmodell hinterlegt: Qualitätseinfluss von Prinzip

Merkmal	Qualitätseinfluss von Prinzip
Information, die hinterlegt wird	Die positiven sowie negativen Einflüsse, die die Anwendung eines Prinzips auf Qualitätsmerkmale hat.
Form, in der die Information hinterlegt wird	Für jedes Prinzip wird eine Liste von Qualitätsmerkmalen hinterlegt. Für jedes der Qualitätsmerkmale wird hinterlegt, ob es durch das Prinzip positiv oder negativ beeinflusst wird und welche Argumentation hinter diesem Einfluss steht. Die Argumentation basiert u.a. auf den zu einem Prinzip hinterlegten Eigenschaften und erfolgt in Form von natürlichem Text.

In POSAAM basiert jeder Einfluss auf die Ausprägung eines Qualitätsmerkmals, welcher durch die Gestaltung der Architektur eines Systems erlangt wird, auf einem Prinzip. Die Einflüsse auf die Ausprägungen von Qualitätsmerkmalen, die durch den Einsatz eines Musters erzielt werden können, basieren somit auch auf Prinzipien. Im POSAAM-Wissensmodell wird dieser Zusammenhang explizit hinterlegt. Für die durch ein Muster beeinflussten Qualitätsmerkmale wird hinterlegt, mit welchen Prinzipien dieser Einfluss zu begründen ist (siehe Tabelle 4.10).

Analog wird zu jedem Sensitivity oder Tradeoff Point eines Musters hinterlegt, durch welche Prinzipien die Einflüsse des Sensitivity oder Tradeoff Points auf ein oder mehrere Qualitätsmerkmale zu begründen sind (siehe Tabelle 4.11).

Tabelle 4.10.: Im Wissensmodell hinterlegt: Zugrunde liegende Prinzipien

Merkmal	Zugrunde liegende Prinzipien
Information, die hinterlegt wird	Zu jedem in Merkmal „Beeinflusste Qualitätsmerkmale“ hinterlegtes Qualitätsmerkmal wird der Einfluss auf das Qualitätsmerkmal mit einem Prinzip in Bezug gebracht. Anhand des Prinzips wird erläutert, wie der Einfluss auf das Qualitätsmerkmal durch den Einsatz des Musters zu erklären ist.
Form, in der die Information hinterlegt wird	Der Einfluss auf die jeweiligen Qualitätsmerkmale durch Prinzipien wird in Form von natürlichem Text hinterlegt. Bei der Beschreibung wird darauf eingegangen, welche Eigenschaften, die für den Einsatz von Prinzipien von Bedeutung sind, in einem Muster in welcher Form vorliegen bzw. beeinflusst werden.

Tabelle 4.11.: Im Wissensmodell hinterlegt: Zugrunde liegende Prinzipien von Sensitivity oder Tradeoff Points

Merkmal	Zugrundeliegende Prinzipien von Sensitivity oder Tradeoff Points
Information, die hinterlegt wird	Zu jedem Sensitivity oder Tradeoff Point (zu finden im Merkmal „Qualitätseinfluss von Variationsmöglichkeiten“) werden die Einflüsse auf die jeweiligen Qualitätsmerkmale mit Prinzipien in Bezug gebracht. Anhand der Prinzipien wird erläutert, wie der Einfluss auf Qualitätsmerkmale durch Variationen der Variationsmöglichkeiten zu erklären ist.
Form, in der die Information hinterlegt wird	Der Einfluss auf die jeweiligen Qualitätsmerkmale durch Prinzipien wird in Form von natürlichem Text hinterlegt. Bei der Beschreibung wird darauf eingegangen, welche Eigenschaften, die für den Einsatz von Prinzipien von Bedeutung sind, durch eine Variationsmöglichkeit verändert werden.

Prinzipien können zueinander in Beziehung stehen. Die Beziehungen werden im POSAAM-Wissensmodell hinterlegt (siehe Tabelle 4.12).

Integration eines Qualitätsmodells in das Wissensmodell

Das POSAAM zugrunde liegende Qualitätsmodell wird im Wissensmodell hinterlegt. Ein Qualitätsmodell spezifiziert Qualität durch Verfeinerung in Qualitätsmerkmale und -teilmerkmale, denen dann Qualitätsindikatoren zugeordnet werden (vgl. Definitionen von Qualitätsmodell und Qualitätsindikatoren

Tabelle 4.12.: Im Wissensmodell hinterlegt: Beziehungen zu anderen Prinzipien

Merkmal	Beziehungen zu anderen Prinzipien
Information, die hinterlegt wird	Es wird hinterlegt, zu welchen anderen Prinzipien das vorliegende Prinzip in Beziehung steht. Es gibt in POSAAM zwei mögliche Formen der Beziehung zwischen Prinzipien.
Form, in der die Information hinterlegt wird	Zu jedem Prinzip wird eine Liste von Einträgen hinterlegt, die jeweils aus der Kombination aus einem der Bezeichner „befolgt regeln“ oder „setzt voraus“ und der Referenz auf ein Prinzip bestehen.

auf Seite 13). Die Qualitätsmerkmale und deren Verfeinerung in Qualitätsteilmerkmale können aus anderen Qualitätsmodellen (z.B. ISO 9126) übernommen oder im Rahmen des Aufbaus einer Wissensbasis neu definiert werden. Die Zuordnung von Qualitätsindikatoren zu den Qualitätsteilmerkmalen entspricht im POSAAM-Wissensmodell der Zuordnung von Prinzipien zu Qualitätsteilmerkmalen. Die Prinzipien sind in POSAAM die Qualitätsindikatoren einer Softwarearchitektur. Da Muster im Wissensmodell auf Prinzipien verweisen, können auch Muster zum Qualitätsmodell in Bezug gebracht werden. Da es sich bei POSAAM um eine qualitative Form der Qualitätssicherung handelt, werden keine Metriken angegeben, mit welchen ein Maß der Qualitätsindikatoren festgestellt werden kann. Die Interpretation von Qualitätsindikatoren befindet sich im Verfahren zur Evaluation. Prinzipien werden im Rahmen des Aufbaus der Wissensbasis aufgenommen.

Im Wissensmodell werden Qualitätsmerkmale und -teilmerkmale abgelegt. Zu jedem Qualitätsmerkmal werden die Qualitätsteilmerkmale abgelegt, aus welchen sich das jeweilige Qualitätsmerkmal zusammensetzt. Zu jedem Qualitätsteilmerkmal können wiederum Qualitätsteilmerkmale abgelegt werden. In Tabelle 4.13 ist beschrieben, wie Qualitätsteilmerkmale in POSAAM hinterlegt werden.

Tabelle 4.13.: Im Wissensmodell hinterlegt: Qualitäts(teil)merkmal

Merkmal	Qualitäts(teil)merkmal
Information, die hinterlegt wird	Zu jedem Qualitätsmerkmal oder Qualitätsteilmerkmal können ein oder mehrere Qualitätsteilmerkmale abgelegt werden.
Form, in der die Information hinterlegt wird	Es wird eine Definition des Qualitäts(teil)merkmal oder die Referenz auf eine anerkannte Definition des Qualitäts(teil)merkmals in natürlicher Sprache hinterlegt.

In Abbildung 4.7 wird ein Überblick über die Struktur des Wissensmodells und der Querverweise innerhalb des Wissensmodells gegeben.

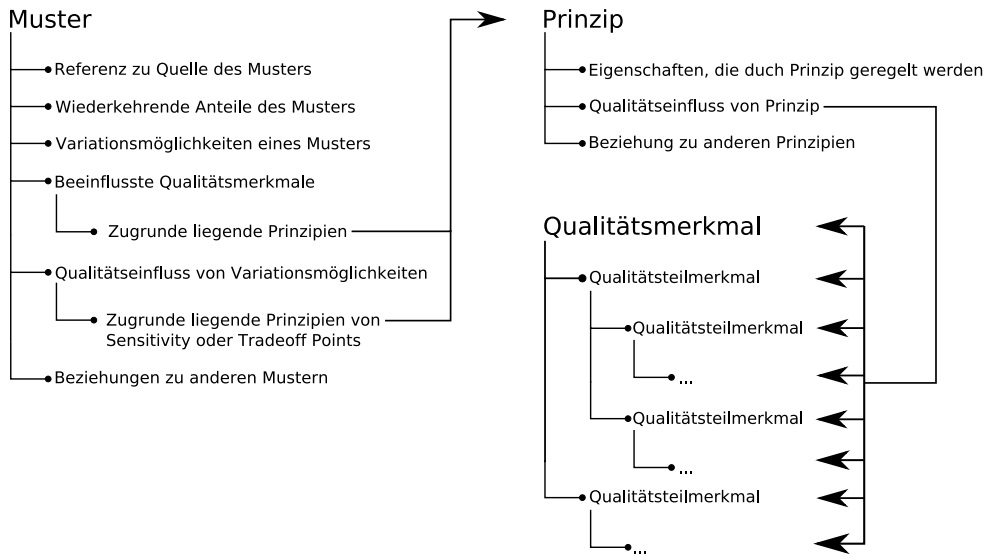

Abbildung 4.7.: Überblick über Struktur und Querverweise des POSAAM-Wissensmodells

4.3. Vorzunehmende Prüfungen

Voraussetzung für eine Evaluation nach POSAAM ist das Vorliegen einer Anforderungsspezifikation sowie einer Architekturspezifikation. Eine Evaluation nach POSAAM beginnt deshalb mit einer Prüfung der beiden Spezifikationen, um sicherzustellen, dass diese für eine Evaluation nach POSAAM ausreichend sind. In den folgenden beiden Abschnitten (4.3.1 und 4.3.2) wird jeweils beschrieben, wie eine Anforderungs- bzw. Architekturspezifikation gestaltet sein müssen, um bei einer Evaluation nach POSAAM verwendet werden zu können und wie der Prüfungsprozess zu Anfang einer POSAAM Evaluation durchgeführt wird.

4.3.1. Prüfung der Anforderungsspezifikation

Qualität definiert sich über festgelegte oder vorausgesetzte Erfordernisse (siehe Definition und Diskussion zu Qualität in Abschnitt 2.1 auf den Seiten 12ff.). Im Bereich der Softwareentwicklung stellen Anforderungen die Erfordernisse an ein Softwaresystem dar. Ergo sind Anforderungen notwendig, um die Qualität

eines Systems feststellen zu können. Die Architektur ist eine Eigenschaft eines Systems, die die Erfüllung von Anforderungen beeinflussen kann. Um die Qualität der Architektur (im Hinblick auf die Qualität des Systems, welches diese Architektur hat) feststellen zu können, werden demnach die Anforderungen, die an das System gestellt werden, benötigt. In POSAAM wird daher gefordert, dass vor Beginn der Evaluation eine Anforderungsspezifikation vorliegt. Im Folgenden wird beschrieben, in welcher Form die Anforderungen vorliegen müssen. Anschließend wird gezeigt, wie im POSAAM-Prozess eine Prüfung der Anforderungsspezifikation vorzunehmen ist.

Form der Anforderungen für eine Evaluation nach POSAAM

Die wichtigste Eigenschaft, die eine Anforderung erfüllen muss, damit sie bei einer POSAAM-Evaluation Verwendung finden kann, ist, dass die Anforderung einem Qualitätsmerkmal oder Qualitätsteilmerkmal zuzuordnen ist. Über die Zuordnung zu Qualitätsmerkmalen geschieht in POSAAM die Zuordnung zu Mustern.

Eine weitere wichtige Eigenschaft, die eine Anforderung für die Anwendung von POSAAM erfüllen muss, ist, dass die durch die Anforderung geforderte Ausprägung eines Qualitätsmerkmals möglichst präzise und umfassend beschrieben wird. Zwar ist durch Betrachtung einer Architekturspezifikation ohne eine Betrachtung der darunter liegenden Implementierung keine Aussage über die genaue Ausprägung eines Qualitätsmerkmals möglich, aber eine Anforderung, in welcher eine gewünschte Ausprägung eines Qualitätsmerkmals unscharf beschrieben ist, macht eine Evaluation der Architektur schwierig, da die Kombination der unscharf geforderten Ausprägung des Qualitätsmerkmals aus der Anforderung und der unscharfen Aussage über die Ausprägung des Qualitätsmerkmals, die aus der Architekturevaluation gewonnen werden kann, evtl. keine Aussage über der Eignung der Architektur, die geforderte Qualität zu erreichen, mehr zulässt.

Eine scharfe Formulierung einer geforderten Ausprägung eines Qualitätsmerkmals beinhaltet auch Informationen über die Umstände, unter denen die Ausprägung des Qualitätsmerkmals erreicht werden sollen. Diese Informationen können bei einer Architekturevaluation hilfreich sein, da sie u.U. eine Einschränkung der Evaluation auf Teile der Architektur ermöglichen. Im Falle einer Architekturevaluation nach POSAAM können diese Informationen auch die Auswahl geeigneter Muster für spezifische Problemlösungen einschränken und dadurch die Evaluation erleichtern bzw. die Aussagekraft einer Evaluation erhöhen.

Auch eine Priorisierung von Anforderungen kann eine Architekturevaluation nach POSAAM erleichtern. Die Auswahl eines geeigneten Musters sowie die

Prüfung der getroffenen architekturrelevanten Entscheidungen bezüglich einer Musterkonfiguration ist von den Prioritäten der Anforderungen untereinander abhängig.

Aufgrund der Notwendigkeit einer präzisen und umfassenden Beschreibung einer Anforderung wird in POSAAM gefordert, dass alle Anforderungen aus der gegebenen Anforderungsspezifikation in die Form von Szenarien wie sie *Bass et al.* in [BCK03] vorschlagen überführt werden können. Es muss möglich sein aus den Beschreibungen der gegebenen Anforderungsspezifikation alle Elemente eines Szenarios zu belegen. Durch die Spezifikation von Anforderungen mit der Hilfe von Szenarien wird eine spezifische Beschreibung der Ausprägung des Qualitätsmerkmals und der Umstände unter denen eine Ausprägung eines Qualitätsmerkmals gefordert wird, ermöglicht.

Da für die Durchführung einer POSAAM-Evaluation die Anforderungen in schriftlicher Form vorausgesetzt werden und somit keine Interaktion mit den Stakeholdern des Systems vorgesehen ist, ist es wichtig, dass die schriftliche Form der Anforderungen dem Evaluationsteam einen guten Überblick über das zu erstellende System gibt. Dazu ist es zunächst notwendig, dass das Evaluationsteam ein Verständnis für den allgemeinen Zweck des zu erstellenden Systems erhält. *Bass et al.* bezeichnen diesen auch als die Hauptgeschäftstreiber des Systems. Sind dem Evaluationsteam die Hauptgeschäftstreiber klar, kann jede Anforderung auch zu den Hauptgeschäftstreibern in Beziehung gebracht werden und so vom Evaluationsteam richtig eingeordnet werden. In POSAAM wird daher gefordert, dass die Hauptgeschäftstreiber als Teil der Anforderungsspezifikation mitgeliefert werden.

Eine Priorisierung der Anforderungen sowie eine Zuordnung der Anforderungen zu Qualitätsmerkmalen wird in POSAAM nicht explizit gefordert. Kann eine Anforderung nicht einem Qualitätsmerkmal zugeordnet werden, stellt sich die Frage, ob es sich bei der Anforderung um eine architekturrelevante Anforderung handelt. Ist dies der Fall, handelt es sich bei der Anforderung u.U. nicht um eine Qualitätsanforderung, sondern um eine Einschränkung. Diese werden in POSAAM nicht hinterfragt (bzw. evaluiert), sondern als gegeben hingenommen[2]. Handelt es sich bei der Anforderung jedoch tatsächlich um eine Qualitätsanforderung, wird das Qualitätsmerkmal und zugrunde liegende Prinzipien dieses Qualitätsmerkmals in POSAAM entsprechend ergänzt.

[2] Die Erhebung von Einschränkungen sind Teil des Requirement Engineerings und sollten daher auch in dieser Disziplin betrachtet werden.

Prüfung der Anforderungsspezifikation in POSAAM

In Abbildung 4.8 wird eine Übersicht über die bei der Prüfung der Anforderungsspezifikation durchzuführenden Aktionen gegeben.

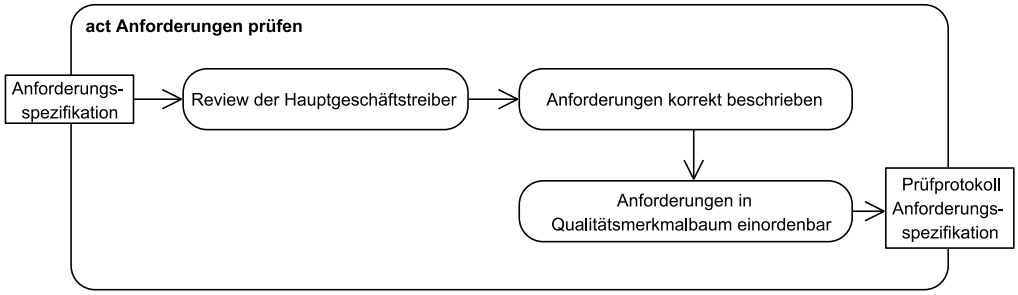

Abbildung 4.8.: Schritte zur Prüfung der Anforderungsbeschreibungen

Review der Hauptgeschäftstreiber. Die Prüfung der Anforderungsspezifikation beginnt mit der Prüfung der Beschreibung der Hauptgeschäftstreiber. Dieser erste Schritt entspricht einem Review. Das Evaluationsteam prüft, ob die gelieferten Beschreibungen verständlich sind und nimmt offene Fragen in das Prüfprotokoll auf. Bei der Prüfung der Hauptgeschäftstreiber muss deutlich werden, zu welchen primären Zwecken das Softwaresystem erstellt werden soll. Es ist herauszuarbeiten, welcher Ist-Stand vor Einsatz des zu entwickelnden Systems existiert und wie der Soll-Stand nach Einsatz des zu entwickelnden Systems aussehen soll. Das Prüfprotokoll enthält allgemeine Fragen und Aussagen zu den Hauptgeschäftstreibern, Fragen und Aussagen bezüglich des Ist-Stands sowie Fragen und Aussagen bezüglich des Soll-Stands. Sind die Hauptgeschäftstreiber nicht ersichtlich, kann keine POSAAM-Evaluation erfolgen.

Anforderungen korrekt beschrieben. Nach der Prüfung der Hauptgeschäftstreiber erfolgt eine Prüfung, ob die Anforderungen korrekt beschrieben werden. Für POSAAM müssen die Anforderungen so vorliegen, dass sie in Szenarien überführt werden können. Da in einer POSAAM Evaluation auf Qualität geprüft wird, ist die geforderte Darstellung der Anforderungen nur für Qualitätsanforderungen erforderlich. Deshalb werden die Anforderungen zunächst in architekturrelevant oder nicht architekturrelevant (siehe Definitionen in Abschnitt 3.3.5 auf Seite 54) und anschließend in Qualitätsanforderung oder Einschränkung kategorisiert. Die Kategorisierung wird im Prüfprotokoll eingetragen.

Im Anschluss an die Kategorisierung wird geprüft, ob die Qualitätsanforderungen in die Form von Szenarien überführt werden können, d.h. ob für je-

des Szenario alle geforderten Elemente (siehe Beschreibung von Szenarien auf Seite 44) beschrieben werden können. Mängel werden im Prüfprotokoll notiert. Sind die Qualitätsanforderungen nicht so beschrieben, dass sie in die Form von Szenarien überführt werden können, kann keine POSAAM-Evaluation erfolgen. Liegen andere Mängel bezüglich der Beschreibungen Anforderungen vor, muss das Evaluationsteam entscheiden, ob mit den vorliegenden Anforderungen eine POSAAM-Evaluation durchzuführen ist.

Anforderungen in Qualitätsmerkmalbaum einordenbar. Zuletzt werden die Anforderungen zu den Qualitätsmerkmalen in Bezug gebracht. Jede Anforderung wird einem Qualitätsmerkmal bzw. -teilmerkmal im Qualitätsmerkmalbaum aus der Wissensbasis zugeordnet. Im Prüfprotokoll wird festgeschrieben zu welchem Qualitäts(teil)merkmal jede Anforderung zugeordnet wurde. Ist es nicht möglich, eine Anforderung einem Qualitäts(teil)merkmal zuzuordnen, handelt es sich bei der Anforderung entweder um eine Einschränkung oder die Anforderung adressiert ein Qualitätsmerkmal, welches noch nicht in der Wissensbasis aufgenommen wurde. Im letzteren Fall wird das Qualitätsmerkmal in die Wissensbasis aufgenommen. Handelt es sich bei der Anforderung um eine Einschränkung, wird die Kategorisierung aus dem ersten Schritt der Prüfung überarbeitet.

Das Prüfprotokoll der Prüfung der Anforderungsspezifikation enthält einen Teil, in welchem beschrieben wird, ob eine POSAAM-Evaluation auf Grundlage der vorliegenden Anforderungsspezifikation für möglich gehalten wird. Die Argumentation basiert dabei auf den vorherigen Prüfungen.

4.3.2. Prüfung der Architekturspezifikation

Bei einer Architekturevaluation nach POSAAM wird, anders als in vergleichbaren Architekturevaluationsmethoden, das Vorliegen einer Architekturspezifikation zum Zeitpunkt der Prüfung voraus gesetzt. Um eine POSAAM-Evaluation mit Hilfe dieser Architekturspezifikation durchführen zu können, muss die Architekturspezifikation einige Voraussetzungen erfüllen.

Analog zur Prüfung der Anforderungsspezifikation wird im Folgenden beschrieben, in welcher Form die Architekturspezifikation vorliegen muss. Anschließend wird gezeigt, wie im POSAAM-Prozess eine Prüfung der Architekturspezifikation vorzunehmen ist.

Form der Architekturspezifikation für eine Evaluation nach POSAAM

Eine POSAAM-Evaluation wird anhand der dokumentierten Form der Architekturspezifikation durchgeführt. Deshalb muss die Architekturspezifikation einem Mindestmaß an Qualität genügen. Dabei geht es nicht um die Qualitätseigenschaften, die ein durch die spezifizierte Architektur beschriebenes System vorweist, sondern um die Qualität der Beschreibungen der Architektur. Für eine POSAAM-Evaluation sind insbesondere die folgenden Merkmale einer Architekturbeschreibung von Bedeutung:

- Die Beschreibungen müssen verständlich sein. Die grundsätzliche Funktionsweise der Architektur muss für das Evaluationsteam ersichtlich sein.
- Es muss für das Evaluationsteam möglich sein, die Anwendung von Mustern in der Architekturbeschreibung zu erkennen.
- Es muss für das Evaluationsteam möglich sein, die Anwendung von Prinzipien in der Architekturbeschreibung zu erkennen.

Verständlichkeit der Architektur. Um die Verständlichkeit der Architekturbeschreibungen zu gewährleisten, muss die Funktionsweise des Systems durch die Darstellungen der Architekturspezifikation ersichtlich werden. In POSAAM wird dazu gefordert, dass in der Architekturspezifikation gezeigt wird, wie die in der Anforderungsspezifikation erläuterten Hauptgeschäftsziele erreicht werden. Zu diesem Zweck wird eine logische Sicht auf die Architektur (wie von *Beneken* beschrieben [Ben08, Kapitel 3, Abschnitt 3.3]) gefordert. Diese Sicht ist insbesondere losgelöst von technischen Implementierungsdetails und zeigt auf welche Funktionen von welchen Anteilen des Systems wahrgenommen werden. Je nach Größe des zu erstellenden Systems kann die logische Sicht in mehrere Hierarchieebenen untergliedert sein.

Weiterhin werden mindestens eine Sicht, die den statischen Aufbau des Systems darstellt (eine Sicht aus dem „Module Viewtype“ nach *Clements et al.* [CBB+02]; z.B. die technische Sicht, wie von *Beneken* in [Ben08, Kapitel 3, Abschnitt 3.3] beschrieben), sowie eine Sicht, die die Dynamik des Systems darstellt (eine Sicht aus dem „Component and Connector Viewtype“ nach *Clements et al.* [CBB+02]; z.B. die Laufzeitsicht, wie von *Beneken* in [Ben08, Kapitel 3, Abschnitt 3.3] beschrieben), verlangt. In der Sicht, in welcher die Dynamik des Systems dargestellt wird, ist auch die Systemgrenze und die Interaktion des Systems mit dessen Umwelt bzw. mit Aktoren aus dessen Umwelt darzustellen.

Aus Gründen der Verständlichkeit und Nachvollziehbarkeit wird in POSAAM auch gefordert, dass eine Architekturspezifikation konform zur IEEE 1471 [IEE00]

ist. In dieser werden z.B. auch Beschreibungen der Standpunkte verlangt, nach denen die Sichten aufgestellt werden. Auch weitere Bestimmungen der IEEE 1471 wirken sich positiv auf die Verständlichkeit der Architekturbeschreibungen aus.

Identifikation von Mustern und Prinzipien. Für eine POSAAM-Evaluation ist es nötig, dass das Evaluationsteam die Verwendung von Mustern in der Architekturspezifikation erkennen kann. Dazu ist es nötig, dass nicht nur die Struktur (also die Elemente bzw. Bausteine, deren Eigenschaften und Beziehungen zueinander), sondern auch der Sinn der Bausteine und ihrer Form der Interaktion hinterlegt ist. Im Umfeld der Pattern-Community wird dies oft als die Verantwortlichkeit der Bausteine bezeichnet. Eine Beschreibung der Verantwortlichkeiten der Elemente einer Architektursicht und des Sinns der Interaktionen zwischen den Elementen kann sehr pragmatisch geschehen. Eine Form der Beschreibung, ähnlich der der CRC-Karten [BC89], bei der für jedes Element beschrieben wird, welche Aufgabe es innerhalb der Architektur in Koordination mit welchen anderen Elementen wahrnimmt, ist hierfür ausreichend. Die Beschreibung der Verantwortlichkeiten geschieht für die Elemente aller geforderten Sichten (logischer, technischer und Laufzeitsicht).

Um in einer POSAAM-Evaluation prüfen zu können, ob in einem Teilbereich der Architektur das korrekte Muster eingesetzt wurde und ob das eingesetzte Muster korrekt konfiguriert wurde, muss ersichtlich sein, aus welchem Grund eine vorliegende Architektur gewählt wurde. Deshalb wird zusätzlich zu den Beschreibungen in Form von Sichten und zusätzlich zur Beschreibung des Sinns der Bausteine und ihrer Form der Interaktion gefordert, dass Begründungen über architekturrelevante Entscheidungen (siehe Definition in Abschnitt 3.3.5 auf Seite 54) hinterlegt werden, weshalb eine Form der Strukturierung gewählt wurde. Eine informelle Beschreibung der architekturrelevanten Entscheidungen ist für die Zwecke von POSAAM ausreichend. Wichtig ist dabei, dass die möglichen Alternativen, die bei der Entscheidung möglich gewesen wären, aufgezeigt werden und eine Begründung für die gewählte Alternative erfolgt. Für eine detaillierte und systematische Beschreibung der architekturrelevanten Entscheidung wird eine Beschreibung, wie sie *Tyree und Akerman* in [TA05] erarbeiten, vorgeschlagen.

In der Beschreibung der architekturrelevanten Entscheidungen kann auf die Anforderungen, die an das System gestellt werden, für welches die Architektur konzipiert ist, eingegangen werden. Da ein Tracing der Anforderungen in der Architekturspezifikation in der industriellen Praxis jedoch noch nicht zum Stand der Technik gehört, wird für eine POSAAM-Evaluation nicht gefordert, dass die Architekturspezifikation ein solches Tracing beinhaltet. Bei der Prüfung der Beschreibungen der Architekturspezifikation wird festgestellt, inwieweit sich die ar-

chitekturrelevanten Entscheidungen auf die Anforderungen zurückführen lassen. Im Rahmen der POSAAM-Evaluation werden diese Entscheidungen überprüft und Lücken zwischen Anforderungen und Architekturspezifikation geschlossen.

Um die Anwendung von Prinzipien in Beschreibungen von Softwarearchitekturen erkennen zu können, müssen in den Beschreibungen die Anteile sichtbar werden, die zum einen die Anwendung von Prinzipien beeinflussen und zum anderen durch die Anwendung von Prinzipien beeinflusst werden. Im Wissensmodell von POSAAM (siehe Abschnitt 4.2.2) wird definiert, dass die Eigenschaften, die den Einsatz eines Prinzips beeinflussen oder die durch den Einsatz eines Prinzips beeinflusst werden, in einer POSAAM-Wissensbasis abgelegt werden. Für das Qualitätsmerkmal der Skalierbarkeit existiert ein Prinzip, das besagt, dass die Ressourcen, die durch das System bei einer entsprechenden Skalierung (z.B. Erhöhung der Anzahl der Nutzer, die das System zeitgleich in Anspruch nehmen) belastet werden, in einer Form vom Rest des Systems „abgekoppelt“ werden müssen, die es erlaubt, die Ressourcen zu vervielfältigen, ohne dass dies einen Einfluss auf die Gestaltung des restlichen Systems hätte. In diesem Fall sind die Eigenschaften, die die Anwendung des Prinzips beeinflussen, die Belastung von Ressourcen bei zu erwartenden Skalierungen (z.B. Erhöhung der Anzahl der Nutzer, die das System zeitgleich in Anspruch nehmen) des Systems. Die Eigenschaften, die durch die Anwendung des Prinzips beeinflusst werden, sind in diesem Fall, die „Kopplung“ der Ressourcen an den Rest des Systems.

Sichten, die Eigenschaften von Prinzipien, welche die in den Anforderungen geforderten Qualitätsmerkmale positiv beeinflussen, wiedergeben, sind bei einer POSAAM-Evaluation hilfreich. Da aber auch in diesem Fall gilt, dass das vorliegen solcher Sichten in der industriellen Praxis nicht üblich ist, wird aus Gründen des Pragmatismus in POSAAM auf solche Sichten verzichtet. Eine Prüfung auf das Vorliegen geeigneter Sichten, um den Einsatz von Prinzipien nachvollziehen zu können, wird in POSAAM dennoch durchgeführt. Liegen entsprechende Sichten vor, können diese bei der Evaluation hinzugezogen werden. Ein Beispiel für eine solche Sicht ist die Verteilungssicht, da bei dieser Sicht die knappen Ressourcen (Kommunikationsverbindungen sowie Prozessoren), die zu Engpässen bei der Laufzeiteffizienz führen können, sichtbar werden.

In POSAAM wird nicht gefordert, dass die Architekturspezifikation einem evtl. bereits implementierten System entsprechen muss. Obwohl dies offensichtlich sinnvoll ist, würde eine Prüfung dieser Forderung nicht in POSAAM passen. In POSAAM wird vorausgesetzt, dass die zu prüfende Architekturspezifikation einer bereits existierenden oder einer noch durchzuführenden Implementierung entspricht.

Prüfung der Architekturspezifikation in POSAAM

In Abbildung 4.9 wird ein Überblick über die bei der Prüfung der Architekturspezifikation durchzuführenden Aktionen gegeben.

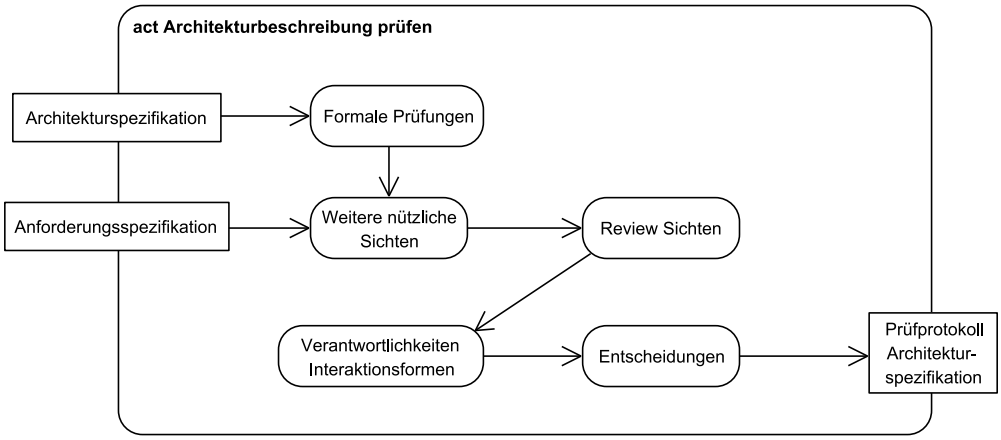

Abbildung 4.9.: Schritte zur Prüfung der Architekturbeschreibungen

Formale Prüfungen. Die Prüfung beginnt damit, dass formale Anforderungen an die Architekturspezifikation sicher gestellt werden. Zu den formalen Anforderungen gehören z.B. das Vorhandensein der drei geforderten Sichten der Architektur (eine logische Sicht, eine Sicht der Statik sowie eine der Dynamik des Systems), die Darstellung der Hauptgeschäftstreiber über die Architektur,die Darstellung der Systemgrenzen sowie die Konformität zur IEEE 1471. Sind formale Anforderungen an die Form der Beschreibung der Architekturspezifikation nicht erfüllt, kann eine POSAAM-Evaluation nicht stattfinden. Die Resultate der formalen Prüfungen werden im Prüfprotokoll zur Prüfung der Architekturspezifikation festgehalten.

Weitere nützliche Sichten. Aus der Betrachtung der Qualitätsanforderungen können Erkenntnisse über weitere, für die Evaluation nutzbare Sichten gewonnen werden. Dazu werden die Prinzipien, die die gestellten Qualitätsanforderungen beeinflussen, aus der POSAAM-Wissensbasis gewonnen (siehe Abschnitt 4.2.2; insbesondere Merkmal „Qualitätseinfluss von Prinzip“). Über das Merkmal „Eigenschaften, die durch Prinzipien geregelt werden“ kann dann festgestellt werden, welche Eigenschaften in Sichten hinterlegt sein müssen, um die Anwendung der Prinzipien zu erkennen. Liegen diese Sichten nicht vor, so wird im Prüfprotokoll vermerkt, dass für genauere Aussagen über die jeweiligen Qualitätsanforderungen die Sichten hilfreich wären. Eine Begründung, die über die Kopplung

der Qualitätsanforderungen zu den Prinzipien und dessen Eigenschaften zu den Sichten führt, wird dem Vermerk beigefügt. Eine POSAAM-Evaluation kann auch ohne das Vorhandensein solcher Sichten stattfinden.

Review der Sichten. Sind alle Prüfungen auf das Vorliegen von Sichten abgeschlossen, werden die Sichten inhaltlich geprüft. Dafür werden Reviews durchgeführt. Besondere Eigenschaften, die bei einem Review geprüft werden, sind die Prüfung der konzeptuellen Integrität innerhalb und zwischen den Sichten. Zur Prüfung der Konsistenz zwischen den Sichten können die Nutzungsszenarien des Systems zur Hilfe genommen werden, mit deren Hilfe durch die Elemente der Sichten navigiert wird (ähnlich dem Vorschlag der „+1"-Sicht von *Kruchten* [Kru95]). Fehler, die beim Review der Sichten festgestellt werden, werden ins Prüfprotokoll aufgenommen. Ob eine POSAAM-Evaluation dennoch stattfinden kann, wird vom Evaluationsteam entschieden und im Prüfprotokoll begründet.

Verantwortlichkeiten und Interaktionsformen. Die für die Identifikation von Mustern in der Architekturbeschreibung wichtige Angabe von Verantwortlichkeiten der Elemente und Form der Interaktion zwischen den Elementen zur Erfüllung von Aufgaben wird im Anschluss an den Review der Sichten geprüft. Erneut entspricht diese Prüfung einem Review. Das Evaluationsteam stellt sicher, dass für die in den Sichten beschriebenen Elemente verständlich ist, welche Aufgabe die Elemente erfüllen und wie (durch welche Interaktionen mit weiteren Elementen) diese Aufgaben erfüllt werden. Kann für Elemente nicht nachvollzogen werden, welche Verantwortlichkeit diese haben oder wenn Unklarheiten bezüglich der Verantwortlichkeiten auftreten, werden entsprechende Einträge im Prüfprotokoll vorgenommen. Ob eine POSAAM-Evaluation dennoch stattfinden kann, wird vom Evaluationsteam entschieden und im Prüfprotokoll begründet. Können für einen Großteil der Elemente keine Verantwortlichkeiten zugeordnet werden, ist eine POSAAM-Evaluation nicht möglich.

Entscheidungen. Auch die Prüfung der Beschreibung von architekturrelevanten Entscheidungen geschieht durch die Durchführung eines Reviews. Dabei stellt das Evaluationsteam in erster Linie sicher, dass die bestehende Architektur durch Entscheidungen begründet wird und dass für alle Entscheidungen Alternativen aufgeführt werden und begründet wird, weshalb die Entscheidung für die ausgewählte Alternative getroffen wurde. Sind wichtige Bestandteile der Architektur nicht auf entsprechende Entscheidungen zurückzuführen, wird dies im Prüfprotokoll vermerkt. Ebenfalls wird im Prüfprotokoll vermerkt, für welche Entscheidungen keine Alternativen aufgeführt werden oder die Entscheidung

für eine Alternative nicht ausreichend begründet ist. Zudem wird im Prüfprotokoll vermerkt, ob die Begründungen für Entscheidungen auf die Anforderungen zurückzuführen sind und welche Anforderungen nicht durch Entscheidungen begründet werden.

Analog zum Prüfprotokoll für die Prüfung der Anforderungsspezifikation enthält auch das Prüfprotokoll für die Prüfung der Architekturspezifikation einen Teil, in welchem Beschrieben wird, ob eine POSAAM-Evaluation auf Grundlage der vorliegenden Architekturspezifikation für möglich gehalten wird. Die Argumentation basiert dabei auf den vorherigen Prüfungen.

4.4. Durchführen der Evaluation

Nach den Prüfungen der Anforderungs- und Architekturspezifikation kann mit den Tätigkeiten begonnen werden, bei denen geprüft wird, wie die Anforderungen in der Architektur berücksichtigt wurden. Dazu wird in einem Zyklus jede Anforderung einzeln betrachtet. Der Zyklus, der jeweils für eine Anforderung durchlaufen wird, stellt den Hauptteil der POSAAM-Evaluation dar. Der Verlauf der Prüfung in diesem Zyklus wird in Abschnitt 4.4.1 dargelegt.

Zwei der Aktionen in diesem Zyklus werden in weiteren Abschnitten detaillierter betrachtet. In Abschnitt 4.4.2 wird auf die Prüfung von identifizierten Mustern eingegangen und auf die Identifikation alternativer Architekturmechanismen zum Beeinflussen eines Qualitätsmerkmals in Abschnitt 4.4.3.

4.4.1. Hauptzyklus der POSAAM-Evaluation

In Abbildung 4.10 wird ein Überblick über die Aktionen gegeben, die im Rahmen des Zyklus für jede Anforderung durchlaufen werden.

Drei der Aktionen in Abbildung 4.10 wurden zu Anfang dieses Kapitels im Rahmen der Übersicht über die zentralen Konzepte (siehe Abschnitt 4.1.2 auf Seite 68 bzw. Abbildung 4.2) bereits angesprochen. Die Aktionen werden nun im Kontext dargelegt und erläutert.

Mustermenge zur Umsetzung der Anforderung ableiten. Der Zyklus beginnt mit der Betrachtung einer einzelnen architekturrelevanten Anforderung aus der vorliegenden Anforderungsspezifikation. Eingaben für die erste Aktion sind die aktuell betrachtete Anforderung sowie die POSAAM-Wissensbasis. In der ersten Aktion wird aus der Wissensbasis eine Menge von Mustern gewonnen, die zur Umsetzung der aktuell betrachteten Anforderung (in jedem Durchlauf des Zyklus

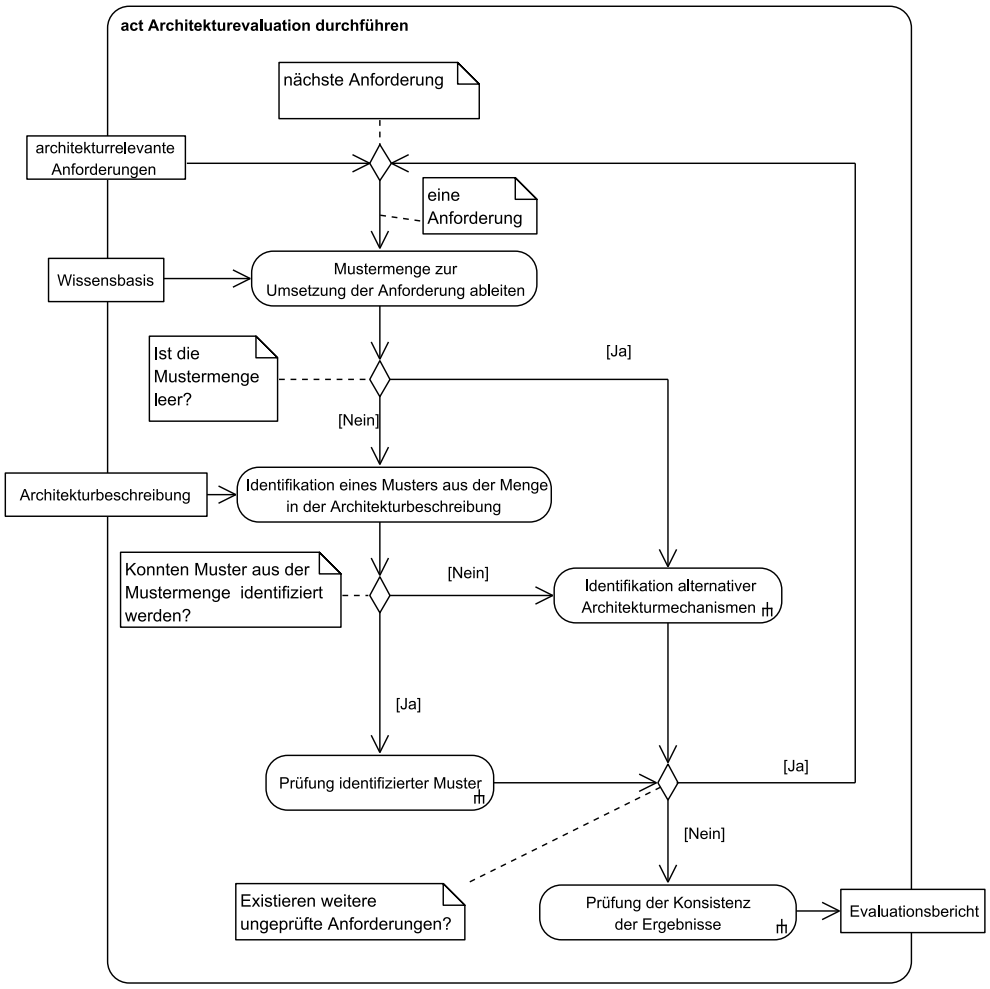

Abbildung 4.10.: Überblick über die Schritte zur Evaluation der Architektur

wird eine andere Anforderung betrachtet) geeignet sein könnten. Dazu wird festgestellt, welches Qualitätsmerkmal durch die Anforderung gefordert wird. Dies ist immer möglich, da bei der Prüfung der Anforderungen jede Anforderung einem Qualitätsmerkmal oder -teilmerkmal zugeordnet wurde (siehe Abschnitt 4.3.1). Wenn bekannt ist, welches Qualitäts(teil)merkmal durch die Anforderung gefordert wird, werden aus der Wissensbasis alle Muster selektiert, mit denen das Qualitäts(teil)merkmal in der gegebenen Architektur beeinflusst werden kann. Dazu wird das Merkmal „Beeinflusste Qualitätsmerkmale“ des Wissensmodells und die Beschreibungen zur vorliegenden Anforderung verwendet. Die aus der Selektion resultierende Menge von Mustern ist die Ausgabe der ersten Aktion.

Ist die Menge der Muster leer, wird mit der Aktion „Identifikation alternativer

Architekturmechanismen“ fortgefahren, ansonsten mit der Aktion „Identifikation eines Musters aus der Menge in der Architekturbeschreibung“.

Identifikation eines Musters aus der Menge in der Architekturbeschreibung. Die Menge von Mustern, die in der Aktion „Mustermenge zur Umsetzung der Anforderungen ableiten“ erzeugt wird, ist eine Eingabe für diese Aktion. Weitere Eingaben für diese Aktion sind die Architekturspezifikation sowie die aktuell betrachtete Anforderung.

In dieser Aktion ist es Aufgabe des Evaluationsteams, die Muster aus der Eingabemenge von Mustern in der Architekturspezifikation zu identifizieren. Dazu wird das Merkmal „Wiederkehrende Anteile eines Musters“ aus dem POSAAM-Wissenmodell verwendet. Das Evaluationsteam sucht nach Elementen mit den zu den wiederkehrenden Anteilen des Musters korrespondierenden Verantwortlichkeiten. Die aktuell betrachtete Anforderung kann dabei verwendet werden, um den „Ort“ ausfindig zu machen, an dem das Muster zum Einsatz kommen kann. Mit der Bezeichnung „Ort“ wird kein räumlicher Ort bezeichnet, sondern ein Teil der Architektur. Es gibt Qualitätsanforderungen, die an funktionale Anforderungen gekoppelt sind. Bei diesen Anforderungen kann das Evaluationsteam die Anforderung verwenden, um festzustellen, welche Teile der Architektur an der Erfüllung der funktionalen Anforderung beteiligt sind. Dadurch kann das Evaluationsteam den Teil, in welchem ein Architekturmuster zum Einsatz kommen kann, einschränken. Auch bei Qualitätsanforderungen, die nicht an eine funktionale Anforderung gekoppelt sind (z.B. Anforderungen an die Wartbarkeit der Anwendung), können über die Anforderungen ggf. Informationen über die in der Architektur betroffenen Anteile gewonnen werden.

Die Ausgabe dieser Aktion ist die Menge von Mustern, die als Eingabe übergeben wurde, zusammen mit Annotationen, welches Muster in der Architekturspezifikation gefunden werden konnte und welche Elemente der Architekturspezifikation die durch das Muster vorgeschriebenen Verantwortungen wahrnehmen. Die Muster, die nicht in der Architekturspezifikation identifiziert werden konnten, werden in der Menge weitergeführt, da sie in der Aktion „Prüfung identifizierter Muster“ noch benötigt werden. Konnten keine Muster aus der eingegebenen Mustermenge in der Architekturbeschreibung identifiziert werden, wird mit der Aktion „Identifikation alternativer Architekturmechanismen“ fortgefahren.

Die Suche nach Mustern in der Architekturspezifikation kann nicht automatisiert erfolgen, da für eine Automatisierung sowohl die Architekturspezifikation als auch die Musterbeschreibungen in einer maschinenlesbaren Repräsentation vorliegen müssten. Bereits diese Voraussetzung ist in der industriellen Praxis unüblich, aber selbst dann wäre eine Automatisierung der Suche nur möglich,

wenn für die Musterbeschreibungen ausreichende Informationen für eine eindeutige Identifikation der Muster in maschinenlesbarer Form vorlägen. Da sich manche Muster in ihrer statischen Struktur stark ähneln oder gar gleich sind, würde eine Repräsentation durch Elemente und Topologie, wie sie üblich ist, nicht ausreichen, um Muster von anderen Mustern oder von der Architektur eines beliebigen Teils des Systems unterschieden zu können. Wenn die dynamische Form der Interaktion zwischen den Elementen bei der Automatisierung der Erkennung hinzugezogen würde (wie es z.B. *Pettersson* in [Pet05] versucht), müsste dazu in der verfügbaren Repräsentation die Art der Interaktion zwischen den Elementen enthalten sein. Selbst dann, würde diese Information immer noch nicht ausreichen, um Muster eindeutig erkennen zu können, da erst der Zweck/die Verantwortlichkeiten die Muster eindeutig differenzieren. Diese Informationen maschinenlesbar darzustellen wäre in der industriellen Praxis aufgrund des damit verbundenen hohen Aufwands nicht praktizierbar. *Buschmann et al.* bezeichnen die Idee, Muster in Quellcode oder Modellen automatisiert erkennen zu lassen, gar als naiv [BHS07c, S. 17].

Prüfung identifizierter Muster. Diese Aktion wird immer dann durchgeführt, wenn ein Muster in der vorliegenden Architekturspezifikation identifiziert werden konnte. Eingabe für die Aktion sind das identifizierte Muster sowie die Architekturspezifikation selbst, die aktuell betrachtete sowie alle weiteren Anforderungen.

In dieser Aktion findet der Kern der Bewertung anhand von Mustern statt. Die Aktivitäten zu dieser Aktion sind deshalb in einem eigenen Abschnitt (der folgende Abschnitt 4.4.2) beschrieben. In dieser Aktion wird geprüft, ob für die aktuell betrachtete Anforderung das geeignetste Muster ausgewählt wurde, ob das Muster korrekt eingesetzt wurde, ob das Muster für die Gesamtheit der betrachteten Anforderungen korrekt konfiguriert wurde (ob die variierenden Anteile korrekt belegt wurden) und inwiefern weitere Musterabhängigkeiten zu prüfen sind. Die Resultate der Aktion werden in den Evaluationsbericht übertragen.

Nach dieser Aktion beginnt der Hauptzyklus von POSAAM erneut, vorausgesetzt es existieren weitere zu prüfende Anforderungen. Ist dies nicht der Fall, werden die im Laufe des Hauptzyklus erarbeiteten Ergebnisse auf Konsistenz geprüft.

Identifikation alternativer Architekturmechanismen. Diese Aktion wird immer dann durchgeführt, wenn eine Bewertung anhand von Mustern nicht erfolgen kann. Gründe dafür können entweder sein, dass in der Wissensbasis kein Muster für die Adressierung einer Anforderung gefunden werden konnte oder dass die Muster, die in der Wissensbasis zur Adressierung einer Anforderung gefunden

wurden, nicht in der vorliegenden Architekturspezifikation identifiziert werden konnten. Eingaben für diese Aktion sind die aktuell betrachtete Anforderung, alle weiteren Anforderungen, die Architekturspezifikation sowie ggf. die Menge von Mustern, die für die Adressierung der aktuell betrachteten Anforderung geeignet sind.

Auch diese Aktion stellt einen wichtigen Teil von POSAAM dar und wird deshalb in einem eigenen Abschnitt (4.4.3) erläutert. In der Aktion wird geprüft, ob aus der Architekturspezifikation ersichtlich ist, ob die aktuell bearbeitete Anforderung, für die kein Muster zugeordnet werden konnte, durch andere Architekturmechanismen adressiert wurde. Auch wird geprüft, ob für die Umsetzung der Anforderung kein Muster zum Einsatz hätte kommen können. In der Aktion werden auch Maßnahmen zur Pflege der Wissensbasis vorgenommen, bei welchen u.U. neue Muster oder Prinzipien in die Wissensbasis aufgenommen werden. Die Resultate der Aktion werden im Evaluationsbericht festgehalten.

Prüfung der Konsistenz der Ergebnisse. Haben alle architekturrelevanten Anforderungen den Hauptzyklus der POSAAM-Evaluation durchlaufen, werden zuletzt die Ergebnisse des Evaluationsberichts auf Konsistenz geprüft. Als Eingabe für diese Aktion werden der vorläufige Stand des Evaluationsberichts und die Architekturspezifikation verwendet.

Während des Hauptzyklus der POSAAM-Evaluation wurde zu jeder Anforderung ein Eintrag in den Evaluationsbericht vorgenommen. Dies bedeutet jedoch nicht, dass für alle Bausteine der vorliegenden Architekturbeschreibung entsprechende Eintragungen vorgenommen wurden. In dieser Aktion werden alle Architekturbestandteile der Architekturspezifikation betrachtet. Werden dabei Architekturbestandteile identifiziert, die weder einer funktionalen Anforderung zugeordnet werden können noch durch ein Muster oder ein Prinzip, welches im Evaluationsbericht erfasst wurde, benötigt werden, liegt eine Inkonsistenz vor. In diesem Fall versucht das Evaluationsteam festzustellen, ob durch den oder die Bausteine, die eine Inkonsistenz darstellen, ein Qualitätsmerkmal beeinflusst wird. Dazu nutzt das Evaluationsteam das Merkmal der „Eigenschaften, die durch ein Prinzip geregelt werden“ aus der Wissensbasis. Auch ein Gedankenspiel, bei welchem das Team die Folgen für die Architektur des Systems abzuschätzen versucht, wenn die entsprechenden Architekturbestandteile aus der Spezifikation entfernt würden, kann dazu verwendet werden. Gelingt es dem Evaluationsteam, festzustellen, welche Qualitätsmerkmale durch die inkonsistenten Architekturbestandteile adressiert werden, wird geprüft, inwieweit sich diese auf die Anforderungen auswirken. In allen Fällen wird die Inkonsistenz in den Evaluationsbericht aufgenommen. Kann die Inkonsistenz einer Anforderung zugeordnet werden, werden die Architekturbestandteile zur Inkonsistenz

im Evaluationsbericht unter dieser Anforderung als „alternative Architekturmechanismen“ aufgeführt. Kann die Inkonsistenz keiner Anforderung zugeordnet werden, wird ein entsprechender Eintrag in den Evaluationsbericht vorgenommen.

Das Vorgehen in dieser Aktion ist dem Vorgehen des Schritt sechs von ATAM ähnlich (vgl. Abschnitt 3.2 auf Seite 39). Deshalb ist für diese Aktion der Bedarf an Experten hoch. Dennoch wird durch die Zuordnung zu Prinzipien und zu den Eigenschaften, die durch diese Prinzipien geregelt werden, die Nachvollziehbarkeit im Vergleich zu ATAM erhöht.

4.4.2. Prüfung der Muster

Das Vorgehen zur Prüfung der Muster lässt sich in die vier Aktionen aus Abbildung 4.11 unterteilen.

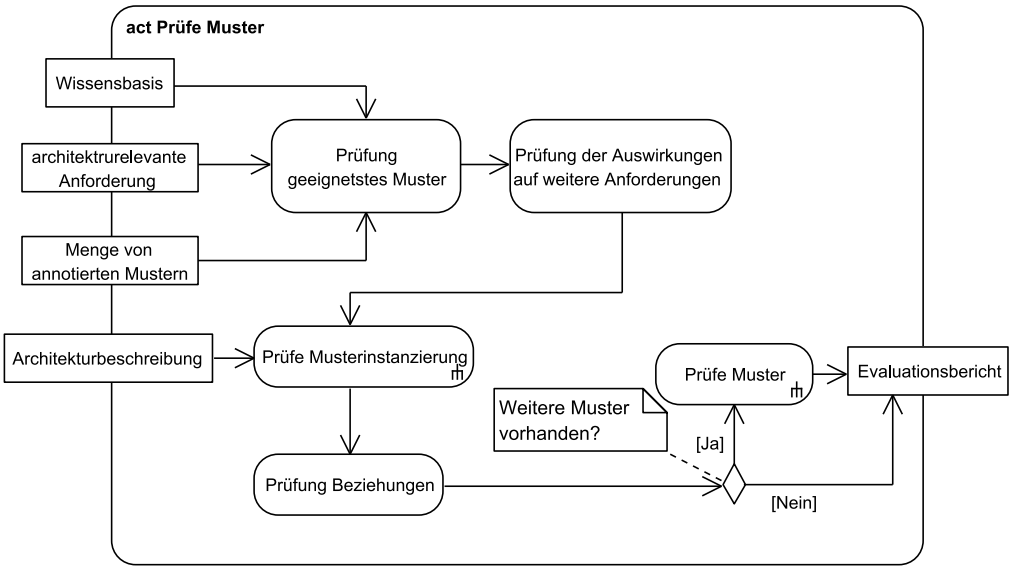

Abbildung 4.11.: Vorgehen zur Prüfung eines identifizierten Musters

Jede der Aktionen aus Abbildung 4.11 wird im Folgenden beschrieben.

Prüfung der Wahl des geeignetsten Musters. Wurde ein Muster in der Architekturspezifikation identifiziert, wird zunächst überprüft, ob das Muster sich für die jeweilige Anforderung oder das jeweilige Problem eignet und ob es ggf. geeignetere Muster gegeben hätte. Eingaben für diese Aktion sind die aktuell

betrachtete Anforderung, das zu prüfende Muster und ggf. die Menge von annotierten Mustern aus der Aktion „Identifikation eines Musters aus der Menge in der Architekturbeschreibung“.

Liegt zur Beginn der Ausführung dieser Aktion noch keine Menge von annotierten Mustern aus einer vorherigen Aktion vor, wird zunächst nach Alternativen für das zu prüfende Muster gesucht. Dabei wird das Merkmal „Beziehungen zu anderen Mustern“ aus dem Wissensmodell verwendet. Existiert eine Beziehung „verfeinert“ zu einem anderen Muster, so stellt das Muster, welches verfeinert wird (Stamm-Muster), sowie alle Muster, die das Stamm-Muster verfeinern, eine mögliche Alternative zum zu prüfenden Muster dar. Ist z.B. das zu prüfende Muster das Muster „Role-Based Access Control (RBAC)“ [SFBH+05, S. 249], sind sowohl dessen Stamm-Muster „Authorization“ [SFBH+05, S. 245] als auch weitere Verfeinerungen des Stamm-Musters (z.B. „Multilevel Security“ [SFBH+05, S. 253]) mögliche Alternativen zum zu prüfenden Muster.

Wenn die möglichen Alternativen zum zu prüfenden Muster vorliegen, wird geprüft, ob geeigentere Muster hätten ausgewählt werden können. Dazu wird der Kontext des Musters verwendet. Der Kontext von Mustern ist im POSAM-Wissensmodell nicht hinterlegt. Es wird auf den Kontext aus der Musterbeschreibung zurückgegriffen, auf die im Merkmal „Referenz zu Quelle des Musters“ verwiesen wird. Die Kontexte des zu prüfenden Musters und der möglichen Alternativen werden jeweils mit dem bestehenden Kontext verglichen. Liegt eine Diskrepanz zwischen dem bestehenden Kontext und dem Kontext des zu prüfenden Musters vor oder ist der Kontext einer möglichen Alternative passender als der Kontext des zu prüfenden Musters und ist durch die Dokumentation der Entscheidungen in der Architekturspezifikation nicht nachvollziehbar weshalb, wird ein entsprechender Eintrag im Evaluationsbericht vorgenommen. Im Eintrag wird die mögliche Alternative vorgeschlagen und der Vorschlag durch den Vergleich der Kontexte begründet.

Prüfung der Auswirkungen auf Qualitätsmerkmale. In dieser Aktion werden die Auswirkungen des identifizierten Musters auf Qualitätsmerkmale allen Qualitätsanforderungen gegenüber gestellt. Für diese Aktion wird das zu prüfende Muster benötigt.

Die Einflüsse des zu prüfenden Musters auf Qualitätsmerkmale sind im Wissensmodell unter dem Merkmal „Beeinflusste Qualitätsmerkmale“ abgelegt. Das Evaluationsteam vergleicht die Eintragungen zu den Einflüssen auf Qualitätsmerkmale mit den bestehenden Anforderungen an das System. Stehen Einflüsse des zu prüfenden Musters auf Qualitätsmerkmale zu Anforderungen des Systems im Widerspruch, wird das Muster als Risiko für die entsprechende Anforderung im Evaluationsbericht aufgenommen.

Wurden bei der Prüfung der Kontexte alternative Muster identifiziert, die sich zur Anwendung eignen würden, werden die Einflüsse der Alternativen auf das Qualitätsmerkmal, welches durch das zu prüfende Muster in negativer Weise beeinflusst wird, untersucht. Wird eine Alternative identifiziert, bei welcher keine oder geringere negative Einflüsse auf die entsprechenden Qualitätsmerkmale existieren, wird auch für dieses Muster eine Prüfung bezüglich der Anforderungen durchgeführt. Das zu prüfende Muster wird der potentiellen Alternative bezüglich aller Qualitätsmerkmale gegenüber gestellt. Sprechen keine Gründe gegen den Einsatz der potentiellen Alternative, wird die Verwendung der Alternative und die Analyse der Qualitätsmerkmale im Evaluationsbericht vorgeschlagen.

Prüfung der korrekten Instanzierung des Musters. Nach den Prüfungen zu möglichen Alternativen des Musters wird der korrekte Einsatz (die Instanzierung) des Musters geprüft. Als Eingabe zu dieser Aktion wird das zu prüfende Muster sowie die Architekturbeschreibung benötigt.

Die Prüfung der korrekten Instanzierung untergliedert sich in die Prüfung der wiederkehrenden Anteile eines Musters, die Prüfung der variierdenen Anteile eines Musters und in die Prüfung der Auswirkungen der variierdenden Anteile eines Musters auf Qualitätsmerkmale (vgl. Abbildung 4.12).

Die Prüfung der wiederkehrenden Anteile eines Musters untergliedert sich wiederum in die zwei Aktionen zur Prüfung der Elemente und die Prüfung der Topologie und Interaktionen eines Musters. Bei der ersten Aktion prüft das Evaluationsteam, ob sich alle notwendigen, wiederkehrenden Elemente des Musters in der Architekturbeschreibung wiederfinden und ob diese den Einschränkungen, die durch das Muster ggf. vorgegeben werden, entsprechen. Dazu verwendet das Evaluationsteam die Beschreibungen aus dem Merkmal „Wiederkehrende Anteile eines Musters“ aus der Wissensbasis.

Analog verlaufen die Prüfungen der Topologie der Elemente und der auf dieser Topologie aufbauenden Form der Interaktion zwischen den Elementen. Es werden jeweils die Beschreibungen aus dem Merkmal „Wiederkehrende Anteile eines Musters“ den Beschreibungen aus der Architekturspezifikation gegenüber gestellt und darauf geachtet, dass auch die Einschränkungen, die das Muster ggf. vorgibt, beachtet werden. Ist die Beschreibung der Instanzierung des Musters in der Architekturspezifikation unvollständig (da z.B. die Form der Interaktion nicht vollständig beschrieben wird), wird im Evaluationsbericht aufgenommen, welche wiederkehrenden Anteile eines Musters in der Architekturspezifikation nicht beschrieben werden. Werden Widersprüche zwischen den Beschreibungen der wiederkehrenden Anteile eines Musters und den Beschreibungen in der Architekturspezifikation festgestellt, handelt es sich um eine fehlerhafte Instanzie-

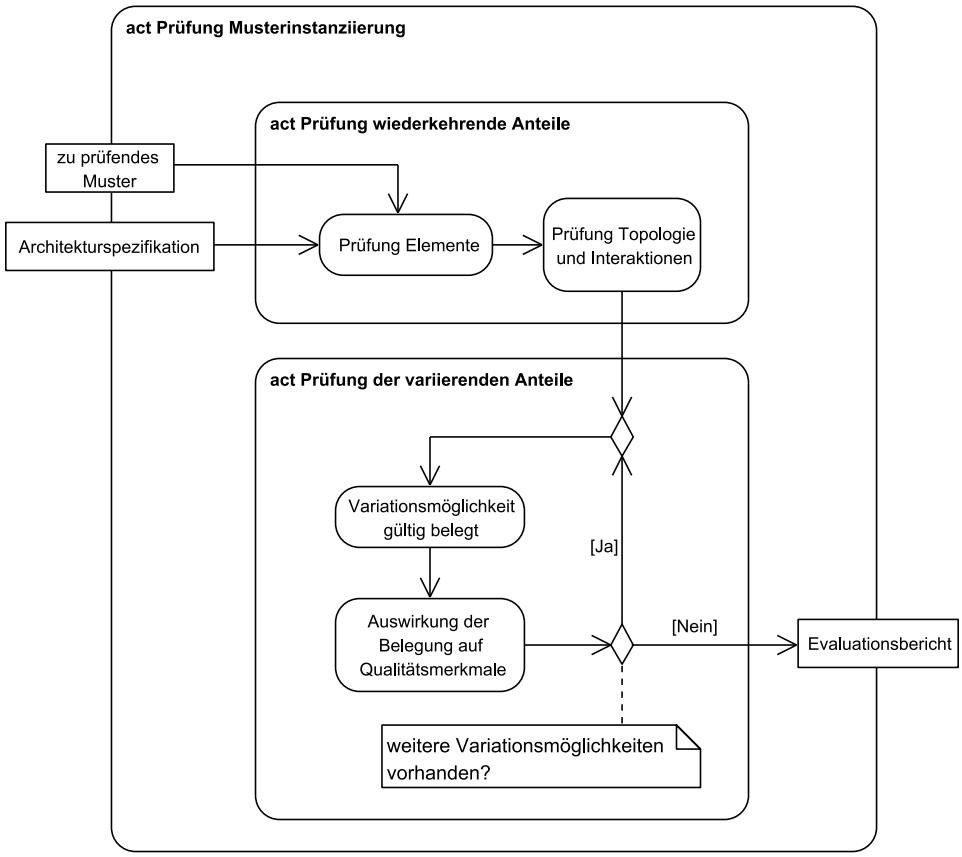

Abbildung 4.12.: Vorgehen zur Prüfung der korrekten Instanzierung eines Musters

rung des Musters. Die fehlerhafte Instanzierung des Musters wird im Evaluationsbericht beschrieben. Über das Merkmal „Zugrunde liegende Prinzipien von Qualitätsmerkmalen eines Musters“ wird vom Evaluationsteam erläutert, welche Auswirkungen die fehlerhafte Instanzierung des Musters auf die durch das Muster beeinflussten Qualitätsmerkmale haben. Wirkt sich die fehlerhafte Instanzierung des Musters negativ auf ein Qualitätsmerkmal aus, welches über die Anforderungen gefordert wird, wird die fehlerhafte Instanzierung des Musters im Evaluationsbericht zusätzlich als Risiko aufgenommen.

Nach der Prüfung der wiederkehrenden Anteile eines Musters, werden die variierenden Anteile eines Musters geprüft. Für jede Variationsmöglichkeit, die im Merkmal „Variationsmöglichkeiten eines Musters“ hinterlegt ist, werden die beiden Aktionen „Variationsmöglichkeit gültig belegt“ und „Auswirkungen der Belegung auf Qualitätsmerkmale“ durchlaufen. Bei der Prüfung, ob die betrachtete

Variationsmöglichkeit gültig belegt ist, wird durch das Evaluationsteam geprüft, ob für die Variationsmöglichkeit in der Architekturspezifikation ersichtlich ist, wie die Variationsmöglichkeit belegt wurde (vgl. Definition zur Belegung einer Variationsmöglichkeit auf Seite 29) und ob die in der Architekturspezifikation ausgewählte Belegung eine gültige Belegung ist. Ist die Belegung der Variationsmöglichkeit in der Architekturspezifikation nicht ersichtlich, ist die Architekturspezifikation an dieser Stelle unterspezifiziert. Ein entsprechender Eintrag im Evaluationsbericht wird vorgenommen. Liegt eine ungültige Belegung vor, handelt es sich wiederum um eine fehlerhafte Instanzierung des Musters, welche analog zur fehlerhaften Instanzierung der wiederkehrenden Anteile eines Musters behandelt wird, außer dass das Evaluationsteam in diesem Fall das Merkmal „Zugrunde liegende Prinzipien von Sensitivity oder Tradeoff Points“ verwendet, um zu erläutern, welche Auswirkungen sich auf die durch die Variationsmöglichkeit beeinflussten Qualitätsmerkmale durch die fehlerhafte Instanzierung ergeben.

Nach den Prüfungen der Belegung der betrachteten Variationsmöglichkeit werden die Auswirkungen der Belegung auf Qualitätsmerkmale untersucht. Über das Merkmal „Qualitätseinfluss von Variationsmöglichkeiten“ stellt das Evaluationsteam fest, welche Qualitätsmerkmale auf welche Weise durch die Variationsmöglichkeit beeinflusst werden. Jedes beeinflusste Qualitätsmerkmal wird den Anforderungen bezüglich dieses Qualitätsmerkmal gegenüber gestellt. Steht die Beeinflussung zu einem geforderten Qualitätsmerkmal in Konflikt, handelt es sich um ein Risiko, welches im Evaluationsbericht als solches festgehalten wird. Erfüllt oder verstärkt die Beeinflussung ein gefordertes Qualitätsmerkmal, wird die Beeinflussung als Sensitivity oder Tradeoff Point im Evaluationsbericht festgehalten. Bestehen bezüglich der Beeinflussung keine spezifizierten Anforderungen, wird kein Eintrag im Evaluationsbericht vorgenommen.

Prüfung der Beziehungen zu weiteren Mustern. Mit Abschluss der Prüfungen zur korrekten Instanzierung eines Musters werden die möglichen Zusammenhänge zu anderen Mustern geprüft. Dazu wird das Merkmal „Beziehungen zu anderen Mustern“ verwendet. Aus den im Merkmal „Beziehungen zu anderen Mustern“ hinterlegten Beziehungen werden in dieser Aktion nur Beziehungen der Typen „besteht aus“ und „verwendet“ genutzt. Sind Beziehungen zu anderen Mustern des Typs „besteht aus“ hinterlegt, so müssen die unter dieser Beziehung hinterlegten Muster (oder Verfeinerungen dieser Muster) in der Architekturspezifikation identifiziert werden können. Ist dies nicht der Fall, liegt eine fehlerhafte Instanzierung des Musters vor, die im Evaluationsbericht entsprechend dokumentiert wird. Können die Muster dieser Beziehung in der Architekturspezifikation identifiziert werden, müssen sie wiederum jeweils den Prüfungsprozess für Muster durchlaufen (vgl. Abbildung 4.11 auf Seite 105).

Ähnlich wird auch bei Beziehungen des Typs „verwendet“ verfahren. Der Unterschied liegt darin, dass Muster, die unter den Beziehungen des Typs „verwendet“ geführt werden, nicht zwingend verwendet werden müssen (vgl. Abschnitt 4.2.1 auf Seite 76). Aus diesem Grund ist für Muster, die unter dem Beziehungstyp „verwendet“ aufgeführt werden, durch das Evaluationsteam zunächst zu prüfen, ob die Rahmenbedingungen und der Bedarf für die Verwendung des Musters vorliegen. Die Informationen zur Prüfung der Rahmenbedingungen und des Bedarfs sind nicht im POSAAM-Wissensmodell hinterlegt. Diese Informationen müssen aus den Musterbeschreibungen gewonnen werden, auf die aus dem POSAAM-Wissensmodell mittels des Merkmals „Referenz zu Quelle des Musters“ verwiesen wird. Sind die Rahmenbedingungen und der Bedarf für die Verwendung des Musters gegeben, prüft das Evaluationsteam, ob das Muster in der Architekturspezifikation identifiziert werden kann. Ist dies der Fall, wird für das Muster die Aktivität zur Prüfung des Musters durchlaufen. Kann das Muster nicht identifiziert werden, wird mit der Aktion „Identifikation alternativer Architekturmechanismen“ weiter verfahren. Die Rahmenbedingungen und der Bedarf an das Muster, die aus dem eingesetzten Muster resultieren, werden für die Identifikation alternativer Architekturmechanismen mitgeführt.

4.4.3. Identifikation alternativer Architekturmechanismen zur Beeinflussung von Qualitätsmerkmalen

In den vorangegangenen Abschnitten wurde erläutert, wie in POSAAM eine Prüfung anhand der Identifikation und Analyse von Mustern in der zu prüfenden Architekturspezifikation vorgenommen wird. Eine Prüfung nur mit der Hilfe von in Mustern gekapseltem Expertenwissen durchzuführen ist für den Einsatz in der industriellen Praxis nicht geeignet, da in der industriellen Praxis auch Probleme zu lösen sind, für die keine Muster bekannt sind oder für die andere Lösungen geeigneter sind. Außerdem kann es vorkommen, dass Muster für ein gegebenes Problem nicht verwendet wurden, aber trotzdem eine dem Problem angemessene Lösung auf andere Weise realisiert wurde. Aus diesen Gründen basiert POSAAM nicht alleine auf der Identifikation von Mustern in einer gegebenen Architekturspezifikation. In POSAAM wird auch beschrieben, wie die Prüfung vorzunehmen ist, wenn keine Muster in der zu prüfenden Architekturspezifikation identifiziert werden können. Die Beschreibung dieses Verfahrens erfolgt in diesem Abschnitt.

Durch die Identifikation von Mustern wird in POSAAM die Abhängigkeit von Experten reduziert und die Systematik und damit auch die Nachvollziehbarkeit der Methode erhöht[3]. Um diese Eigenschaften auch ohne die Nutzung von

[3] Im Vergleich zu anderen Architekturbewertungsmethdoen wie z.B. ATAM

Mustern erhalten zu können, wird in POSAAM auf die Identifikation von Architekturmechanismen, die Prinzipien erfüllen, zurückgegriffen. Um die alternativen Architekturmechanismen in der zu prüfenden Architekturspezifikation identifizieren zu können, gibt es zwei Verfahren. Diese hängen davon ab, ob die Suche nach alternativen Architekturmechanismen geschieht, weil ein Muster im Verlauf der Evaluation zur Lösung eines Problems als geeignet betrachtet wurde und in der Architekturspezifikation nicht identifiziert werden konnte oder ob für eine Anforderung kein geeignetes Muster zur Lösung der Anforderung gefunden wurde. Abhängig davon, welcher Fall vorliegt, wird auf andere Aktionen verzweigt, um festzustellen, welche Prinzipien zur Lösung eines Problems bzw. zur Erfüllung einer Anforderungen geeignet sind (vgl. Abbildung 4.13).

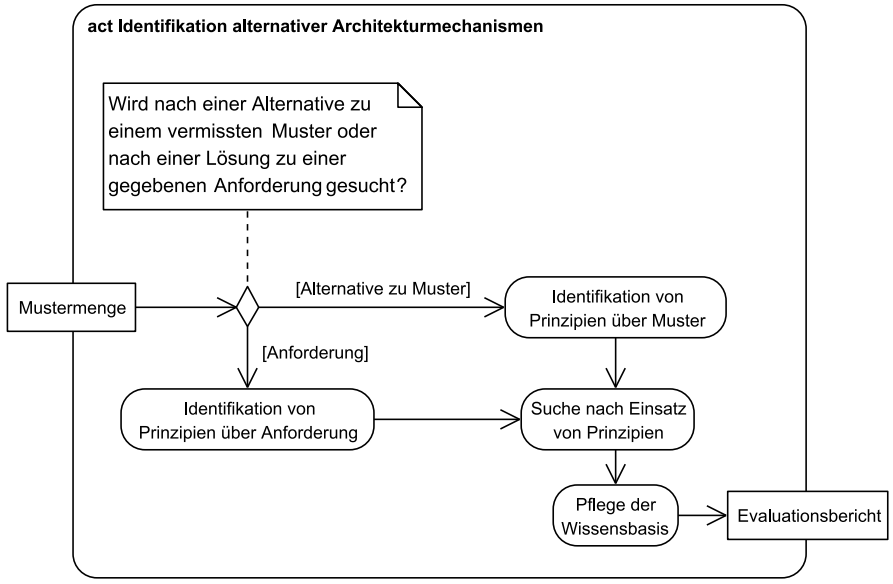

Abbildung 4.13.: Vorgehen zur Identifikation alternativer Architekturmechanismen zur Beeinflussung von Qualitätsmerkmalen

Identifikation von Prinzipien über Anforderung

Konnte in der Wissensbasis von POSAAM für eine Anforderung kein Muster gefunden werden, welches sich für die Erfüllung der Anforderung eignen würde, geschieht die Identifikation des Prinzips über die Anforderung. Da die Anforderung einem Qualitätsmerkmal zugeordnet werden kann (siehe Abschnitt 4.3.1) und auch Prinzipien über das Merkmal „Qualitätseinfluss von Prinzip“ des POSAAM Wissensmodells Qualitätsmerkmalen zugeordnet werden können, ist es möglich, einen Zusammenhang zwischen der Anforderung und dazugehörigen

Prinzipien herzustellen. Die Ausgabe dieser Aktion besteht aus einer Menge von Prinzipien, die genutzt werden können, um das durch die betrachtete Anforderung geforderte Qualitätsmerkmal positiv zu beeinflussen.

Identifikation von Prinzipien über Muster

Erfolgt die Suche nach alternativen Architekturmechanismen zur Beeinflussung von Qualitätsmerkmalen, weil keines der erwarteten Muster in der Architekturspezifikation identifiziert werden konnte (siehe Abschnitt 4.4.1), wird die Menge von Mustern als Eingabe für die Identifikation von geeigneten Prinzipien genutzt. In diesem Fall können die Prinzipien der in der nachfolgenden Aktion zu suchenden Architekturmechanismen über die vermissten Muster identifiziert werden. Zu allen Mustern sind in der POSAAM Wissensbasis über das Merkmal „Zugrunde liegende Prinzipien von Qualitätsmerkmalen eines Musters“ Prinzipien hinterlegt, über die im Muster die Qualitätsmerkmale beeinflusst werden. Die Ausgabe dieser Aktion besteht aus der Menge von Prinzipien, die durch die Muster, die als Eingabe dieser Aktion übergeben wurden, genutzt werden, um Qualitätsmerkmale zu beeinflussen.

Suche nach Einsatz von Prinzipien

Wenn bekannt ist, welche Prinzipien verwendet werden können, um die Qualitätsmerkmale zu beeinflussen, die durch vermisste Muster oder durch spezifizierte Anforderungen gefordert werden, sucht das Evaluationsteam in der Architekturspezifikation nach Architekturmechanismen, die die gesuchten Prinzipien realisieren. Zu diesem Zweck nutzt das Evaluationsteam Informationen über die Eigenschaften, die durch Prinzipien geregelt werden. Diese Informationen sind im POSAAM Wissensmodell unter dem Merkmal „Eigenschaften, die durch Prinzipien geregelt werden“ hinterlegt. Diese Eigenschaften sind in der industriellen Praxis nicht in allen Architekturspezifikationen hinterlegt. Wenn sie aber hinterlegt sind, können sie verwendet werden, um zu prüfen, ob die Eigenschaften wie gewünscht beeinflusst werden. Sind die Eigenschaften der Prinzipien in der vorliegenden Architekturspezifikation nicht hinterlegt, versucht das Evaluationsteam anhand der Regeln, die die Prinzipien zur Strukturierung der Architektur vorgeben, die Anteile der Architektur zu identifizieren, die das Prinzip realisieren. Zusätzlich werden die Eigenschaften vom Evaluationsteam selbst geschätzt und im Evaluationsbericht hinterlegt. Kann der Einsatz eines Prinzips durch das Evaluationsteam nicht nachvollzogen werden, wird ein Risiko im Evaluationsbericht eingetragen. Das Risiko wird mit einer Begründung hinterlegt, in welcher entweder die Erfüllung einer Anforderung bemängelt oder der Einsatz eines Musters vermisst wird.

Wurden durch das Evaluationsteam Architekturmechanismen identifiziert, die die gesuchten Prinzipien realisieren, ist für das weitere Verfahren eine Fallunterscheidung zu treffen. Handelt es sich um Architekturmechanismen, die dazu dienen, die Qualitätsanforderungen einer Anforderung zu erfüllen, für die in der POSAAM Wissensbasis keine Muster identifiziert werden konnten, ist ggf. die Wissensbasis zu erweitern. Wie dabei zu verfahren ist, wird in der folgenden Aktion „Pflege der Wissensbasis“ erläutert.

Handelt es sich um Architekturmechanismen, die eine Alternative zu vermissten Mustern darstellen, ist zunächst durch das Evaluationsteam zu prüfen, ob die Alternative tatsächlich angebracht war. Dazu prüft das Evaluationsteam in einem ersten Schritt, ob in der Architekturspezifikation die Architekturentscheidung, das Muster nicht zu verwenden und stattdessen die alternativen Architekturmechanismen zu verwenden dokumentiert ist. Ist dies nicht der Fall, wird im Evaluationsbericht das Fehlen der Architekturentscheidung festgehalten. In einem weiteren Schritt werden die Auswirkungen der Prinzipien auf Qualitätsmerkmale, die beim Einsatz des Musters gegriffen hätten, mit den Auswirkungen der Prinzipien, die beim Einsatz der identifizierten alternativen Architekturmechanismen festgestellt werden, verglichen. Ergibt das Resultat des Vergleichs, dass beim Einsatz des Musters eine günstigere Ausprägung eines durch eine Anforderung geforderten Qualitätsmerkmals zu erwarten wäre, wird im Evaluationsbericht zusätzlich zum Fehlen der Dokumentation der Architekturentscheidung notiert, dass ein Risiko vorliegt. Ergibt der Vergleich, dass durch den Einsatz der in der vorliegenden Architekturspezifikation verwendeten Alternative eine für die durch die Anforderungen geforderten Qualitätsmerkmale günstigere Ausprägung zu erwarten ist, kann ggf. eine Erweiterung der Wissensbasis nötigt sein. Wie dabei zu verfahren ist, wird in der folgenden Aktion beschrieben.

Pflege der Wissensbasis

Immer wenn in der Aktion „Suche nach Einsatz von Prinzipien“ eine alternative Form der Beeinflussung eines Qualitätsmerkmals festgestellt wird, ist zu prüfen, ob eine Pflege der Wissensbasis möglich ist. Dazu wird geprüft, ob ein neues Muster in die Wissensbasis aufgenommen werden soll. Dies ist immer dann der Fall, wenn die alternativen Architekturmechanismen, die zur Beeinflussung eines Qualitätsmerkmals verwendet wurden, die Charakteristika eines Musters vorweisen. Das Evaluationsteam muss dementsprechend prüfen, ob

1. sich die Architekturmechanismen auch losgelöst vom konkreten Fall verwenden lassen.
2. die verwendeten Architekturmechanismen, auf andere Fälle angewendet, wiederkehrende architekturelle (Elemente, Topologie, Interaktionsform,

Einschränkungen) Anteile aufweisen.

3. die verwendeten Architekturmechanismen, auf andere Fälle angewendet, variierenden architekturelle Anteile aufweisen.

Sind alle drei Eigenschaften gegeben, handelt es sich bei den alternativen Architekturmechanismen um ein Architekturmuster, welches durch das Evaluationsteam in die Wissensbasis aufgenommen wird. Dabei werden zwei Fälle unterschieden.

1. Es wurden Architekturmechanismen zur Beeinflussung eines Qualitätsmerkmals festgestellt, für welche keine Form der Beeinflussung durch Muster in der Wissensbasis hinterlegt waren.
2. Es wurden alternative Architekturmechanismen zu einem in der Wissensbasis vorhandenem Muster festgestellt, für die eine günstigere Ausprägung von geforderten Qualitätsmerkmalen zu erwarten ist als die Ausprägung, die zu erwarten wäre, wenn das in der Wissensbasis dokumentierte Muster eingesetzt würde.

Die Fälle unterscheiden sich hinsichtlich der Beziehungen, die für die aufzunehmenden Muster in der Wissensbasis hinterlegt werden. Im ersten Fall werden lediglich die Beziehungen zu anderen Mustern aufgenommen, die durch das Evaluationsteam durch Betrachtung des vorliegenden Falls identifiziert werden. Im zweiten Fall ist das neu aufzunehmende Muster eine Alternative zu einem bereits in der Wissensbasis vorhandenem Muster. Da es im POSAAM Wissensmodell keine Beziehung des Typs „Alternative“ gibt, müssen beide Muster eine Beziehung vom Typ „verfeinert“ zu einem weiteren (demselben) Muster haben. Existiert kein solches Muster in der Wissensbasis, muss dieses zusätzlich hinzugefügt werden. Dazu stellt das Evaluationsteam die gemeinsamen Anteile beider Muster fest. Ist das Resultat kein Muster, sind die beiden alternativen Muster keine tatsächlichen Alternativen, da sie nicht im gleichen Kontext verwendet werden können.

4.5. Ergebnisse und weiteres Verfahren

Nach der Erstellung des Evaluationsberichts können die erzeugten Ergebnisse im weiteren Verlauf der Softwareentwicklung verwendet werden. Damit die Ergebnisse des Evaluationsberichts nicht als Dokumente archiviert und im weiteren Entwicklungsprozess ignoriert werden, sind in POSAAM weitere Aktionen zur Verwendung der Evaluationsergebnisse explizit vorgesehen (vgl. Überblick über POSAAM; Abbildung 4.1 auf Seite 70). Wie die Aktivitäten genau durchzuführen sind und welche Änderungen und Ergänzungen dazu notwendig sind,

wird nicht als zentraler Gegenstand der vorliegenden Arbeit angesehen. Daher wird an einigen Stellen auf den Ausblick verwiesen.

In diesem Abschnitt wird die Struktur und der Inhalt der während der Evaluation produzierten Ergebnisse kurz wiedergegeben und erläutert, wie die Ergebnisse in weiteren Aktionen verwendet werden können. Im Abschnitt 4.5.1 wird die Struktur der Ergebnisse zusammengefasst und in Abschnitt 4.5.2 wird auf die weitere Verwendung der Ergebnisse eingegangen.

4.5.1. Ergebnisse

Zentrales Ergebnis von POSAAM ist der Evaluationsbericht. Zwei weitere Ergebnisse sind die Prüfprotokolle der Anforderungs- und Architekturspezifikation.

Prüfprotokoll Anforderungsspezifikation

Die Struktur der Anforderungsspezifikation wird in Tabelle 4.14 dargelegt.

Tabelle 4.14.: Struktur des Prüfprotokolls der Anforderungsspezifikation

Einleitung	
Review der Hauptgeschäftstreiber	
	Allgemeine Fragen und Aussagen
	Fragen und Aussagen zum Ist-Stand
	Fragen und Aussagen zum Soll-Stand
Prüfung der korrekten Darstellung	
	Kategorisierung der Anforderungen
	Mängel bei der Beschreibung der Qualitätsanforderungen
Zuordnung zu Qualitätsmerkmalen	
Möglichkeit der Durchführung einer POSAAM-Evaluation	

Die Struktur spiegelt den Verlauf der Prüfung wider, die in Abschnitt 4.3.1 detailliert erläutert wird.

Prüfprotokoll Architekturspezifikation

Die Struktur des Prüfprotokolls der Architekturspezifikation wird in Tabelle 4.15 dargelegt.

Die Struktur spiegelt den Verlauf der Prüfung wider, die in Abschnitt 4.3.2 detailliert erläutert wird.

Tabelle 4.15.: Struktur des Prüfprotokolls der Architekturspezifikation

Einleitung	
Einhaltung der formalen Anforderungen	
	Darstellung der Hauptgeschäftstreiber
	Logische, statische und dynamische Sicht
	Darstellung der Systemgrenzen
	Konformität zur IEEE 1471
Weitere nützliche Sichten	
Review der inhaltlichen Gestaltung der Sichten	
Verantwortlichkeiten und Interaktionsformen der Bausteine	
Architekturrelevante Entscheidungen	
Möglichkeit der Durchführung einer POSAAM-Evaluation	

Der Evaluationsbericht

Der Evaluationsbericht ist das wichtigste Ergebnis einer POSAAM Evaluation. Die Struktur des Evaluationsberichts wird in Tabelle 4.16 dargelegt.

Der Evaluationsbericht enthält an erster Stelle eine Aufzählung aller in der zu evaluierenden Architekturspezifikation identifizierten Architekturmechanismen. Diese können Muster oder Prinzipien sein. Sowohl bei Mustern als auch bei Prinzipien wird im Evaluationsbericht festgehalten, in welchen Teilen der Architekturspezifikation sie identifiziert wurden. Bei Mustern wird festgehalten, welche Elemente und Verantwortlichkeiten, die durch die wiederkehrenden und variierenden Anteile des Musters spezifiziert werden, durch welche Elemente in der Architekturspezifikation instanziert werden. Bei Prinzipien wird entsprechend festgehalten, durch welche Teile der Architektur die Regeln, die durch ein Prinzip vorgegeben werden, befolgt bzw. umgesetzt werden. Im Abschnitt zur Instanzierung der Muster werden zusätzlich noch die Resultate der Prüfung bezüglich der korrekten Instanzierung des Musters festgehalten. Sowohl für Muster als auch für Prinzipien werden alternative Muster (falls im Rahmen des Evaluationsprozess identifiziert) aufgelistet. Muster, die im Rahmen des Evaluationsprozesses als geeignetere Alternativen zum eingesetzten Architekturmechanismus befunden wurden, werden unter den Risiken aufgelistet. Schließlich werden die im Rahmen der Evaluation identifizierten Sensitivity und Tradeoff Points hinterlegt.

Abschließend existiert im Evaluationsbericht eine Anforderungseinflussmatrix, in welcher alle Anforderungen allen identifizierten Architekturmechanismen gegenüber gestellt werden. Die Einflüsse jedes Architekturmechanismus werden für jede Anforderung notiert. Dies dient zum einen der Kontrolle, dass alle Anfor-

Tabelle 4.16.: Struktur des Prüfprotokolls des Evaluationsberichts

Einleitung
Identifizierte Architekturmechanismen
Identifizierte Musterinstanzen
Musterinstanz 1
Instanzierung des Musters
Potentielle alternative Muster
Musterinstanz 2
Instanzierung des Musters
Potentielle alternative Muster
Musterinstanz ...
Identifizierte Einsätze von Prinzipien
Einsatz Prinzip 1
Nach Prinzip gestaltete Teile der Architektur
Potentielle alternative Muster
Einsatz Prinzip 2
Nach Prinzip gestaltete Teile der Architektur
Potentielle alternative Muster
Einsatz Prinzip ...
Risiken
Sensitivity und Tradeoff Points
Anforderungseinflussmatrix
Schluss

derungen soweit möglich in der spezifizierten Architektur berücksichtigt wurden und dass alle negativen Einflüsse auf Anforderungen als Risiken aufgenommen wurden (vgl. auch Paragraf zur Aktion zur Prüfung der Konsistenz der Ergebnisse in Abschnitt 4.4.1) und zum anderen der Übersicht für den Architekten, damit dieser besser Abschätzen kann welche Risiken Missstände darstellen.

4.5.2. Weiteres Verfahren

Nach Fertigstellung des Evaluationsberichts werden die Ergebnisse der Evaluation im Entwicklungsprozess verwertet. Da bei qualitativen Architekturbewertungsverfahren kein Maß an Qualität errechnet wird, sondern lediglich potentielle Missstände (Risiken) und mögliche Verbesserungen aufgedeckt werden, liegt es am Architekt oder am Team von Architekten, die die Anwendung entworfen haben, abzuschätzen, ob die Missstände zu beheben sind und ob die möglichen Verbesserungen in die Anwendung aufgenommen werden und ggf. wie die Verbesserungen umzusetzen sind. Eine Beteiligung des Evaluationsteams, welches

die POSAAM-Evaluation durchgeführt hat, an den folgenden Aktionen ist nicht mehr notwendig.

Für den Fall, dass die Architekten die im Rahmen der Evaluation identifizierten Risiken als Missstände betrachten, können die Missstände entweder durch eine Veränderung der Architektur oder durch eine Veränderung der Anforderungen behoben werden. Mit den Informationen, die im Rahmen einer POSAAM Evaluation über die in einer Anwendung eingesetzten Muster und Sensitivity und Tradeoff Points gewonnen werden, können qualitative Architekturbewertungsmethoden zielgerichteter durchgeführt werden. Liegen keine Missstände (mehr) vor, muss sicher gestellt werden, dass die durch die Architektur begünstigten Qualitätsmerkmale im Rahmen des weiteren Entwicklungsprozesses erhalten bleiben bzw. erfüllt werden. Im Folgenden wird dargestellt, wie die genannten Aktionen im Einzelnen durchgeführt werden können.

Architektur verändern

Wurden im Rahmen der Evaluation potentielle Mängel festgestellt, ist es naheliegend, diese in der Architektur zu beheben. Da im Rahmen einer POSAAM Evaluation die Architektur eines Systems immer anhand einer auf Dokumenten basierten Architekturspezifikation vorgenommen wird, können die Ergebnisse der Evaluation auf Mängel hindeuten, die auf die dokumentierte Repräsentation der Architektur zurückzuführen sind. In diesem Fall sind die Mängel zunächst in der Architekturdokumentation zu beheben. Als potentielle Mängel sind in POSAAM alle Risiken zu verstehen. Ein Risiko in POSAAM liegt vor, wenn

R1 ein in der Architekturspezifikation identifiziertes Muster einen unerwünschten Einfluss auf ein durch eine Anforderung gefordertes Qualitätsmerkmal hat.

R2 ein in der Architekturspezifikation identifiziertes Muster, welches in seiner üblichen Anwendung ein durch eine Anforderung gefordertes Qualitätsmerkmal in gewünschter Weise beeinflusst, fehlerhaft instanziiert wurde und diese fehlerhafte Instanziierung sich auf die Einflüsse auf das geforderte Qualitätsmerkmal in negativer Weise niederschlägt.

R3 die Anwendung eines Prinzips in der Architekturspezifikation identifiziert wurde, welches einen unerwünschten Einfluss auf ein durch eine Anforderung gefordertes Qualitätsmerkmal hat.

R4 für eine Anforderung keine Entsprechung durch ein Muster oder die Anwendung eines Prinzips in der Architekturbeschreibung identifiziert werden konnte.

R5 für die Anwendung eines Musters der Einsatz eines weiteren Musters vorgesehen ist, welches aber nicht in der Architekturspezifikation identifiziert werden kann und für welches auch keine alternative Gestaltung in der Architekturspezifikation identifiziert werden kann.

Vorausgesetzt, dass eine Veränderung der Architektur aus Sicht der Architekten die angemessene Reaktion auf den im Evaluationsbericht festgehaltenen Risiko darstellt, unterstützen die Informationen über mögliche Alternativen, die ggf. zu einem Risiko festgehalten wurden, den Architekten dabei, den Missstand zu beheben. Eine Veränderung der Architektur ist evtl. nicht die angemessene Reaktion auf ein dokumentiertes Risiko (potentieller Missstand). Es kann sein, dass ein Teil der Architektur, welcher sich negativ auf eine Anforderung auswirkt, anderen Zwecken dient und deshalb aus Sicht der Architekten nicht verändert werden kann oder soll. In diesem Fall ist durch die Architekten zu bewerten, ob tatsächlich ein Missstand vorliegt oder ob kein Missstand vorliegt, da zu erwarten ist, dass das erfasste Risiko die Erfüllung der Anforderung, auf welche sich das Risiko bezieht, nicht verhindert. Ist eine solche Aussage nicht möglich, da die genaue Ausprägung eines Qualitätsmerkmals durch die vorliegende Architektur nicht abzuschätzen ist, kann die Durchführung einer quantitativen Architekturbewertung (auch für Teile der Architektur) angebracht sein. Ist eine Aussage möglich und es liegt aus Sicht des Architekten tatsächlich ein Missstand vor, müssen die Anforderungen entsprechend nachbearbeitet werden.

Anforderungen verändern

Eine Veränderung der Anforderungen ist immer dann angebracht, wenn durch die Evaluation Risiken aufgedeckt wurden, die der Architekt als Missstände bewertet, die aber aus Sicht des Architekten nicht behoben werden können oder sollen, da die Teile der Architektur auf die sich die Missstände beziehen, für die Erfüllung weiterer Anforderungen benötigt werden. Hat der Architekt sich gegen eine Änderung der Architektur bezüglich des identifizierten Missstands entschlossen, ist eine Abschwächung der Anforderung, durch die der Missstand entstanden ist, durchzuführen. Eine Veränderung der Anforderungen muss ggf. mit dem Auftraggeber abgestimmt werden.

Übergang zur quantitativen Architekturevaluation

Die Durchführung von quantitativen Formen der Architekturevaluation ist immer dann sinnvoll, wenn die Aussagen aus qualitativen Formen der Architekturevaluation nicht aussagekräftig genug sind bzw. wenn bezüglich der genauen

Ausprägung eines Qualitätsmerkmals, welches eine wichtige Anforderung darstellt, keine ausreichend genaue Aussage getroffen werden kann.

Über die in der qualitativen Bewertung gewonnenen Kenntnisse über Sensitivity Points können quantitative Analysen gezielt für Teile der Architektur durchgeführt werden. Über die Kenntnisse über Tradeoff Points können nach einer quantitativen Analyse Rückschlüsse auf die weiteren beeinflussten Qualitätsmerkmale gemacht werden.

Voraussetzungen für die Durchführung einer quantitativen Architekturevaluation sind,

- dass es sich bei den Qualitätsmerkmalen, für die eine quantitative Architekturevaluation durchzuführen ist, um solche Qualitätsmerkmale handelt, für die Methoden der quantitativen Architekturevaluation existieren (z.B. Verfügbarkeit oder Performanz).
- dass zu den Qualitätsmerkmalen, zu denen quantitative Berechnungen durchgeführt werden sollen, Erfahrungs- oder Schätzwerte vorliegen, die als Eingabe für die Berechnungen verwendet werden können.
- dass die Architekturspezifikation genügend Informationen beinhaltet, um die quantitativen Methoden anwenden zu können.

In den von *Klein et al.* entwickelten Attribute Based Architectural Styles (ABAS) werden Architekturmuster mit Modellen für die Berechnung der Ausprägungen von Qualitätsmerkmalen gekoppelt [KK99, KKB+99]. Entsprechende Informationen könnten auch im Wissensmodell von POSAAM hinterlegt werden. Im Ausblick werden in Abschnitt 6.2.4 entsprechende Vorschläge gemacht.

Konstruktive Maßnahmen zur Qualitätssicherung einleiten

In [ABC+97, S. 2] schreiben *Abowd et al.* :

> *„[...] it should be noted that an architecture cannot guarantee the functionality or quality required of a system. Poor downstream design, implementation, testing, or management decisions can always undermine an acceptable architectural framework. Decisions at all stages of the life cycle – from high-level design to coding and maintenance – affect quality. A software architecture evaluation assesses the ability of the architecture to support the desired qualities. Refinements of the architecture into implementation that preserve the qualities are necessary for the final product to actually achieve those qualities.“*

Wie *Abowd et al.* richtig darstellen, wird durch die Evaluation der Softwarearchitektur lediglich festgestellt, ob die evaluierte Softwarearchitektur die gewünschten (bzw. geforderten) Qualitätsmerkmale unterstützt (bzw. nicht verhindert). Damit die geforderten Qualitätsmerkmale im weiteren Verlauf des Entwicklungsprozesses berücksichtigt werden, können aufgrund der Ergebnisse von POSAAM Maßnahmen für den Entwurf, die Implementierung und den Test von Teilen des Systems ergriffen werden.

Unter den gängigen Formen für die Beschreibung von Architekturmustern (vgl. [BHS07c]) existieren bereits Formen, in welchen Informationen zum Entwurf und zur Implementierung der Muster enthalten sind. Weitere Informationen für die Durchführung von Tests (insbesondere Tests, bei welchen der Erhalt der Effekte von Mustern auf Qualitätsmerkmale getestet wird) können entsprechend in die Musterbeschreibungen bzw. in das POSAAM Wissensmodell aufgenommen werden. In Abschnitt 6.2.4 des Ausblicks wird auf diese mögliche Erweiterungen des Wissensmodells eingegangen. Für die Integration in den Entwicklungsprozess ist wichtig, dass die Durchführung der entsprechenden Aktivitäten für die identifizierten Muster in die Planungen des weiteren Verlaufs aufgenommen werden.

4.6. Einordnung in Klassifikation nach Eicker et al.

In Abschnitt 3.4.1 wurde eine Klassifikation von Architekturbewertungsmethoden nach *Eicker et al.* vorgestellt (siehe Tabelle 3.1 auf Seite 58). In diesem Abschnitt wird POSAAM in diese Klassifikation eingeordnet. Dazu wird POSAAM bezüglich aller in Abschnitt 3.4.1 aufgelisteten Kriterien eingeordnet (Abschnitt 4.6.1). Anschließend werden in Abschnitt 4.6.2 einige neue Kriterien der Klassifikation hinzugezogen. Sowohl ATAM als auch POSAAM werden bezüglich dieser neuen Kriterien positioniert.

4.6.1. Einordnung von POSAAM in vorhandene Kriterien

Die Kriterien der Klassifikation nach *Eicker et al.* werden in Abschnitt 3.4.1 auf den Seiten 56ff. erläutert. Nachfolgend wird POSAAM bezüglich jeder dieser Kriterien positioniert.

Qualitätsmerkmal. Analog zu ATAM ist POSAAM nicht auf ein bestimmtes Qualitätsmerkmal begrenzt. POSAAM ist auf beliebige Qualitätsmerkmale ausgelegt.

Berücksichtigung von Beziehungen. Auch in POSAAM werden die möglichen Wechselwirkungen, die die Erlangung von Qualitätsmerkmalen nach sich ziehen können[4], berücksichtigt.

Voraussetzungen für die Anwendung. Für die Anwendung einer POSAAM Evaluation sind Beschreibungen der Anforderungen sowie der zu evaluierenden Architektur notwendig. Zudem wird eine Wissensbasis mit Informationen zu Architekturmustern, Prinzipien, Qualitätsmerkmalen und deren Beziehungen untereinander benötigt.

Reifegrad / Validierung. Da POSAAM bisher lediglich exemplarisch anhand einer Fallstudie evaluiert wurde und darauf ausgelegt ist, durch mehrere Iterationen eine Wissensbasis aufzubauen, ist der Reifegrad von POSAAM niedrig.

Detaillierungsgrad der Prozessbeschreibung. Die Prozessbeschreibung von POSAAM ist sehr ausführlich. Es werden alle Aktivitäten mit dazugehörigen Ein- und Ausgaben beschrieben.

Ziel der Methode. Analog zu ATAM ist Ziel der Methode POSAAM die frühzeitige Erkennung von Fehlentwicklungen bzw. Risiken. Auch werden Sensitivity und Tradeoff Points in POSAAM festgehalten.

Projektphase. *Eicker et al.* unterscheiden in diesem Kriterium lediglich zwischen einer Evaluation zu einer frühen oder einer späten Projektphase (siehe [EHM07, S.6]). Zwar ist POSAAM danach auch den Methoden zuzuordnen, die eine frühe Evaluation ermöglichen (vor der Implementierung), aber bei POSAAM müssen bereits eine fertige Beschreibung der Architektur des Systems sowie der Anforderungen vorhanden sein. Da diese Information bereits im Kriterium der Voraussetzungen für eine Evaluation hinterlegt ist, wird in der tabellarischen Zusammenfassung (Tabelle 4.17) lediglich die Information „früh“ ausgegeben.

Bewertungsansatz. Der Bewertungsansatz von POSAAM basiert auf der Zuordnung von Anforderungen zu Mustern bzw. Prinzipien und der Identifikation

[4] Anmerkung: Nicht die Qualitätsmerkmale selbst stehen zueinander in einer Wechselwirkenden Beziehung, sondern die Mittel, durch die die Erlangung einer Ausprägung eines Qualitätsmerkmal erreicht werden.

dieser in Architekturbeschreibungen. Mit Hilfe von Informationen über die Zusammenhänge zwischen Mustern, Prinzipien und Qualitätsmerkmalen, die in einer Wissensbasis hinterlegt sind, findet die Evaluation statt.

Beteiligung Stakeholder. An einer Evaluation nach POSAAM sind keine Stakeholder beteiligt.

Erfahrung und Kenntnisse des Evaluationsteams. Für die Durchführung einer POSAAM Evaluation sind Kenntnisse im Umgang mit Mustern im Allgemeinen und mit Notationen und Beschreibungen von Softwarearchitekturen notwendig. POSAAM beruht darauf, dass Mitglieder des Evaluationsteams Architekturmuster in Architekturbeschreibungen identifizieren können. Erst wenn ein Muster in der Beschreibung identifiziert wurde, kann eine Unterstützung des Teams durch eine aufgebaute Wissensbasis erfolgen. Weitere Kenntnisse des Evaluationsteams sind nötig, wenn Muster nicht identifiziert werden können, da in diesem Fall das Evaluationsteam eine Zuordnung der verwendeten Mechanismen zu Prinzipien vornehmen muss. Auch für die kontinuierliche Verbesserung der POSAAM Wissensbasis sind Kenntnisse über Muster, Prinzipien und deren Zusammenhänge zu Qualitätsmerkmalen von notwendig. Das vorliegende Kriterium wird aus diesem Grund für POSAAM als „Mittel“ eingestuft, da bei einer POSAAM Evaluation weniger Erfahrungen und Kenntnisse durch das Evaluationsteam benötigt werden, als bei einer ATAM Evaluation, bei welcher dieses Kriterium mit „Hoch“ eingestuft wurde.

In Tabelle 4.17 werden die obigen Ausführungen in die Klassifikationstabelle von *Eicker et al.* (siehe dazu Ausführungen in Abschnitt 3.4.1) eingebaut. Für den Vergleich wird an dieser Stelle nur die Klassifikation von ATAM wieder gegeben.

4.6.2. Ergänzung von Kriterien

Einige der Alleinstellungsmerkmale von POSAAM treten in der Klassifikation nach *Eicker et al.* nicht auf. Deshalb wird die Klassifikation nach *Eicker et al.* um die folgenden Kriterien erweitert:

Systematik und Vorhersehbarkeit bzw. Wiederholbarkeit. Ein positiver Effekt von analytischen Qualitätssicherungsmethoden ist, dass Produktverantwortliche versuchen, die Qualität des Produkts bereits vor der analytischen Qualitätssicherung möglichst hoch zu halten. Dies ist nur möglich, wenn der Produktverantwortliche die Qualität des Produkts selbst abschätzen kann. Dazu muss der Produktverantwortliche die Möglichkeit haben, die Ergebnisse der

Tabelle 4.17.: Einordnung von POSAAM in die Klassifikation von Methoden zur Architekturbewertung nach *Eicker et al.*

	POSAAM	**ATAM**
Qualitätsmerkmale	Verschiedene	Verschiedene
Berücksichtigung von Beziehungen	Ja	Ja
Voraussetzungen für die Anwendung	Beschreibungen der Architektur und Anforderungen + Wissensbasis zu Architekturmuster	Durch die Analyse- und Befragungsphase hoher Ressourcenbedarf
Reifegrad / Validierung	Eine Fallstudie zur Validierung; dennoch unausgereift	Sehr ausgereift, mittlerweile in 2. Version, Fallstudien und Trainings, validiert in vielen Projekten
Detaillierungsgrad der Prozessbeschreibung	Ausführlich, inklusive Fallstudie	Ausführlich, inklusive Fallstudie
Ziele der Methode	Risiko, Sensitivity, Tradeoff	Sensitivity, Tradeoff, Risiko
Projektphase	Früh	Früh
Bewertungsansatz	Szenarios, Muster, Prinzipien	Utility-Baum, Szenarios
Beteiligung Stakeholder	Nein	Ja
Erfahrung und Kenntnisse des Evaluationsteams	Mittlere Anforderung an das Evaluationsteam wegen der Identifikation von Mustern und Prinzipien in Beschreibungen und ggf. der Pflege der Wissensbasis	Hohe Anforderung an das Evaluationsteam wegen Aufstellen des Utility-Baums und der Identifizierung der Architekturansätze

analytischen Qualitätssicherung vorherzusehen. Bei diesem Kriterium werden Methoden danach unterschieden, wie sehr sich Ergebnisse von Evaluationen vorhersehen lassen bzw. inwieweit bei zwei unabhängig durchgeführten Evaluationen gleiche Ergebnisse auftreten.

Durch die vorgegebene Systematik in POSAAM und die Nutzung einer Wissensbasis können Architekten zu beanstandende Ergebnisse u.U. vorhersehen und ihre Entscheidungen im Voraus begründen oder entsprechend anpassen. Eine absolute Vorhersehbarkeit der Ergebnisse ist auch in POSAAM nicht gegeben, da die Methode auf die Identifikation von Mustern durch Menschen basiert. Bei ATAM können die Ergebnisse einer Evaluation lediglich durch einen Experten abgeschätzt werden, der die gleichen Kenntnisse zu Architekturen und die gleichen subjektiven Meinungen über geeignete Lösungen im Rahmen der Gestaltung von Architekturen hat.

Nachvollziehbarkeit. In [TKL08, S. 309] schreiben *Tang et al.*, dass zwischen Architekten, die Architekturen entwerfen, und Architekten, die Architekturen bewerten, Skepsis bezüglich der Expertise der jeweils anderen Architekten existieren und dass die subjektiven Erfahrungen der Architekten ihre Ansichten beeinflussen. Auch berichten *Tang et al.* darüber, dass erfahrenere Architekten bei Diskussionen eine dominantere Rolle einnehmen können und somit unerfahrenere Architekten ihre Argumentation nicht erfolgreich durchsetzen können. Mit diesem Kriterium werden Methoden danach unterschieden, wie objektivierbar und somit nachvollziehbar die Ergebnisse der jeweiligen Methoden sind.

In POSAAM werden alle Ergebnisse auf Muster oder Prinzipien zurückgeführt. Eine vollständige Objektivierung ist dadurch nicht gegeben, aber zumindest werden die produzierten Ergebnisse mit Expertenwissen (in Form von Architekturmustern) und anerkannte, u.U. wissenschaftlich fundierte Praktiken (Prinzipien) untermauert. Bei ATAM ist die Nachvollziehbarkeit abhängig von der Argumentation der Evaluatoren.

Weiterentwicklung und Pflege. Qualitative Architekturbewertungsmethoden beruhen auf Expertenwissen. Bei der Durchführung einer Bewertung kann neues Expertenwissen zutage kommen. In diesem Kriterium werden die Methoden danach verglichen, wie das im Laufe einer Bewertung neu gewonnene Expertenwissen festgehalten und für spätere Bewertungen nutzbar gemacht wird.

In ATAM wird für die dritte Phase (nach der Durchführung der eigentlichen Evaluation) vorgegeben, dass die erhobenen Szenarien sowie verwendete Analysis Questions für weitere Evaluationen dokumentiert werden (vgl. [CKK02, S. 78ff.]). Die Dokumentation weiterer Erkenntnisse wird nicht explizit vorgegeben. In POSAAM wird eine Wissensbasis während der Durchführung der Evaluation mit neuen Erkenntnissen (Mustern) gefüllt.

Einbettung in den Softwareentwicklungsprozess Die Durchführung analytischer Qualitätssicherungsmaßnahmen ist nur dann sinnvoll, wenn die Resultate der Maßnahmen dazu führen, dass die Produkte, die analysiert wurden, bezüglich der Ergebnisse der Analyse verbessert werden. Insbesondere bei Resultaten, die in Form von Dokumenten vorliegen, kann es geschehen, dass die Resultate nicht in die Produkte einfließen. Bei diesem Kriterium werden die Methoden danach verglichen, inwieweit sie die Berücksichtigung ihrer Resultate im weiteren Entwicklungsprozess forcieren.

Mit Fertigstellung des Evaluationsberichts enden die Betrachtungen in ATAM. Die weitere Verwendung der Ergebnisse des Evaluationsberichts ist in POSAAM explizit vorgesehen. Damit ist die Evaluation nicht abgeschlossen, solange nicht

Maßnahmen zur Bearbeitung der Evaluationsergebnisse in die Planung des Entwicklungsprozesses aufgenommen oder durchgeführt wurden.

In Tabelle 4.18 werden nachfolgend die soeben dargelegten Gegenüberstellungen der Methoden POSAAM und ATAM bezüglich der neu aufgestellten Kriterien zusammen gefasst.

Tabelle 4.18.: Neue Klassifikationskriterien für Methoden zur Architekturbewertung

	POSAAM	**ATAM**
Vorhersehbarkeit	Durch Systematik und Wissensbasis ist eine Vorhersehbarkeit in Teilen gegeben.	Vorhersehbarkeit nur bei Experten mit gleichem Kenntnisstand und gleicher Meinung.
Nachvollziehbarkeit	Zurückführung auf Muster und Prinzipien.	Argumentation des Evaluationsteams.
Pflege	Pflege der Wissensbasis mit Mustern während der Evaluation.	Dokumentation von Szenarien und Analysis Questions nach der Evaluation.
Integration in Entwicklungsprozess	Bearbeitung der Ergebnisse in Methode explizit vorgesehen.	Bearbeitung der Ergebnisse offen gelassen.

KAPITEL 5

Fallstudie

In diesem Kapitel wird die erarbeitete Methode validiert. Dazu wird gezeigt, wie die in vorherigen Kapiteln dargelegten Maßnahmen und Techniken auf eine echte Fallstudie angewendet wurden.

Inhalt

5.1. Erkenntnisse bei der Durchführung von Fallstudien

Für die Erprobung der in dieser Arbeit entwickelten Methode POSAAM wurde nach geeigneten Softwareentwicklungsprojekten gesucht. Es wurden Unternehmen bezüglich der Teilnahme an ATAM-Evaluierungen angesprochen. Weitere Unternehmen wurden bezüglich einer Einsicht in Projektunterlagen hinsichtlich Architektur sowie des Austauschs mit Architekten zum Zwecke der Durchführung einer Architekturevaluation angesprochen. Bei der Suche nach geeigneten industriellen Softwareentwicklungsprojekten, die sich für die Durchführung einer POSAAM Evaluation eignen würden sowie bei der Durchführung der Fallstudie „Batch-Frame“, die in Abschnitt 5.3 beschrieben wird, sind allgemeine Erkenntnisse gewonnen worden.

Die gewonnenen Erkenntnisse werden im folgenden dargelegt. Das die Softwarearchitektur eines Systems in der industriellen Softwareentwicklung als wichtiges Gut erkannt wurde, wird in Abschnitt 5.1.1 begründet. Dass die Praxis des Umgangs mit Softwarearchitektur dieser Erkenntnis nicht gerecht wird, wird durch die Erfahrungen, die in Abschnitt 5.1.2 dargelegt werden, belegt. In Abschnitt 5.1.3 wird kurz auf den Zusammenhang zwischen diesen Erkenntnissen und der Architekturevaluation eingegangen.

5.1.1. Bedeutung der Architektur in der Industrie

Bereits bei der Suche nach einer geeigneten Fallstudie ist deutlich geworden, dass Softwarearchitektur von Unternehmen als geistiges Eigentum mit hohem Wert betrachtet wird. Dies manifestiert sich dadurch, dass jegliches Wissen über existierende Softwarearchitekturen sowie Wissen zum Umgang mit Softwarearchitektur im Allgemeinen bzw. der Evaluation von Softwarearchitektur im Speziellen durch die Unternehmen stark geschützt wurde. Zwar wurde von Ansprechpartnern der Industrie stets betont, dass ein Interesse an Erkenntnissen im Bereich der Softwarearchitektur bestünde, gleichzeitig existierten hinsichtlich der Einbindung von unternehmensfremden Personen große Bedenken.

In drei Fällen wurde eine Zusammenarbeit, bei der eine Einsicht in Methoden zur Softwarearchitektur, eine Teilnahme einer unternehmensfremden Person an einer ATAM-Evaluation oder eine Einsicht in Unterlagen zu Architekturen von Entwicklungsprojekten hätte stattfinden sollen, abgelehnt. Es erfolgte der explizite Hinweis, dass, selbst bei Abschluss einer Vertraulichkeitsvereinbarung, die Einsicht durch oder Teilnahme von unternehmensfremden Personen durch Vorgesetzte nicht genehmigt worden sei. Bei einem Fall erfolgte der Hinweis, dass eine Teilnahme an einer ATAM-Evaluation nur möglich sei, wenn sich ein

Projekt finde, dessen Projektleiter sich dazu bereit erkläre, eine unternehmensfremde Person an der Evaluation teilnehmen zu lassen.

Auch bei Unternehmen, die einer Zusammenarbeit positiv gegenüber stehen, existieren Bedenken bezüglich einer Veröffentlichung der Ergebnisse, bei denen die Softwarearchitektur vollständig offen gelegt wird[1].

Aus diesen Erkenntnissen wird deutlich, dass Softwarearchitektur von den Unternehmen als wichtiger, grundlegender Teil der Problemlösung verstanden wird und somit einen hohen Wert für das Unternehmen darstellt, den es zu schützen gilt.

5.1.2. Praxis des Umgangs mit Architektur

Obwohl der Disziplin der Softwarearchitektur in der Industrie eine hohe Bedeutung beigemessen wird, ist die Praxis des Umgangs mit Softwarearchitektur kaum in den Unternehmen institutionalisiert. Der Umgang mit Themen rund um die Softwarearchitektur geschieht ad hoc.

Diese Entwicklung nimmt ihren Anfang mit der Überführung der Anforderungsspezifikation in die Gestaltung der Architektur. Eine Spezifikation einer Architektur für das zu entwickelnde System vor Beginn der Implementierung oder Modellierung ist unüblich bzw. findet nur in den Köpfen der Entwickler statt. Eine explizite Auseinandersetzung mit der Gestaltung der Architektur ist nicht vorgesehen bzw. institutionalisiert. Unter Umständen findet eine solche Auseinandersetzung in Diskussionen innerhalb von Meetings auf Whiteboards oder mit der Unterstützung von Präsentationen statt. Wie welche Anforderungen durch die Architektur des zu erstellenden Systems in welcher Form adressiert werden, ist im Nachhinein nicht mehr nachvollziehbar.

In der Implementierung ist die Architektur des Systems implizit vorhanden (siehe Definition von Softwarearchitektur in Abschnitt 2.2). Eine explizite Darstellung der Architektur des Systems in Form einer Architekturdokumentation existiert meist nicht. Zwar ist in den Vorgehensmodellen der Unternehmen meist die Erstellung eines Dokuments vorgesehen, in welchem die Architektur des erstellten oder zu erstellenden Systems gefordert wird, jedoch wird in diesem Dokument meist nicht die Architektur, sondern

- die Umgebung,
- ein aus dem Quellcode generiertes Modell der Paket- oder Klassenstruktur oder

[1] Aus diesem Grund ist die vorliegende Fallstudie anonymisiert und nicht in allen Details wiedergegeben.

- die verwendeten Technologien

des Systems beschrieben.

Diese Beschreibungen verdeutlichen die Diskrepanz zwischen dem Verständnis von Softwarearchitektur und dessen Dokumentation, welches in Unternehmen vorherrschend ist, und dem Verständnis von Softwarearchitektur und dessen Dokumentation, wie es in Lehrbüchern (z.B. [BCK03, CBB+02]) oder z.B. dem IEEE Standard 1471 [IEE00] vermittelt wird. Zwar umfasst Architektur alle Strukturen eines Systems (siehe Definition der Softwarearchitektur in Abschnitt 2.2) und somit auch die Paket- oder Klassenstruktur oder die verwendeten Technologien, ausreichend ist eine einzelne Beschreibung jedoch nicht. Zumal die einzelne Beschreibung insbesondere wenn diese aus dem Quellcode generiert wird, entweder zu abstrakt oder zu detailreich ausfällt. Außerdem sind die Beschreibungen für Projektexterne meist nicht mehr lesbar, da sie auf implizitem Wissen basieren.

Ein weiterer Aspekt, in welchem sich das Verständnis der Unternehmen vom Verständnis der Lehrbuchmeinung unterscheidet, ist das Verständnis, was ein Muster ist. Das vorherrschende Verständnis eines Musters in der Industrie ist, dass ein Muster eines der Entwurfsmuster des Buchs über Entwurfsmuster von *Gamma et al.* [GHJV95] ist. In keinem der betrachteten Fälle bei der Suche nach einer geeigneten Fallstudie für die Anwendung von POSAAM unterschied sich das Verständnis eines Musters von dem soeben beschriebenen.

5.1.3. Bezug zur Evaluation

Einige der gewonnenen Erkenntnisse haben einen direkten Bezug zur Evaluation von Softwarearchitekturen. Die Form der vorliegenden Beschreibungen zur Softwarearchitektur, die im vorangegangenen Abschnitt beschrieben wurde, lässt nur begrenzt Rückschlüsse auf die zu erwartenden Qualitätseigenschaften eines mit dieser Architektur implementierten Softwaresystems zu. Dies stellt eine Bestätigung des pragmatischen Ansatzes, die Anforderungs- und Architekturspezifikationen vor einer Architekturevaluation zu erheben, wie er in ATAM oder weiteren Methoden vorgesehen ist (siehe Kapitel 3), dar. Dennoch wird in POSAAM weiterhin davon abgesehen, das Erheben der Anforderungen und der Architektur in die Methode zu integrieren. Diese Tätigkeiten werden nicht als zentrale Aufgabe der Architekturbewertung gesehen. Wie die Anforderungs- und Architekturspezifikationen für die Evaluation nach POSAAM erzeugt werden, bleibt für POSAAM nach wie vor offen. Es wird lediglich festgestellt, dass es ein gängiges Problem darstellt, dass diese Spezifikationen in Industrieprojekten u.U. nicht in der benötigten Form vorliegen.

Eine weitere Erkenntnis, die gewonnen werden konnte, ist, dass die Probleme, die beim Entwurf eines Systems gelöst werden müssen, tatsächlich nicht einer gänzlich neuen Natur sind. Die Softwareentwickler der befragten Unternehmen bestätigten, dass in den von ihnen bearbeiteten Projekten Probleme, die insbesondere aus dem Bereich der Qualitätsanforderungen stammen, wiederkehrten und nach den gleichen Schemata gelöst wurden. Diese Erkenntnis stützt den Ansatz, die Nutzung von Mustern und Prinzipien für die Durchführung einer Architekturevaluation zu verwenden. In den Abschnitten zur Beschreibung der Durchführung der POSAAM-Evaluation anhand des Beispiels Batch-Frame und den daraus gewonnenen Erkenntnissen (Abschnitte 5.3, 5.4 und 5.5) wird dies bestätigt.

5.2. Einschränkungen der Fallstudie

Die umfassende empirische Erprobung von Methoden in der Disziplin des Software Engineerings ist i.A. schwierig, da es zum einen nicht möglich ist, kontrollierte Experimente zur Softwareentwicklung durchzuführen und zum anderen eine große Anzahl an Experimenten nur mit hohem Kostenaufwand durchzuführen ist. Kontrollierte Experimente zur Softwareentwicklung durchzuführen, ist nicht möglich, weil die Softwareentwicklung durch die Teilnahme von Menschen inhärent mit komplexen sozialen Strukturen zu tun hat, die zahlreiche Faktoren, die sich auf die Ausführung des Experiments auswirken, einbringen. Eine große Anzahl an Experimenten durchzuführen, ist zwar möglich, aber nicht realistisch, da die Softwareentwicklung von Softwaresystemen kosten- und zeitintensiv ist.

Aus diesem Grund sind in der Disziplin des Software Engineerings zwei andere Verfahren zur empirischen Erprobung von Methoden gängig: Eine Variante ist die Nutzung von sog. „Toy Examples“ (Bezeichnung aus [Sha01] übernommen), bei welchen ein triviales Softwaresystem entwickelt wird, an welchem die Methode erprobt werden kann. Bei der zweiten Variante wird die Methode an einem oder mehreren echten Softwareentwicklungsprojekten erprobt. Bei beiden Varianten existieren Nachteile. Die Aussagekraft eines „Toy Examples“ ist u.U. nicht groß, da bei der Durchführung eines „Toy Examples“ die Probleme, die bei der Entwicklung großer Softwaresysteme auftauchen, üblicherweise nicht gegeben sind. Nichts desto trotz können „Toy Examples“ für manche empirische Erprobungen von großem Wert sein. Bei der Erprobung von Methoden an echten Softwareentwicklungsprojekten ist durch die Vielfalt der Faktoren, die in einem solchen Softwareentwicklungsprojekt auf die Ergebnisse einwirken, u.U. nicht sicher gestellt, welche Einflüsse die zu erprobende Methode auf Ergebnisse des Softwareentwicklungsprojekts hatte. Bei dieser Variante ist zu prüfen, welche

Aussagen bezüglich einer zu erprobenden Methode auf welche Weise getroffen werden können.

Bei der Erprobung der Methode POSAAM existieren weitere Faktoren, die eine Erprobung anhand einer Fallstudie erschweren.

Stand der industriellen Praxis. Wie in den Abschnitten 5.1.2 und 5.1.3 dargestellt, entsprechen die Methoden zur Spezifikation und Dokumentation von Anforderungen und Architektur in der industriellen Praxis nicht einem Standard, der für eine POSAAM-Evaluation ausreichend ist. Ein Teil der Arbeiten zur Durchführung einer Fallstudie zur Erprobung von POSAAM ist deshalb die nachträgliche Umformung der Darstellungen von Anforderungen und Architekturbeschreibungen.

Aufbau einer Wissensbasis. Die Durchführung einer POSAAM-Evaluation basiert auf einer Wissensbasis, die darauf ausgelegt ist, durch die Anwendung in mehreren POSAAM-Evaluationen auf- und ausgebaut zu werden. Bisher existiert noch keine derartige Wissensbasis. Die Wissensbasis, die *Malich* aufgebaut zu haben angibt, ist nicht öffentlich verfügbar. Selbst wenn diese Wissensbasis verfügbar wäre, müssten deren Inhalte erst in die Struktur, die dem POSAAM-Wissensmodell zugrunde liegt, überführt werden (siehe Abschnitte 3.5.1 und 4.2.2). Auch bei der Aufnahme von Mustern aus bereits veröffentlichten Musterbeschreibungen in eine Wissensbasis nach dem POSAAM-Wissensmodell, bleiben in der Wissensbasis Einträge offen, da diese nicht in den Musterbeschreibungen vorhanden sind (siehe Abschnitt 4.2.2).

Fehlende Werkzeugunterstützung. Ein effizienter Umgang mit der Wissensbasis im Rahmen einer Evaluation (z.B. eine schnelle Suche nach einem für eine Anforderung geeignetem Muster oder Prinzip) ist erst durch geeignete Werkzeugunterstützung möglich. Zwar ist eine Wissensbasis in Form eines strukturierten Dokuments (z.B. einer Tabelle), wie sie in [Mal08] von *Malich* vorgeschlagen wird, anfänglich ausreichend, für die langfristige Nutzung im Rahmen von Evaluationen umfassender Softwaresysteme wird der Umgang mit einer solchen Wissensbasis schnell unhandlich.

Aus den soeben genannten Gründen hätte die Durchführung einer vollständigen POSAAM-Evaluation wenig Aussagekraft. Stattdessen wird im Rahmen der Fallstudie gezeigt, dass bei einem echten Softwareentwicklungsprojekt, welches nicht den Einflüssen eines kontrollierten Experiments unterliegt, Problemstellungen auftreten, die durch den Einsatz von Mustern gelöst werden oder gelöst werden können und dass die Orientierung einer Evaluation an der Anwendung

und Konfiguration von Mustern möglich ist. Analog dazu wird gezeigt, dass für Problemstellungen, für die keine Muster identifiziert werden, die Orientierung einer Evaluation an Prinzipien, mit welchen Qualitätseigenschaften von Softwaresystemen beeinflusst werden können, möglich ist.

5.3. Verarbeitung von Massendaten mit Batch-Frame

Beim Projekt „Batch-Frame“ handelt es sich um ein Softwareentwicklungsprojekt zur Neuentwicklung eines betrieblichen Informationssystems. Das Projekt wurde für die vorliegende Veröffentlichung verfremdet. Der Name Batch-Frame ist frei erfunden. Es handelt sich um ein real existierendes Projekt, welches von einem mittelständischen Unternehmen bearbeitet wird. In den folgenden Ausführungen wurden Details des Projekts (beispielsweise Bezeichnungen von Komponenten, Umfeld, Namen aus der realen Welt, usw.) verfremdet, so dass der Zusammenhang zwischen dem Projekt und der vorliegenden Arbeit nicht ohne weiteres hergestellt werden kann. Im weiteren Verlauf der Arbeit werden das Unternehmen, welches Batch-Frame herstellt, als „Hersteller“ und der Kunde, bei dem Batch-Frame zum Einsatz kommen soll, als „Kunde“ bezeichnet.

Das Projekt zur Entwicklung von Batch-Frame hat Anfang 2008 begonnen und ist bis zum Ende 2008 geplant. In diesem Zeitraum sind für Batch-Frame Aufwände von zwischen zehn und 15 Personenmonaten vorgesehen, die von bis zu sechs Personen erbracht werden. Es ist ein Nachfolgeprojekt mit 250 Personentagen Aufwand geplant.

Der Name „Batch-Frame“ ist in Anlehnung an die Hauptfunktionalität, die das zu erstellende Softwaresystem erbringen soll, gewählt worden. Es wird ein Rahmenwerk (Framework) für die Verarbeitung von Massendaten im Stapelverarbeitungsbetrieb (Batch) entwickelt. Eine genauere Beschreibung der Hauptgeschäftsziele von Batch-Frame folgt im Abschnitt zur Anforderungsbeschreibung (5.3.1). Im Anschluss an die Erläuterung der Anforderungen wird in Abschnitt 5.3.2 die Architektur von Batch-Frame vorgestellt[2].

5.3.1. Anforderungsbeschreibung

Beim Betrieb großer betrieblicher Informationssysteme entstehen umfangreiche Datenmengen, die meist in Datenbanken abgelegt werden. Durch den Betrieb

[2]Die Durchführung der zentralen Schritte der POSAAM-Evaluation wird erst im folgenden Abschnitt 5.4 nachvollzogen.

der Informationssysteme über einen langen Zeitraum können in den Daten Redundanzen und Inkonsistenzen entstehen (beispielsweise bei der Fusion der Datenbasen zweier Betriebe, bei der Migration der Datenbasis auf andere Datenbankmanagementsysteme oder bei einer Änderung der Struktur der zu hinterlegenden Daten aber z.B. auch durch schlichtes „altern“ der Daten, durch welches Inkonsistenzen zwischen den Daten und den Bedingungen in der realen Welt entstehen). Die Redundanzen und Inkonsistenzen können den Wert von betrieblichen Vorgängen vermindern (z.B. durch Verfälschungen von Statistiken oder mehrfaches Kontaktieren desselben Kunden). Im weiteren Verlauf der Arbeit wird diesbezüglich über die „Datenqualität“ der Daten in der Datenbasis gesprochen.

Die Bereinigung der Datenbasen „von Hand“ (durch Menschen, die jeden fehlerhaften Datensatz nacheinander einzeln korrigieren) ist durch die Größe der Datenbasen zu teuer. Auch die Automatisierung eines einzelnen Bereinigungsvorgangs durch ein Programm ist letztendlich zu teuer, da durch ein Programm nur eine Form der Bereinigung vorgenommen werden kann. Für unterschiedliche Bereinigungsvorgänge müssten jeweils eigene Programme entwickelt werden.

Das Werkzeug Batch-Frame ist ein Rahmenwerk, welches die Erstellung und Kombination von Vorgängen, durch die im Stapelverarbeitungsbetrieb Änderungen in umfangreichen Datenbasen vorgenommen werden, ermöglicht. Die Erstellung und insbesondere die Kombination von Vorgängen geschieht in einer Form, die auch von Personen durchgeführt werden kann, die nicht über Kenntnisse über die formalen Sprachen, die zur Interaktion mit den Datenbanken verwendet werden, verfügen.

Nachfolgend wird der aktuelle Stand der zukünftigen Systemumgebung beim Kunden kurz dargestellt. Anschließend wird dargestellt, wie das zu entwickelnde System sich in diese Umgebung integrieren bzw. die Umgebung verändern und damit einen neuen Stand (den Soll-Stand) herstellen wird. Auf dem Soll-Stand aufbauend werden anschließend die zentralen funktionalen Anforderungen, Einschränkungen sowie Qualitätsanforderungen dargelegt. Eine umfassende Darstellung aller Anforderungen erfolgt an dieser Stelle aus Gründen der Übersichtlichkeit und Vertraulichkeit nicht. Für die Demonstration der Konzepte von POSAAM ist die umfassende Darstellung aller Anforderungen nicht nötig.

Ist-Stand

Zur Veränderung der Daten der Datenbasis des Betriebs des Kunden stehen den Filialen des Kunden über 800 Webservices zur Verfügung. Die Nutzung dieser Webservices ist für die Angestellten der Filialen nicht möglich, da sie nicht über die notwendigen fachlichen Kenntnisse verfügen. Für die Filialen besteht

lediglich die Möglichkeit, Änderungen der Datenbasis in Auftrag zu geben. Des Weiteren ist über den Inhalt der Datenbasis bekannt, dass Redundanzen und Inkonsistenzen vorliegen. Der Bedarf an automatisierten Verarbeitungen der Daten aus der Datenbasis zur Verbesserung der Datenqualität (z.B. Angleichung des Formats von Telefonnummern aller Einträge und Differenzierung in Mobiltelefon oder Festnetzanschluss) ist absehbar.

Soll-Stand

Den Mitarbeitern des Kunden, die nicht über die technischen Kenntnisse zur Bearbeitung der Daten aus der Datenbasis über die Nutzung von Webservices verfügen, soll ermöglicht werden über das zu entwickelnde System (Batch-Frame) Änderungen an der Datenbasis durchzuführen. Sie sollen dazu keine besondere Interaktionsform erlernen müssen, sondern über ihre fachlichen Kenntnisse sowie über ihre Kenntnisse über die Struktur und die Bedeutung der in der Datenbasis hinterlegten Daten und über die Führung durch die Benutzerschnittstelle von Batch-Frame die Änderungen durchführen können.

Es soll nicht für jede mögliche Form der Änderung an der Datenbasis ein eigenständiges Programm entwickelt werden. Stattdessen sollen die Wiederverwendung, Konfiguration und (Re-)Kombination von grundlegenden Formen der Änderung der Datenbasis sowie die leichte Erweiterung durch zusätzliche, neue grundlegende Formen der Änderung der Datenbasis nutzbar gemacht werden.

Zentrale funktionale Anforderungen

Es folgt eine Aufzählung der funktionalen Anforderungen, durch welche die Kernfunktionalität der Anwendung Batch-Frame abgedeckt wird. Die genannten funktionalen Anforderungen sind auf abstrakte Weise formuliert und bedürfen der weiteren Verfeinerung, um in einer Implementierung umgesetzt werden zu können. Das Dokument mit den vollständigen Anforderungen zu Batch-Frame kann aus Gründen der Vertraulichkeit an dieser Stelle nicht wiedergegeben werden. Für die Zwecke dieser Arbeit ist die Darstellung der Anforderungen in dieser abstrakten Form ausreichend.

F1 Die Anwendung Batch-Frame wird es Benutzern ermöglichen, über eine Benutzerschnittstelle Aufträge zu formulieren, durch welche im Stapelverarbeitungsbetrieb Aktionen auf Einträgen aus der Datenbasis durchgeführt werden können.

F2 Auch das Erstellen und Ausführen von Aufträgen im interaktiven Betrieb wird durch Batch-Frame ermöglicht.

F3 Die Kombination von Aktionen wird über die Benutzerschnittstelle von Batch-Frame ermöglicht.

F4 Bei der Kombination von Aktionen können über die Benutzerschnittstelle von Batch-Frame Prädikate definiert werden, durch die Bestimmt wird, für welche Einträge der Datenbasis die Aktionen ausgeführt werden.

F5 Die Konfiguration von Aktionen wird über die Benutzerschnittstelle von Batch-Frame ermöglicht. Erläuterung: Aktionen können von generischer Natur sein (z.B. Veränderung der Telefonnummer). Über die Benutzerschnittstelle von Batch-Frame wird es möglich sein, die Aktionen für einen konkreten Ablauf spezifisch zu konfigurieren (z.B. alle Telefonnummern werden in ein Format gebracht, bei dem die Ziffern von rechts nach links paarweise durch ein Leerzeichen getrennt werden).

F6 Die Ausführung von Aufträgen wird von Batch-Frame protokolliert.

F7 Die Anwendung Batch-Frame wird Schnittstellen bieten, über welche neue Aktionen zur Verarbeitung von Einträgen aus der Datenbasis in Batch-Frame integriert werden können. Die Integration umfasst

   a) die Konfiguration der neuen Aktion über die Benutzerschnittstelle von Batch-Frame.

   b) die Möglichkeit zur Kombination der neuen Aktion mit anderen, in Batch-Frame integrierten Aktionen über die Benutzerschnittstelle von Batch-Frame.

Die genannten funktionalen Anforderungen bestimmen den Kern der zu entwickelnden Anwendung. Weitere feingranularere funktionale Anforderungen (z.B. dass die Kombination von Aktionen gespeichert und exportiert werden können soll) werden in dieser Arbeit nicht benötigt.

Zentrale Einschränkungen und Qualitätsanforderungen

Zusätzlich zu den Anforderungen, durch die die Funktionalität der Anwendung bestimmt wird, gibt es einige Einschränkungen, wie die Funktionalität zu erbringen ist und einige Qualitätsanforderungen. Im folgenden werden zunächst die zentralen Einschränkungen genannt.

E1 Für die Erstellung der Anwendung Batch-Frame wird die Programmiersprache/Technologie Java verwendet, da die Kenntnisse der Entwickler des Herstellers in dieser Programmiersprache/Technologie gefestigt sind.

E2 Batch-Frame wird als web-basierte verteilte Anwendung entwickelt, um die Ausführung von vielen Arbeitsplätzen mit unterschiedlichen Betriebssystemplattformen aus, ohne die Notwendigkeit einer vorherigen Installation zu ermöglichen.

Nachfolgend werden die Qualitätsanforderungen an die Entwicklung von Batch-Frame genannt.

Q1 In der Anwendung Batch-Frame werden Benutzer identifiziert, um

a) die Ausführung von Aktionen abhängig von Benutzer einschränken zu können. Das heißt, dass es Tupel von Benutzern und Aktionen gibt, für die gilt, dass Batch-Frame die Ausführung der Aktion durch diesen Benutzer nicht zulässt.

b) feststellen zu können, welcher Benutzer einen Auftrag gegeben hat.

c) die Kosten, die durch einen Auftrag entstanden sind, einem Benutzer zuordnen zu können.

Q2 Die Anwendung Batch-Frame wird Mechanismen enthalten, die die Weitergabe der Anwendung von einer Filiale an eine andere Filiale unterbinden.

Q3 Die Anwendung Batch-Frame wird Mechanismen enthalten, die eine Weitergabe von Aktionen von Filiale zu Filiale unterbinden.

Q4 Durch die Form der Lizenzierung der Anwendung Batch-Frame kann die Nutzung der vollen Funktionalität von Batch-Frame wie folgt eingeschränkt werden:

a) Die Anzahl der parallelen Ausführungen von Aktionen kann eingeschränkt werden.

b) Die Möglichkeit zur Kombination von Aktionen kann eingeschränkt werden. Beispielsweise kann durch die Lizenzierung eine Einschränkung vorliegen, durch welche geregelt wird, dass in einem Ablauf höchstens drei Aktionen miteinander kombiniert werden können.

Q5 Bei der Veränderung der Datenbasis durch einen Auftrag darf die gesamte Ausführungszeit des Auftrags die Ausführungszeit der Webservices nicht um mehr als das doppelte überschreiten.

Q6 Wird eine neue Version einer Aktion in Batch-Frame aufgenommen, so werden die Kombinationen von Aktionen, die die erneuerte Aktion in ihrer Kombination enthalten, weiterhin ausführbar sein. Das heißt, an den Kombinationen werden keine Änderungen notwendig sein, damit diese aufgerufen werden können.

Q7 Der Aufbau der Anwendung Batch-Frame wird so gestaltet, dass bei Änderungen der Webservices (z.B. wegen Änderungen der Struktur der dahinter liegenden Datenbank) für eine Anpassung von Aktionen, die von der Änderung betroffen sind, jeweils nicht mehr Aufwand als fünf Personentage zu erwarten ist.

Analog zu den funktionalen Anforderungen gilt, dass die genannten Qualitätsanforderungen für die Zwecke dieser Arbeit genügen.

5.3.2. Zu evaluierende Architektur

In diesem Abschnitt folgt eine Beschreibung der Architektur, die vom Hersteller für die Anwendung Batch-Frame entworfen wurde. Dazu wird die Architektur aus drei Standpunkten beschrieben. Dem logischen Standpunkt, dem technischen Standpunkt und dem Laufzeitstandpunkt. Die Betrachtung aus diesen Standpunkten resultiert in den entsprechenden Sichten, die nachfolgend wiedergegeben werden. Nach der Beschreibung der Sichten der Architektur von Batch-Frame erfolgt eine Beschreibung von Architekturentscheidungen, die beim Entwurf der Architektur von Batch-Frame getroffen wurden.

Logische Sicht der Architektur

In Abbildung 5.1 ist die logische Architektur des Gesamtsystems von Batch-Frame abgebildet. Abbildung 5.1 entspricht keiner festgelegten grafischen Notation. In der Abbildung wird lediglich die Aufteilung der zentralen Funktionen des Systems in zusammengehörige Teile (bzw. Elemente), die von diesen Teilen wahrzunehmenden Aufgaben und Formen der möglichen Interaktionen zwischen diesen Teilen gezeigt. In dieser Sicht existiert keine Aussage über die spätere Realisierung der Funktionen durch Implementierungseinheiten (z.B. Softwaremodule, Klassen, Funktionen).

Die Elemente der in Abbildung 5.1 dargestellten logischen Architektur werden im Folgenden einzeln erläutert. Es wird beschrieben, welche Aufgaben die Elemente

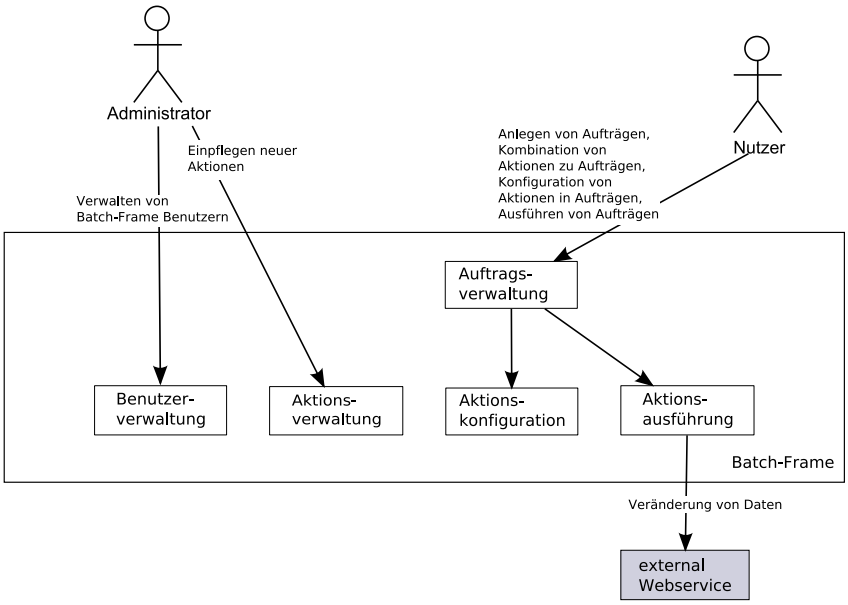

Abbildung 5.1.: Die logische Sicht auf die Gesamtsystemarchitektur von Batch-Frame

innerhalb der Architektur wahrnehmen und mit welchen anderen Elementen diese Elemente zu diesem Zweck auf welche Weise interagieren.

Benutzerverwaltung. Für die Anwendung von Batch-Frame müssen Benutzer verwaltet werden. Es muss möglich sein, Benutzer zum System hinzuzufügen, zu löschen und ihre Eigenschaften zu verändern. Über die logische Komponente Benutzerverwaltung werden diese Funktionen ermöglicht. Informationen über Benutzer werden von anderen logischen Komponenten benötigt. Die nötigen Informationen werden bei Bedarf abgefragt.

Aktionsverwaltung. Aktionen, die Veränderung an der externen Datenbasis über die Webservices vornehmen, sollen zu Batch-Frame hinzugefügt, gelöscht oder verändert werden können. Über die logische Komponente Aktionsverwaltung können neue Aktionen im System registriert und dem System anschließend zur Verfügung gestellt werden.

Auftragsverwaltung. Über die logische Komponente Auftragsverwaltung kann der Benutzer von Batch-Frame Aufträge formulieren und von Batch-Frame ausführen lassen. Zur Formulierung von Aufträgen gehört die Selektion und ggf.

Kombination und Konfiguration von Aktionen, die auf der externen Datenbasis durchgeführt werden sollen.

Aktionskonfiguration. Die Aktionskonfiguration bietet Funktionalität, um Aktionen für die Ausführung zu konfigurieren und vorzubereiten. Die Konfiguration von Aktionen geschieht im Rahmen der Erstellung von Aufträgen und wird deshalb von der logischen Komponente Auftragsverwaltung verwendet.

Aktionsausführung. Die Ausführung der Veränderungen in der externen Datenbasis geschieht losgelöst von der Interaktion mit dem Benutzer. Aus diesem Grund existiert die Komponente Aktionsausführung, die die Ausführung der Aktionen auf der externen Datenbasis autonom durchführt.

Technische Sicht der Architektur

In Abbildung 5.2 ist die technische Architektur des Gesamtsystems von Batch-Frame abgebildet.

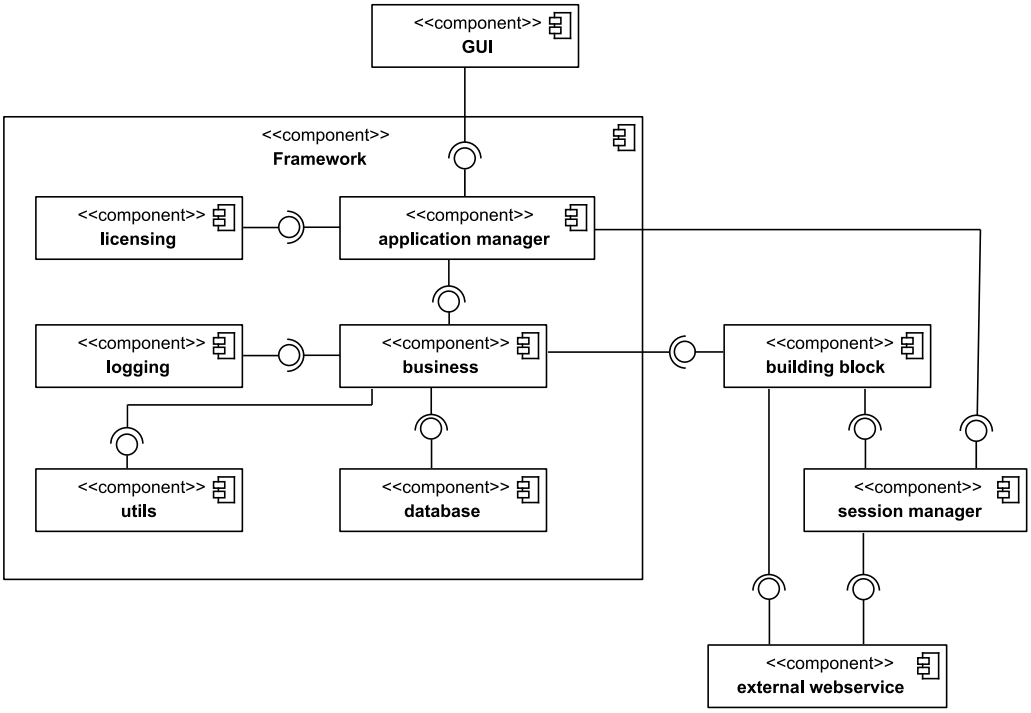

Abbildung 5.2.: Technische Architektur von Batch-Frame

Die Elemente der in Abbildung 5.2 dargestellten technischen Architektur werden im Folgenden einzeln erläutert. Es wird beschrieben, welche Aufgaben die Elemente wahrnehmen und mit welchen anderen Elementen diese Elemente zu diesem Zweck auf welche Weise interagieren.

GUI. Über die Komponente GUI (Graphical User Interface) wird dem Benutzer eine grafische Benutzerschnittstelle zur Interaktion mit Batch-Frame zur Verfügung gestellt. Aufgaben, die von der Komponente GUI wahrgenommen werden, sind:

- Die Verfolgung des Interaktionszustands mit dem Benutzer.
- Die Darstellung von Interaktionsmöglichkeiten mit der Anwendung Batch-Frame. Dazu gehört die Darstellung der für den jeweiligen Zustand der Interaktion des Benutzers mit dem System möglichen Befehle.
- Die Weiterleitung von vom Benutzer abgegebenen Befehlen an die Komponente Framework.
- Die Darstellung von Informationen über den Zustand der Abarbeitung von Befehlen.
- Die Zwischenspeicherung von vom Server übermittelten Informationen.

Um diese Aufgaben zu erfüllen interagiert die Komponente GUI mit dem Benutzer und der Komponente Framework.

Die Darstellung der grafischen Benutzerschnittstelle geschieht in einem Browser. Diese Form der Darstellung wurde gewählt, um die Anwendung ohne vorherige Installation von beliebigen Arbeitsplätzen (des Intranet des Kunden) gewährleisten zu können.

Framework. Die Komponente Framework bildet den Kern der Anwendung Batch-Frame. Sie untergliedert sich (wie in Abbildung 5.2 zu sehen) in die Komponenten Application-Manager, Business, Licensing, Logging, Utils und Database. Aufgaben, die von der Komponente Framework wahrgenommen werden, sind:

- Die Annahme und Verarbeitung von Befehlen, die von der Komponente GUI entgegengenommen werden. Zu den Befehlen zählen insbesondere Aufträge.
- Die persistente Verwaltung von Daten bezüglich der Benutzer der Anwendung Batch-Frame und der Zustände ihrer Aufträge.
- Die Protokollierung der Ausführung von Aufträgen.

- Die Autorisation von Befehlen, so dass diese nur von berechtigten Nutzern ausgeführt werden können.
- Die Autorisation der Nutzung der Anwendung Batch-Frame, so dass diese nur von berechtigten Nutzern genutzt werden kann. Diese Aufgabe bezieht sich auf die Lizenzierung von Batch-Frame.

Die Komponente Framework wird von der Komponente GUI verwendet und verwendet ihrerseits Komponenten des Typs Building-Block bei der Abarbeitung von Aufträgen und die Komponente Session-Manager bei der Autorisation von Befehlen. Es folgt eine Beschreibung der Subkomponenten der Komponente Framework bevor mit der Beschreibung der weiteren Komponenten der Anwendung fortgefahren wird.

Application-Manager Die Komponente Application-Manager nimmt die Befehle der Komponente GUI entgegen und initiiert und regelt deren Bearbeitung. Aufgaben, die von der Komponente Application-Manager wahrgenommen werden, sind:

- Die Annahme, Initiierung und Regelung der Abarbeitung von Befehlen, die von der Komponente GUI entgegengenommen werden. Zu den Befehlen zählen insbesondere Aufträge sowie die Übermittlung von Daten.
- Die Autorisation von Befehlen, so dass diese nur von berechtigten Nutzern ausgeführt werden können.
- Die Autorisation der Nutzung der Anwendung Batch-Frame, so dass diese nur von berechtigten Nutzern genutzt werden kann. Diese Aufgabe bezieht sich auf die Lizenzierung von Batch-Frame.

Die Komponente Application-Manager nutzt bei der Autorisation der Nutzung der Anwendung die Komponente Lizenz, bei der Autorisation von Befehlen die Komponenten Session-Manager und Business und bei der Initiierung und Steuerung der Abarbeitung von Befehlen die Komponente Business. Genutzt wird die Komponente Application-Manger von der Komponente GUI.

Business. Aufgaben, die von der Komponente Business wahrgenommen werden, sind:

- Die persistente Verwaltung von Daten bezüglich der Benutzer der Anwendung Batch-Frame und der Zustände ihrer Aufträge. Zur persistenten Verwaltung von Daten gehören:
  - Das Anlegen, Bearbeiten und Löschen von Nutzern in der Datenbank.

  - Das Anlegen, Bearbeiten und Löschen von Aufträgen in der Datenbank.
  - Das Anlegen, Bearbeiten und Löschen von Kombinationen von Aktionen in der Datenbank.
  - Das Anlegen, Bearbeiten und Löschen von Aktionen (gekapselt in der Komponente Building-Blocks) in der Datenbank.
- Das Entgegennehmen und Bearbeiten von Befehlen von der Komponente Application-Manager. Dazu gehören sowohl die Befehle zur persistenten Verwaltung von Daten als auch Befehle zur Initiierung und Ausführung von Aufträgen.
- Die Transformation von Daten im Format von CSV (Comma Seperated Values) und die Abarbeitung der darin enthaltenen Befehle bzw. Aktionen.
- Die Initiierung der Protokollierung der Ausführung von Aufträgen.
- Die Steuerung der Abarbeitung von Aufträgen.

Für alle Aufgaben der persistenten Speicherung von Daten in der Datenbank nutzt die Komponente Business die Komponente Datenbank. Zum Zwecke der Protokollierung der Ausführung von Aufträgen wird die Komponente Logging verwendet. Zum Zwecke der Transformation und Interpretation von CSV-Tabellen wird die Komponente utils verwendet. Für die Abarbeitung eines Auftrags wird eine Komponente des Typs Building Block verwendet. Welche Komponente des Typs Building Block verwendet wird, hängt von den im jeweiligen Auftrag durchzuführenden Aktionen ab. In den Aktionen ist jeweils hinterlegt, welche Komponente des Typs Building Block für die Abarbeitung zu verwenden ist. Genutzt wird die Komponente Business ausschließlich von der Komponente Application Manager.

Logging. Die Komponente Logging wird von allen Komponenten genutzt, um den Verlauf der Ausführung der Anwendung zum Zwecke der Fehlerauffindung und -korrektur zu protokollieren. Die Komponente Logging muss zu diesem Zweck auf keine weiteren Komponenten zugreifen. In Abbildung 5.2 wird die Interaktionsmöglichkeit aller Komponenten mit der Komponente Logging nicht sichtbar, um die Übersichtlichkeit der Abbildung nicht zu beeinträchtigen.

Utils. Die Komponente Utils wird von allen Komponenten zu allgemeinen Berechnungen und Datentransformationen verwendet. Auch die Interaktionsmöglichkeiten aller Komponenten mit der Komponente Utils tauchen aus Gründen der Übersichtlichkeit nicht in Abbildung 5.2 auf.

Lizenz. Die Komponente Linzenz wird von der Komponente Application-Manager dazu verwendet, um die Autorisation der Nutzung der Anwendung Batch-Frame durchzuführen. Die Komponente Lizenz hat dabei lediglich die Aufgabe zu prüfen, ob Anwendung und die in der Anwendung eingesetzten Building-Blocks korrekt lizenziert wurden.

Datenbank. Für die Datenbank wird ein Fertigprodukt verwendet. In dieser Komponente werden sämtliche Daten bezüglich der Nutzer der Anwendung Batch-Frame sowie die zu Nutzern gehörenden Aufträge und deren Zustände persistent abgelegt. Der Zugriff auf die Datenbank geschieht ausschließlich über die Komponente Business.

Session-Manager. Für den Zugriff auf die Webservices (Komponente External Webservice) ist eine Identifikation und Authentifikation notwendig. Durch die Anwendung Batch-Frame werden an der Datenbasis, die durch die Webservices angesprochen wird, bei jedem Auftrag mehrere Aktionen (oft mehrere tausend Aktionen) durchgeführt. Jede Aktion wird nacheinander, einzeln durch die Nutzung eines Webservice ausgeführt. Da die Identifikation und Authentifikation aus Gründen des Zeitverhaltens der Anwendung nicht für jede Aktion einzeln durchgeführt werden sollen, werden durch die Komponente Session-Manager „Sessions“ verwaltet, mit welchen mehrere Aktionen im Rahmen einer einmaligen Identifikation und Authentifikation ausgeführt werden können. Aufgaben, die von der Komponente Session-Manager wahrgenommen werden, sind:

- Die Annahme von Anfragen von Komponenten des Typs Building-Block und die entsprechende Versorgung von Komponenten dieses Typs mit Verbindungen zu den Webservices.
- Der Aufbau von Verbindungen zu den Webservices. Zum Aufbau gehört auch die Identifikation und Authentifikation eines Benutzers bei der Komponente External Webservice.
- Die Verwaltung von bestehenden Verbindungen zu den Webservices.

Die Komponente Session-Manager nimmt Anfragen der Komponente Application-Manager entgegen und baut Verbindungen zur Komponente External Webservice auf. Komponenten des Typs Building-Block stellen Anfragen an die Komponente Session-Manager, um aktuell bestehende „Sessions“ abzufragen und mit diesen die Verbindung zum External Webservice aufzubauen.

Building-Block. Für den Betrieb von Batch-Frame sind mehrere Komponenten vorgesehen, die zwar für den Betrieb notwendig, jedoch nicht fester Bestandteil

der Anwendung sind. Es handelt sich dabei um die Komponenten, mit welchen über die zur Verfügung gestellten Webservices Veränderungen an der Unternehmensdatenbasis vollzogen werden. In Abbildung 5.2 werden Komponenten dieser Art durch die Komponente Building-Block repräsentiert. Da es mehrere Komponenten gibt, die Aktionen an der Unternehmensdatenbasis durchführen können, handelt es sich im Falle von Building-Block nicht um eine Komponente, sondern um einen Komponententyp. Komponenten des Typs Building-Block werden für die Durchführung von Aktionen auf der Datenbasis verwendet. Alle Komponentnen des Typs Building-Block weisen dieselbe syntaktische Schnittstelle auf (sie implementieren Funktionen mit derselben Signatur) unterscheiden sich jedoch in der Form der Ausführung.

External-Webservice. Die Komponente External-Webservice ist nicht Teil der Anwendung Batch-Frame. Sie wird in der Darstellung der technischen Architektur als Komponente Aufgenommen, ist aber Teil der Systemumgebung von Batch-Frame. Über die von der Komponente External-Webservice zur Verfügung gestellten Schnittstellen können die Änderungen an der Datenbasis des Betriebs des Kunden vorgenommen werden. Diese Schnittstellen werden von den Komponenten Building-Block verwendet.

Mit Ausnahme der Interaktion mit den Komponenten GUI dem externen Webservice geschieht die Interaktion zwischen den Komponenten über Funktions- bzw. Methodenaufrufe. Die Interaktion mit der Komponente GUI geschieht über das HTTP-Protokoll. Die Interaktion mit dem externen Webservice geschieht über das SOAP-Protokoll. Die Schnittstellen der Komponenten (Signaturen der Methoden, abstrakte Superklassen bzw. im Fall von Java meist Interfaces, mögliche HTTP-Aufrufe sowie mögliche Aufrufen von Webservices) sind dokumentiert, werden jedoch in dieser Arbeit aus Gründen der Vertraulichkeit nicht wieder gegeben.

Laufzeitsicht der Architektur

Die Verarbeitung von Aufträgen in Batch-Frame verläuft sequenziell (vgl. Architekturentscheidung AE2 auf Seite 147). Ein Aufruf aus der Benutzerschnittstelle verläuft immer nach dem gleichen Schema. Um zu verdeutlichen, wie der Ablauf vom Aufruf durch einen Nutzer verläuft, wird an dieser Stelle dargelegt, wie ein beispielhafter Aufruf durch einen Nutzer von Batch-Frame verarbeitet wird, welche Leichtgewichtprozesse welche Komponenten aus der technischen Sicht durchlaufen. Als Beispiel dient die Übermittlung eines Stapelverarbeitungsauftrags durch einen Nutzer. Der Stapelverarbeitungsauftrag erfolgt durch die Übermittlung einer CSV-Datei. Der Beispielhafte Aufruf wird als eine Reihe von Schritten

beschrieben. Bei der Beschreibung der Schritte wird auf den Kern des Ablaufs eingegangen. Details werden abstrahiert. Auf eine grafische Darstellung wird an dieser Stelle verzichtet. Es wird auf die Abbildung der technischen Architektur (Abbildung 5.2 auf Seite 140) verwiesen.

1. Der Ablauf beginnt mit der Interaktion durch einen Nutzer mit der graphischen Benutzerschnittstelle (GUI), die in einen Browser eingebettet ist. In der graphischen Benutzerschnittstelle wählt der Nutzer die entsprechende Funktion und kann die CSV-Datei übermitteln.

2. Durch einen HTTP-Aufruf wird in der Komponente Application-Manager innerhalb eines Web-Servers ein neuer Leichtgewichtprozess L erzeugt, welcher die Interaktion mit dem Browser abwickelt. Dazu wird durch den Leichtgewichprozess L zunächst geprüft, ob der Nutzer, der die Aktion angestoßen hat, eine Berechtigung dazu hat. Dies geschieht zum einen durch einen Abgleich mit der Komponente Session-Manager und zum anderen durch einen Abgleich mit den Daten aus der Datenbank, also durch einen Aufruf der Komponente Business. Ist der Nutzer zu der Aktion berechtigt, wird eine Funktion zur Abwicklung eines Auftrags durch Übermittlung einer CSV-Datei in der Komponente Business aufgerufen. Der Funktion wird als Parmeter die CSV-Datei übergeben.

3. Durch den Aufruf der entsprechenden Funktion in der Komponente Business wird der durch die CSV-Datei formulierte Auftrag in Aktionen zerlegt. Dazu werden Funktionen aus der Komponente utils verwendet. Die einzelnen Aktionen werden anschließend in der Datenbank abgelegt (Die Abbildung von Objekten auf Einträge in der relationalen Datenbank erfolgt durch ein Fertigprodukt. Aus Sicht des Programms Batch-Frame erfolgt die Ablage von Daten in der Datenbank lediglich durch die Manipulation von Objekten eines Objektmodells). Nach der Ablage der Aktionen in der Datenbank wird der Leichtgewichtprozess L beendet.

4. In der Komponente Application Manager läuft innerhalb einer Endlosschleife ein Leichtgewichtprozess E, der für alle Benutzer die laufenden Aufträge verwaltet. Der Leichtgewichtprozess E prüft periodisch durch einen Funktionsaufruf in der Komponente Business, ob Aufträge zur Abarbeitung vorliegen. Findet der Prozess E einen auszuführenden Auftrag erfolgt ein Aufruf der Komponente Session-Manager. Sind im Datenmodell der Komponente Session-Manager Informationen über die Identifikation und Authentifikation des aktuellen Benutzers beim Externen Webservice hinterlegt (in Form eines Session Tokens), wird das Session Token an die Komponente Application Manager zurück gegeben. Liegt beim Session Manager kein Session Token vor, werden die Benutzerinformationen des dem

Auftrag zugeordneten Benutzers für die Identifikation und Authentifikation verwendet.

5. Nach Abruf eines gültigen Session Tokens initiiert E die Abarbeitung den nächsten vorliegenden Auftrags. Die Priorisierungsstrategie der Aufträge ist FIFO. Die Abarbeitung des Auftrags geschieht durch den Aufruf einer entsprechenden Funktion in der Komponente Business.

6. Der Leichtgewichtprozess E arbeitet die zu einem Auftrag gehörenden Aktionen ab. Dazu wird durch E für jede Aktion ein Aufruf der Komponente Building-Block vorgenommen. Die Vorgänge, die im Rahmen der Aktion durchzuführen sind, und das Session Token werden als Parameter des Aufrufs mitgeführt.

7. Aus der Komponente Building-Block erfolgt ein Aufruf der Komponente Session-Manager, um zu prüfen, ob das übergebene Session Token noch Gültigkeit besitzt, bevor die Aktion mit einem Aufruf des Externen Webservice durchgeführt werden kann.

8. Mit der Verbindung zum Webservice wird die Aktion durch entsprechende Aufrufe des Webservice von der Komponente Building Block durchgeführt. Nach Abschluss der Aktion geht der Kontrollfluss zurück an die Komponente Business, in welcher der Leichtgewichtprozess E weitere zum aktuellen Auftrag zugehörige Aktionen abarbeitet (Schritte 6 und 7). Wurden alle Aktionen durchgeführt, läuft der Leichtgewichtprozess E zurück in die Komponente Application Manager. Von dort wird durch einen Aufruf der Komponente Session Manager die aktuell gültige Session beim External Webservice beendet. Der Leichtgewichtprozess E läuft anschließend wieder in die Endlosschleife aus Schritt 4.

Architekturentscheidungen

Die beschriebene Architektur von Batch-Frame basiert u.a. auf den nachfolgend beschriebenen Architekturentscheidungen:

AE1 *Komponenten auf Basis von OSGi.* Um die leichte Erweiterbarkeit des Rahmenwerks mit Building-Blocks zu gewährleisten (vgl. Anforderung F7 in Abschnitt 5.3.1 auf Seite 135) basiert Batch-Frame auf dem OSGi Standard. Es wird die Technologie Equinox verwendet.

AE2 *Sequentielle Abarbeitung von Aufträgen.* Um die schnelle Entwicklung einer ersten Fassung von Batch-Frame zu unterstützen

und um unnötige Komplexität bei der Entwicklung zu vermeiden, wird in der ersten Fassung von Batch-Frame auf die parallele Abarbeitung von Aufträgen verzichtet.

AE3 *Zusätzliches Rollen- und Rechtesystem.* Das Rollen- und Rechtesystem, welches über die externen Webservices zur Verfügung gestellt wird, ist für die Nutzung in Batch-Frame nicht ausreichend, da in Batch-Frame beispielsweise definiert werden muss, welcher Nutzer von Batch-Frame Aufträge formulieren, welcher Benutzer Aufträge ausführen und welcher Benutzer das (Batch-Frame) System administrieren kann. Diese Nutzergruppen stimmen nicht mit den durch die externen Webservices zur Verfügung gestellten Nutzergruppen überein. Aus diesem Grund wurde entschieden, in Batch-Frame ein zusätzliches, eigenes Rollen- und Rechtesystem zu verwenden.

AE4 *Client-Server Architektur.* Durch die Nutzung einer Client-Server Architektur, bei welcher der Client in einem Browser eingebettet betrieben wird, muss Batch-Frame lediglich einmalig an zentraler Stelle installiert werden. Die Installation auf Arbeitsplatzrechnern ist nicht nötig. Weitere Vorteile, die sich aus einer Client-Server Architektur ergeben, sind die Möglichkeit, auf der Serverseite ein zentrales Rechtesystem aufzusetzen sowie ein zentrales Accounting und andere Funktionalität umzusetzen, die nur an zentraler Stelle betrieben werden kann (beispielsweise die Drosselung der Abarbeitung von Batch-Aufträgen, um den interaktiven Benutzerbetrieb reibungslos zu gewährleisten).

AE5 *Nutzung einer O-R-Abbildung.* Um die schnelle Entwicklung einer ersten Fassung von Batch-Frame zu unterstützen und um unnötige Komplexität bei der Entwicklung zu vermeiden, wird eine objektrelationale Abbildung (Technologie Hibernate) für den Zugriff auf die von Batch-Frame genutzte Datenbank verwendet.

5.4. Durchführung der Evaluation

Im Folgenden wird demonstriert, wie eine POSAAM-Evaluation anhand der Beschreibungen von Batch-Frame durchgeführt wird. Dabei werden alle grundlegenden Schritte zumindest einmal dargelegt. Einige, für die Zwecke der Demonstration der Evaluation weniger relevanten Schritte, werden nicht dargelegt.

Wenn dies der Fall ist, wird explizit darauf hingewiesen und begründet, weshalb der Schritt in der Demonstration entfallen kann.

Die weitere Struktur dieses Abschnitts orientiert sich am Verfahren zur Evaluation. Daher beginnt der Abschnitt mit der Prüfung der Eingaben (Abschnitt 5.4.1), bei welcher die Anforderungs- und die Architekturspezifikationen, die als Grundlage für die Evaluation dienen, einer Prüfung unterzogen werden. Im Anschluss wird auf den Durchlauf des Hauptzyklus (Abschnitt 5.4.2) eingegangen.

5.4.1. Prüfung der Eingaben

Im vorliegenden Abschnitt wird gezeigt, wie die Prüfung der Eingaben bei der Evaluation der Architektur von Batch-Frame verläuft. Es wird mit der Prüfung der Anforderungsspezifikation begonnen, im Anschluss wird die Prüfung der Architekturspezifikation gezeigt.

Prüfung der Anforderungsspezifikation

Die Prüfung der Anforderungsspezifikation besteht aus den Aktionen „Review der Hauptgeschäftstreiber“, „Anforderungen korrekt beschrieben“ und „Anforderungen in Qualitätsmerkmalbaum einordenbar“. Die Beschreibungen der Inhalte der Aktionen erfolgen in Abschnitt 4.3.1 ab Seite 90.

Review der Hauptgeschäftstreiber. Die Prüfung der Anforderungsspezifikation beginnt mit einem Review der Beschreibungen der Hauptgeschäftstreiber. Aus den vorliegenden Beschreibungen der Fallstudie Batch-Frame sind die primären Zwecke des zu entwickelnden Systems ersichtlich. Sowohl der Ist- als auch der Soll-Stand werden explizit beschrieben.

Anforderungen korrekt beschrieben. Bei der Prüfung, ob die Anforderungen korrekt beschrieben sind, wird jede Anforderung zunächst in architekturrelevant oder nicht architekturrelevant (siehe Definition und Anhaltspunkte in Abschnitt 3.3.5 auf Seite 54) und anschließend in Qualitätsanforderung oder Einschränkung kategorisiert. Das Resultat der Kategorisierung ist in Tabelle 5.1 hinterlegt.

Nach der Kategorisierung der Anforderungen wird für jede Anforderung überprüft, ob diese genügend Informationen enthält, um in Form eines Szenarios nach [BCK03] (siehe auch Erläuterungen zu Szenarien in Abschnitt 3.2 auf Seite 44) formuliert zu werden. Für die Anforderungen von Batch-Frame konnte dies für jede Anforderung bestätigt werden. Zur Demonstration wird an dieser Stelle die Anforderung Q1a in ein Szenario überführt:

Tabelle 5.1.: Klassifikation der Anforderungen an Batch-Frame

Anforderungs-kürzel	**Architektur-relevanz**	**Qualitäts(teil)merkmal**
Q1a	Ja	Informationssicherheit, Integrität; Informationssicherheit, Vertraulichkeit
Q1b	Ja	Informationssicherheit, Verbindlichkeit [Eck01]
Q1c	Ja	Informationssicherheit, Abrechenbarkeit
Q2	Nein	Informationssicherheit, Vertraulichkeit
Q3	Nein	Informationssicherheit, Vertraulichkeit
Q4a	Ja	Informationssicherheit, Vertraulichkeit
Q4b	Ja	Informationssicherheit, Vertraulichkeit
Q5	Ja	Effizienz, Performanz
Q6	Ja	Wartbarkeit
Q7	Ja	Wartbarkeit

Quelle Benutzer des Systems.

Reiz Gibt einen Auftrag an das System. Der Auftrag enthält Aktionen, die Veränderungen an der Datenbasis, die durch die Webservices angesprochen werden, durchführen. Der Benutzer ist zu diesen Veränderungen nicht berechtigt.

Umgebung Das System befindet sich in normalem Betrieb.

Artefakt Batch-Frame.

Antwort Die Aktionen, für die der Benutzer keine Berechtigung hat, werden nicht durchgeführt.

Antwortmetrik Aktion erfolgreich oder Aktion nicht erfolgreich.

Anforderungen in Qualitätsmerkmalbaum einordenbar. Zuletzt wird jede Anforderung zu einem Qualitätsmerkmal oder Qualitätsteilmerkmal in Bezug gebracht. Für die Anforderungen von Batch-Frame ist die Zuordnung in Tabelle 5.1 zusammen mit der Klassifikation in architekturrelevante und nicht architekturrelevante Anforderungen erfolgt.

Prüfung der Architekturspezifikation

Die Prüfung der Architekturspezifikation besteht aus den Aktionen „Formale Prüfungen“, „Weitere nützliche Sichten“, „Review der Sichten“, „Verantwort-

lichkeiten und Interaktionsformen“ und „Entscheidungen“. Die Beschreibungen der Inhalte der Aktionen erfolgen in Abschnitt 4.3.2 ab Seite 94.

Formale Prüfungen. Die Prüfung der Architekturspezifikation beginnt mit einigen formalen Prüfungen. In der Architekturbeschreibung müssen eine logische, eine statische und eine dynamische Sicht auf die Architektur des zu erstellenden Systems vorliegen. Es muss ersichtlich sein, wie durch die Architektur des zu erstellenden Systems die Hauptgeschäftstreiber erreicht werden. Die Systemgrenzen müssen ersichtlich sein und die Architekturbeschreibungen müssen Konform zur IEEE 1471 sein.

Bei der Fallstudie Batch-Frame ist die Erfüllung aller formaler Kriterien bis auf die Konformität zur IEEE 1471 unmittelbar ersichtlich. Auf die Erfüllung der Konformität zur IEEE 1471 wird an dieser Stelle nicht weiter eingegangen, da dies nicht von zentraler Relevanz für die vorliegende Arbeit ist.

Weitere nützliche Sichten. In dieser Aktion werden Sichten identifiziert, die im Rahmen der Evaluation von Nutzen sein könnten. Am Beispiel der Qualitätsanforderung Q5 wird demonstriert, wie über die Nutzung von Prinzipien eine weitere, für die Evaluation nützliche Sicht identifiziert wird. Die Qualitätsanforderung Q5 wurde in Tabelle 5.1 dem Qualitätsmerkmal Effizienz und dem Qualitätsteilmerkmal Performanz zugeordnet. In Anhang B (ab Seite 181) sind Beispieleinträge für die Wissensbasis hinterlegt. In Abschnitt B.2 ist ein Beispieleintrag für das Prinzip „Entlastung einer Ressource“, welches sich positiv auf das Qualitätsmerkmal Effizienz auswirkt. Die Eigenschaften, die durch dieses Prinzip geregelt werden, sind knappe Ressourcen und die Verwendung dieser Ressourcen. Um demnach die Verwendung von Architekturmechanismen zu erkennen, die das Prinzip „Entlastung einer Ressource“ befolgen, müsste eine Sicht vorliegen, in welcher diese Eigenschaften sichtbar werden. In Batch-Frame liegt eine solche Sicht (z.B. eine Verteilungssicht der Architektur) nicht gesondert vor. Aus den in der Architekturspezifikation von Batch-Frame vorliegenden Sichten sind jedoch einige Informationen über knappe Ressourcen und die Verwendung dieser Ressourcen enthalten. Die Verteilung der Komponenten aus der technischen Sicht auf die Architektur ist aus den Beschreibungen ersichtlich, ebenso ist die Art der Verbindung und die Art der Interaktion zwischen den Komponenten aus der technischen Sicht der Architektur ersichtlich. Im Prüfprotokoll für die Prüfung der Architekturspezifikation wird in diesem Fall vermerkt, dass eine gesonderte Sicht mit spezifischen Informationen zu den Ressourcen und dessen Nutzung im Rahmen der Evaluation einen Mehrwert bieten könnte und die hier beschriebene Argumentation über das Prinzip „Entlastung einer Ressource“ zusätzlich hinterlegt.

Review der Sichten. Das Review der Sichten auf die Architektur hinsichtlich konzeptueller Integrität innerhalb und zwischen den Sichten wird an dieser Stelle nicht detailliert betrachtet, da dies sehr umfangreich, aber zugleich nicht zentraler Bestandteil der vorliegenden Arbeit ist. Für die Durchführung einer Evaluation wurde die konzeptuelle Integrität als ausreichend befunden.

Verantwortlichkeiten und Interaktionsformen. Auch auf das Review, in welchem geprüft wird, ob für alle Elemente verständlich ist, welche Aufgaben diese Elemente erfüllen und durch welche Interaktionen mit welchen anderen Elementen diese Aufgaben erfüllt werden, wird im Rahmen dieser Arbeit nicht weiter eingegangen. Für die Beschreibungen, die zur Architektur von Batch-Frame vorliegen, wurde befunden, dass für alle Elemente deutlich wird, welche Aufgaben diese durch welche Interaktionen mit anderen Elementen erfüllen.

Entscheidungen. Zuletzt werden bei der Prüfung der Architekturspezifikation die Architekturentscheidungen geprüft. Dabei wird geprüft, ob die beschriebene Architektur durch Architekturentscheidungen begründet wird, ob zu jeder Architekturentscheidung die möglichen Alternativen ersichtlich sind, ob ersichtlich ist, weshalb eine Alternative gewählt wurde, ob die Begründung, weshalb eine Alternative gewählt wurde, auf die Anforderungen zurückzuführen ist und ob es Anforderungen gibt, die nicht durch Architekturentscheidungen erfasst werden.

Tabelle 5.2 gibt eine kurze Zusammenfassung der Analyse der an dieser Stelle zu prüfenden Fragen. Es fällt auf, dass es Architekturentscheidungen gab (AE2 und AE5), die nicht auf (explizit festgehaltene) Anforderungen zurückzuführen sind. Des Weiteren wird festgehalten, dass für die Anforderungen Q2, Q3, Q4, Q5 und Q6 keine Architekturentscheidungen explizit dokumentiert wurden.

5.4.2. Durchlauf des Hauptzyklus

In diesem Abschnitt wird demonstriert wie eine Anforderung den Hauptzyklus der Evaluation durchläuft. Die dafür notwendigen Beschreibungen finden sich in Abschnitt 4.4 ab Seite 100. Der vorliegende Abschnitt wird nicht analog zur Beschreibung des Verfahrens aus Abschnitt 4.4 strukturiert, in welchem erst der Hauptzyklus als ganzes beschrieben und dann auf einzelne Aktivitäten detailliert eingegangen wird. Stattdessen werden in den anschließenden Ausführungen die Aktivitäten so durchlaufen, wie es in einem Bewertungsvorgang einer Anforderung erforderlich ist. Der Durchlauf des Hauptzyklus der POSAAM-Evaluation wird exemplarisch des Durchlauf eines einzelnen Zyklus für die Anforderung Q5 beschrieben.

Tabelle 5.2.: Prüfung der Architekturentscheidungen

Architektur-entscheidungs-kürzel	Alternative ersichtlich	Begründung vorhanden	Auf Anforderung zurückzuführen
AE1	kein Komponentenmodell	Erweiterbarkeit des Rahmenwerks	F7, Q7
AE2	parallele Abarbeitung von Aufträgen	leichte/schnelle Entwicklung	-
AE3	nur Rollen- und Rechtesystem aus dem Webservice	Benötigte Rollen- und Rechte nicht deckungsgleich mit vorhandenen Rollen und Rechten	Q1a, Q1b, Q1c
AE4	Anwendung läuft vollständig auf Clientseite	keine Installation auf Client notwendig; zentrales Rechtesystem; zentrales Accounting; zentrales Reporting; weitere zentrale Funktionalität	F6, E2, Q1a, Q1b, Q1c
AE5	direkter Zugriff auf Datenbank	leichte/schnelle Entwicklung	-

Mustermenge zur Umsetzung der Anforderung ableiten. Zu jeder Anforderung, die den Hauptzyklus durchläuft, wird in der ersten Aktion eine Menge von Mustern gewonnen, die zur Umsetzung der aktuell betrachteten Anforderung geeignet sein könnten. Aus Tabelle 5.1 auf Seite 150 geht hervor, dass durch die Anforderung Q5 das Qualitätsmerkmal Effizienz, Performanz gefordert wird. Zu diesem Qualitätsmerkmal liegen in Anhang B ab Seite 182 drei beispielhafte Einträge nach den Vorgaben des POSAAM-Wissensmodells für die Architekturmuster „Caching“, „Pooling“ und „Eager Acquisition“ vor. Durch Betrachtung der Anforderung Q5 und der gegebenen Architekturspezifikation geht hervor, dass die Anforderung Q5 an das Verhalten von Batch-Frame gebunden ist, welches durch die in der Laufzeitsicht der Architektur beschriebenen Schritte beschrieben wird. Durch Betrachtung der Schritte kann gefolgert werden, dass die Komponenten GUI, Application Manager, Business, Database, Building Block und Session Manager sowie die Interaktionen zwischen diesen Komponenten für den Einsatz der identifizierten Muster zu betrachten sind. Aus der Betrachtung der Kontexte der Muster geht hervor, dass der Einsatz aller drei Architekturmuster zur Beeinflussung des geforderten Qualitätsmerkmals bzw. zur Erfüllung der Anforderung Q5 genutzt werden könnten.

Identifikation von Mustern aus der Menge in der Architekturbeschreibung. Da die Menge von Mustern, die aus der Aktion „Mustermenge zur Umsetzung der Anforderung ableiten“ resultiert, nicht leer ist, wird mit der Identifikation

von Mustern aus der Menge in der vorliegenden Architekturbeschreibung fortgefahren. Das Architekturmuster „Caching“ kann an zwei Stellen identifiziert werden. Aus der Beschreibung der Komponente GUI in der technischen Sicht auf die Architektur von Batch-Frame geht hervor, dass die vom Server übermittelten Informationen in der Komponente GUI zwischengespeichert werden. Dies deutet auf die Nutzung des Architekturmusters „Cache“ hin. Allerdings können nicht alle wiederkehrenden Elemente in der Architekturbeschreibung getrennt identifiziert werden. Sowohl das Element „Ressource User“ als auch das Element „Ressource Cache“ als auch das Element „Ressource“ werden in der Komponente GUI gekapselt. Lediglich das Element „Ressource Provider“ kann der Komponente Application Manager zugeordnet werden. Bei der noch durchzuführenden Prüfung der identifizierten Muster (s.u.) werden diese Mängel in der Zuordnung festgehalten.

Das Architekturmuster „Caching“ kann an einer weiteren Stelle in der Architekturbeschreibung von Batch-Frame identifiziert werden. Für die Interaktion mit dem Webservice ist eine Identifikation und Authentifikation erforderlich. Diese geschieht durch einen separaten Aufruf des Webservice, welcher ein Session Token zurück liefert. Mit diesem Session Token kann anschießend ein Aufruf des Webservice erfolgen, bei welchem weiter gehende Funktionen des Webservice genutzt werden können. Aus der Beschreibung der Architektur von Batch-Frame geht hervor, dass das Session Token nicht für jeden Aufruf des Webservice erneut erworben wird. Stattdessen wird das Session Token zwischengespeichert und bei jedem Aufruf erneut verwendet. Es handelt sich um einen Cache[3]. Die Elemente „Ressource User“ sind die Komponenten Application Manager sowie Building Block, das Element „Ressource Cache“ ist die Komponente Session Manager, das Element „Ressource“ ist das Session Token und das Element „Ressource Provider“ ist der Webservice.

Weder das Architekturmuster „Pooling“ nach das Architekturmuster „Eager Acquisition“ können in der Architekturbeschreibung von Batch-Frame identifiziert werden. Die Beschreibungen zur Identifikation alternativer Architekturmechanismen erfolgt ab Seite 156. Zunächst wird mit der Prüfung der identifizierten Muster fortgefahren.

Prüfung identifizierter Muster

Die identifizierten Muster werden jeweils einzeln geprüft. Die Grundlage für das Vorgehen zur Prüfung identifizierter Muster ist in Abschnitt 4.4.2 auf den Seiten

[3] Die Nutzung dieses Caches ist durch das Protokoll zur Interaktion mit dem Webservice zwar nahe liegend, aber dennoch handelt es sich um einen Cache.

105ff. hinterlegt. Für die Durchführung der Prüfungen wird erneut auf die Einträge über die Architekturmuster, die in Anhang B, Abschnitt B.1, auf den Seiten 182ff. exemplarisch hinterlegt sind, zurückgegriffen. Die Prüfung identifizierter Muster untergliedert sich in die Aktionen „Prüfung der Wahl des geeignetsten Musters“, „Prüfung der Auswirkungen auf Qualitätsmerkmale“, „Prüfung der korrekten Instanzierung des Musters“ und „Prüfung der Beziehungen zu weiteren Mustern“.

Prüfung der Wahl des geeignetsten Musters. Eine mögliche Alternative zum Architekturmuster „Caching“ ist das Architekturmuster „Pooling“. Der Einsatz des Architekturmusters „Pooling“ wäre für beide identifizierten Fälle des Architekturmusters „Caching“ nicht geeignet gewesen, da beim Architekturmuster „Pooling“ im Gegensatz zum Architekturmuster „Caching“ die wiederverwendeten Ressourcen keine zwischengespeicherte Kopie der Originalressource darstellen, sondern nach jeder Nutzung (neu initialisiert) wiederverwendet werden (vgl. Abschnitt B.1.2 auf den Seiten 185ff.).

Prüfung der Auswirkungen auf Qualitätsmerkmale. Für das zu prüfende Architekturmuster „Cache“ ist in der Wissensbasis hinterlegt, dass es die Speichereffizienz negativ beeinflusst. In Batch-Frame wurden keine Anforderungen an die Speichereffizienz gestellt. Aus diesem Grund werden in dieser Aktion keine Risiken festgestellt.

Prüfung der korrekten Instanzierung des Musters. Bei der Prüfung der korrekten Instanzierung des Musters werden zunächst die wiederkehrenden Anteile des Musters geprüft. Bei der Identifikation von Mustern in der Architekturbeschreibung wurde bereits festgestellt, dass die Beschreibung der Instanzierung des Musters „Cache“ innerhalb der Komponente GUI unvollständig ist. Im Evaluationsbericht wird aufgenommen, dass die Architekturbeschreibung an dieser Stelle nicht ausreichend detailliert ist. Die Instanzierung des Musters muss vollständig beschrieben werden. Auch die Beschreibung der variierenden Anteile des Musters fehlt.

Bei der Beschreibung der Instanzierung des Musters „Cache“ bezüglich des Zugriffs auf den Webservice können alle wiederkehrenden Elemente identifiziert werden. Auch die Topologie und Interaktionen der Elemente sind nachvollziehbar beschrieben. Bei der Prüfung der variierenden Anteile des Musters wird zunächst die Variationsmöglichkeit „Strategie zur Haltung von Daten im Cache“ geprüft. Im vorliegenden Fall wird immer nur ein Session Token zwischengespeichert. Dieses wird immer dann aufgegeben (d.h. aus dem Cache gelöscht), wenn ein

Auftrag komplett abgearbeitet wurde. Die Belegung der Variationsmöglichkeit ist demnach in der Architekturspezifikation ersichtlich und es handelt sich bei der Belegung auch um eine gültige Belegung der Variationsmöglichkeit. Bei der Prüfung der Auswirkung der Belegung auf Qualitätsmerkmale wird festgestellt, dass das zwischenspeichern des Session Tokens für die Speichereffizienz kaum relevant ist, da es sich um ein sehr geringes Datenvolumen handelt, welches zwischengespeichert werden muss. Außerdem gibt es keine Anforderung, die auf die Speichereffizienz abzielt. Die Prüfung der Variationsmöglichkeit „Form der Synchronisation zwischen Original und Cache“ ergibt, dass keine Synchronisation zwischen Original und Cache stattfindet. Verfällt die Identifikation und Authentifikation beim Webservice (z.B. durch einen Timeout) so kann das Session Token nicht mehr verwendet werden (die Nutzung des Webservice misslingt). Eine Identifikation und Authentifikation muss erneut stattfinden. Dies ist eine gültige Belegung der Variationsmöglichkeit. Die Prüfung der Auswirkung der Belegung auf Qualitätsmerkmale ergibt, dass aufgrund des Volumens der zwischengespeicherten Daten kaum eine Auswirkung auf ein Qualitätsmerkmal existiert. Es werden keine Einträge im Evaluationsbericht vorgenommen.

Prüfung der Beziehungen zu weiteren Mustern. Im exemplarischen Eintrag zum Architekturmuster „Cache“ in die Wissensbasis in Anhang B, Abschnitt B.1 ab Seite 182 ist eine Beziehung zum Architekturmuster „Evictor“ hinterlegt. Dieses Architekturmuster wird bei der Organisation der Strategien zur Freigabe von Ressourcen aus dem Cache verwendet. Da die Freigabe von Ressourcen beim ersten identifizierten Einsatz des Architekturmusters „Cache“ nicht beschrieben wird, kann die Notwendigkeit des Einsatzes nicht geprüft werden. Beim zweiten identifizierten Einsatz des Architekturmusters „Cache“ ist die Freigabe von Ressourcen einfach gehandhabt und benötigt keine gesonderte Strategie. Der Einsatz des Architekturmusters „Evictor“ ist für den beschriebenen Fall also nicht nötig.

Identifikation alternativer Architekturmechanismen

Beim Ableiten einer Mustermenge zur Umsetzung der Anforderung Q5 wurden zusätzlich zum Architekturmuster „Caching“ auch die Architekturmuster „Pooling“ und „Eager Acquisition“ vorgeschlagen, welche bei der anschließenden Identifikation in der Architekturbeschreibung nicht identifiziert werden konnten, obwohl sowohl der Einsatz des „Pooling“ Architekturmusters für die Verbindungen zum externen Webservice als auch der Einsatz des „Eager Acquisition“ Architekturmusters beim Ansprechen der Datenbank bzw. der Technologie für die objektrelationale Abbildung eine positive Beeinflussung des Qualitätsmerkmals Performanz zur Folge gehabt hätten (vgl. Abschnitte B.1.2 und B.1.3 auf den

Seiten 185ff. sowie 186ff.). Im Anschluss wird demonstriert, wie die POSAAM-Evaluation in diesem Fall verläuft.

Identifikation von Prinzipien über Muster. Da die Suche nach alternativen Architekturmechanismen erfolgt, weil ein erwartetes Architekturmuster nicht identifiziert werden konnte, kann das erwartete Architekturmuster für die Suche nach Prinzipien herangezogen werden. Im exemplarischen Eintrag in die Wissensbasis für das Architekturmuster „Pooling“ (vgl. Abschnitt B.1.2 ab Seite 185) ist hinterlegt, dass diesem Muster das Prinzip „Verlagerung von Ressourcenbedarf“ (vgl. Abschnitt B.2.3 ab Seite 189) zugrunde liegt, welches wiederum die Regeln des Prinzips „Entlastung einer Ressource“ (vgl. Abschnitt B.2.2 ab Seite 189) befolgt. Beim Durchlauf der Aktion „Suche nach Einsatz von Prinzipien“, der für das Architekturmuster „Pooling“ durchgeführt wird, kann nach diesen beiden Prinzipien gesucht werden.

Im exemplarischen Eintrag in die Wissensbasis für das Architekturmuster „Eager Acquisition“ (vgl. Abschnitt B.1.3 ab Seite 186) ist hinterlegt, dass diesem Muster das Prinzip „Verlagerung der zeitlichen Auslastung von Ressourcen“ (vgl. Abschnitt B.2.4 ab Seite 189) zugrunde liegt, welches wiederum die Regeln des Prinzips „Verlagerung von Ressourcenbedarf“ (vgl. Abschnitt B.2.3 ab Seite 189), welches wiederum die Regeln des Prinzips „Entlastung einer Ressource“ (vgl. Abschnitt B.2.2 ab Seite 189) befolgt. Beim Durchlauf der Aktion „Suche nach Einsatz von Prinzipien“, der für das Architekturmuster „Eager Acquisition“ durchgeführt wird, kann nach diesen drei Prinzipien gesucht werden.

Suche nach Einsatz von Prinzipien. Die Prinzipien „Verlagerung von Ressourcenbedarf“ und „Entlastung einer Ressource“ regeln die Eigenschaften „knappe Ressourcen“, „Verwendung der Ressource“ und „Alternative Ressource“. Bei der Suche nach alternativen Architekturmechanismen für das Architekturmuster „Pooling“ ist die knappe Ressource in Batch-Frame die Verbindung zum Externen Webservice. Die Verwendung der Ressource wird in Batch-Frame nicht entlastet. Für jede Aktion, die durch Batch-Frame durchzuführen ist, wird eine neue Verbindung zum externen Webservice aufgebaut. Es wird keine alternative Ressource belastet. Es kann also geschlossen werden, dass keine Alternative für das Architekturmuster „Pooling“ in der Architektur von Batch-Frame eingesetzt wurde. Dies wird als Risiko festgehalten.

Analog verhält es sich bei der Suche nach alternativen Architekturmechanismen für das Architetkurmuster „Eager Acquisition“. Das Prinzip „Verlagerung der zeitlichen Auslastung von Ressourcen“ regelt die Eigenschaften „knappe Ressource“ und „Zeitliche Verwendung der Ressource“. Im Falle des Einsatzes des

Architekturmuster „Eager Acquisition“ ist die knappe Ressource durch die Daten aus der Datenbank repräsentiert. Der Zugriff auf die Datenbank geschieht bei Batch-Frame immer dann, wenn auf die Daten innerhalb der Technologie, die die objektrelationale Abbildung zur Verfügung stellt, zugegriffen wird. Der Zeitpunkt des Zugriffs wird also nicht verlagert. Auch die Belastung einer alternativen Ressource oder eine sonstige Entlastung der Ressource kann nicht festgestellt werden. Es kann also erneut geschlossen werden, dass keine Alternative für das Architekturmuster „Eager Acquisition“ in der Architektur von Batch-Frame eingesetzt wurde. Auch dies wird als Risiko festgehalten.

Die Aktionen zum weiteren Verfahren (z.B. bezüglich einer Veränderung der Architektur oder der Anforderungen) werden an dieser Stelle nicht erläutert. Sie stellen nicht den Kern von POSAAM dar. Außerdem sind die Aktionen im Rahmen des Unternehmens, welches Batch-Frame entwickelt, durchzuführen.

5.5. Erkenntnisse aus der Fallstudie Batch-Frame

Nachfolgend werden die zentralen Erkenntnisse, die aus der Durchführung der Fallstudie Batch-Frame gewonnen werden konnten, aufgezählt.

Identifikation von Mustern möglich. Trotz der informellen Architekturbeschreibung ist es möglich, den Einsatz von Architekturmustern zu identifizierten. Durch die Identifikation von Architekturmustern können weitere Fragen bezüglich der Konfiguration und der weiteren Auswirkungen auf die verbleibende Architektur systematisch gestellt werden. Die Identifikation von Architekturmustern und die daraus resultierenden Fragen führen entweder zu einem gesteigerten Verständnis des Einflusses der Architektur auf Qualitätsmerkmale oder decken Lücken in der Dokumentation auf.

Arbeit mit Prinzipien möglich. Wenn keine Architekturmuster identifiziert werden, kann durch die Suche nach Eigenschaften, die durch Prinzipien geregelt werden, die Verwendung einer Alternative identifiziert oder ausgeschlossen werden. Der Ausschluss oder die Identifikation einer Alternative führt wiederum zu einem gesteigerten Verständnis des Einflusses der Architektur auf Qualitätsmerkmale. In der Fallstudie Batch-Frame wird die bei der Suche nach alternativen Architekurmechanismen zu den Architekturmustern „Pooling“ und „Eager Acquisition“ deutlich.

Transzendentes Qualitätsverständnis. Das Verständnis von Qualität ist bei den Herstellern von Batch-Frame transzendent [Gar84]. Nach diesem Verständnis gibt es Architekturen, die unabhängig von den an die Architektur gestellten Anforderungen gut oder schlecht sind. Dass bei den Herstellern von Batch-Frame ein transzendentes Verständnis der Qualität von Softwarearchitekturen haben, wird bei Betrachtung der verfügbaren Beschreibungen an zwei Stellen deutlich:

1. Es wurden Architekturentscheidungen getroffen, die auf Anforderungen deuten, die jedoch nicht explizit in die Anforderungen aufgenommen wurden (z.B. die Notwendigkeit der schnellen Entwicklung des Systems in Architekturentscheidungen AE2 und AE5).

2. In der Architekturbeschreibung können Architekturmuster identifiziert werden, die nicht auf explizit genannte Qualitätsanforderungen zurückzuführen sind. Beispielsweise können in Batch-Frame die Muster „MVC“ [BMR+96] sowie „Layers“ [BMR+96] identifiziert werden. Die leichte Anpassung der Benutzerschnittstelle (die durch den Einsatz des MVC-Musters begünstigt wird) wurde in den Anforderungen nicht explizit erfasst.

Auch in Gesprächen mit den Herstellern von Batch-Frame wurde das transzendente Qualitätsverständnis deutlich. So empfanden die Hersteller von Batch-Frame die Gestaltung der Architektur in Form einer Schichtenarchitektur als „saubere Arbeitsweise“, die eine Entwicklung von Softwaresystemen hoher Qualität erst ermögliche.

KAPITEL 6

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde die Architekturbewertungsmethode POSAAM vorgestellt. Nachfolgend wird der Kern des Inhalts der Arbeit zusammengefasst. Anschließend werden Ansatzpunkte für weitere Entwicklungen und Forschung gegeben.

Inhalt

6.1. Zusammenfassung

Qualitative Formen der Architekturbewertung basieren auf der Prüfung des Einsatz von Architekturmechanismen (Formen der Strukturierung von Systemen oder Teilen von Systemen) von denen bekannt ist, dass diese einen erwünschten Einfluss auf geforderte Qualitätsmerkmale haben. Es handelt sich in anderen Worten um die Prüfung des Einsatzes von Expertenwissen. Prüfungen dieser Art werden mit Hilfe von Fragebögen, Checklisten oder durch gesteuerte Formen von Reviews, wie z.B. die Steuerung durch den Einsatz von Szenarien, durchgeführt. Beim Einsatz von Fragebögen und Checklisten ist das Expertenwissen in diesen Hinterlegt. Allerdings sind diese Formen der Qualitätssicherung allgemeiner Natur, d.h. sie sind nicht auf die zu evaluierende Anwendung ausgelegt, sondern werden für alle Anwendungen in gleicher Form durchgeführt. Bei szenariobasierten Verfahren wird die Prüfung durch die Formulierung von Szenarien auf die spezifischen Anforderungen für die zu evaluierende Anwendung ausgerichtet. Bei bisher existierenden szenariobasierten Verfahren beruht die Evaluierung auf der subjektiven Einschätzung der Eignung der Architektur durch Experten. Eine systematische Vorgehensweise bei der Evaluation wird in existierenden Verfahren nur für die nachträgliche Erhebung von Anforderungen sowie die nachträgliche Architekturdokumentation gegeben. Die resultierenden Ergebnisse sind nicht objektiv und die Nachvollziehbarkeit hängt von der Argumentation der Experten ab. Verfahren zur Evaluation von Softwarearchitekturen wurden in Kapitel 3 dargelegt und in Abschnitt 3.4 gegenübergestellt und kritisch betrachtet.

In der vorliegenden Arbeit wurde das qualitative Architekturbewertungsverfahren POSAAM (**P**attern **O**riented **S**oftware **A**rchitecture **A**nalysis **M**ethod) erarbeitet. Bei diesem Verfahren wird das in Architekturmustern hinterlegte Expertenwissen über die Strukturierung von Systemen oder Teilen von Systemen genutzt, um die Evaluation einer Softwarearchitektur zu unterstützen. Zu diesem Zweck wurde im Rahmen der Arbeit eine neue Definition von Mustern und Architekturmustern erarbeitet. Bei der neuen Definition von Mustern wurde besonderer Wert darauf gelegt, dass Muster aus wiederkehrenden und variierenden Anteilen bestehen. Eine Definition von Architekturmustern wie sie in dieser Arbeit verstanden werden und eine Abgrenzung zu gängigen Definitionen wurde in Abschnitt 2.3.1 auf Seite 27 gegeben.

Für Sonderfälle, in welchen eine Evaluation anhand von Mustern nicht durchführbar ist, wurde das Konzept des Prinzips eingeführt. Analog zu einem Architekturmuster handelt es sich bei einem Prinzip um eine anerkannte Weise, um Qualitätsmerkmale durch die Strukturierung eines Systems zu beeinflussen. Der Unterschied besteht darin, dass bei einem Architekturmuster konkrete Strukturen mit Angaben über vorhandene Elemente, deren Verantwortlichkeiten, die Topologie der Elemente (Möglichkeiten der Interaktion) und die Formen

der Interaktion existieren, während Prinzipien aus abstrakten Regeln bestehen. In einer Ontologie (Abschnitt 4.2.1 auf Seite 76) für die Architekturbewertung mit POSAAM wurde u.a. der Zusammenhang zwischen Architekturmustern und Prinzipien dargelegt.

Für die Anwendung der Methode POSAAM wurde ein Wissensmodell definiert, in welchem Architekturmuster, Prinzipien und Qualitätsmerkmale zueinander in Beziehung gesetzt werden. Die Zuordnung von Prinzipien zu Qualitäts(teil)merkmalen beinhaltet das Qualitätsmodell für die Nutzung von POSAAM. Durch die Nutzung des Wissensmodells können Qualitätsmerkmale, die in Anforderungen gefordert werden, auf systematische Weise zu Architekturmustern oder aber zu Prinzipien in Bezug gesetzt werden. Durch diese Systematik wird die Nachvollziehbarkeit der Methode erhöht. Durch die Zuordnung der Anforderungen zu anerkannten Prinzipien wird die Objektivität der Evaluation erhöht. Architekturmuster stehen zu weiteren Architekturmustern in Beziehung. Die Beziehungen der Architekturmuster untereinander werden auch im Wissensmodell abgelegt. Die Nutzung der Beziehungen zwischen den Architekturmustern im Rahmen einer Evaluation ist in der Systematik von POSAAM integriert und erhöht die Nachvollziehbarkeit von Evaluationsergebnissen. Die Erarbeitung des Wissensmodells für POSAAM wurde in Abschnitt 4.2.2 vorgenommen. Die Erarbeitung des Evaluationsverfahrens POSAAM, welches die systematische Nutzung des in einer Wissensbasis hinterlegten Expertenwissens über Architekturmuster und Prinzipien beinhaltet, wurde in Kapitel 4 dargelegt.

Im Rahmen einer Fallstudie innerhalb eines Softwareentwicklungsprojekts in einem mittelständischem Unternehmen konnte die Anwendung einiger zentraler Konzepte von POSAAM demonstriert werden. Eine umfassende Durchführung einer vollständigen POSAAM-Evaluation wird erst nach mehreren Iterationen von POSAAM-Evaluationen möglich sein, da POSAAM auf der Nutzung einer vorhandenen Wissensbasis aufbaut, die mit jeder Anwendung von POSAAM weiter verfeinert und ausgebaut wird. Solange eine solche Wissensbasis nicht vorliegt, ist das Expertenwissen, welches von Evaluationsmitgliedern einzubringen ist, noch ähnlich hoch wie bei vergleichbaren qualitativen Architekturbewertungsmethoden. Die Demonstration der zentralen Konzepte von POSAAM anhand der Fallstudie wurde in Kapitel 5 dokumentiert. Mögliche Formen der umfassenderen Validierung und der Einordnung von POSAAM zu anderen Methoden, die durchgeführt werden können sobald eine umfassendere Wissensbasis vorliegt, werden im nachfolgenden Ausblick vorgeschlagen.

6.2. Ausblick

Für die weitere Arbeit mit POSAAM gibt es Ansatzpunkte von denen einige in diesem Abschnitt aufgezeigt werden. Die Entwicklung geeigneter Werkzeugunterstützung für die Arbeit mit dem Wissensmodell (Abschnitt 6.2.1) stellt eine Voraussetzung für den Einsatz im industriellen Gebrauch dar. Durch die Verwendung von geeigneten Werkzeugen wird auch erst der Aufbau einer umfassenden Wissensbasis (Abschnitt 6.2.2) möglich, durch die die Abhängigkeit von Experten bei der Durchführung von Evaluationen vermindert werden kann. Erst nach der Schaffung dieser Voraussetzungen ist eine umfassendere Validierung der Methode POSAAM möglich. Ansatzpunkte hierzu werden in Abschnitt 6.2.3 gegeben. Das in dieser Arbeit erarbeitete Wissensmodell kann um weitere Merkmale erweitert werden. In Abschnitt 6.2.4 werden hierzu einige Vorschläge gemacht.

6.2.1. Werkzeugunterstützung

Für den Zugriff auf und die Pflege von Daten aus der Wissensbasis ist die Entwicklung eines Werkzeugs wichtig, da die Pflege von Daten über Muster und Prinzipien anhand von Dokumenten mit textuellen Verweisen für Menschen schnell unübersichtlich wird. Denkbar ist zu diesem Zweck eine Anwendung, in welcher die Daten in einer Datenbank abgelegt werden und durch Eingabe- und Abfragemasken abgerufen und verändert werden können.

Zu einer weiter gehenden Werkzeugunterstützung würden Werkzeuge gehören, die das gesamte Verfahren der Evaluation unterstützen. Denkbar wäre eine Unterstützung der Evaluation angefangen bei der Möglichkeit der Zuordnung der Anforderungen zu Qualitätsmerkmalen über die Unterstützung der systematischen Suche nach Architekturmechanismen für die Realisierung der Anforderungen über Vorschläge für geeignete Architekturmuster oder Prinzipien sowie die Prüfung der daraus resultierenden notwendigen Architekturentscheidungen (Beispielsweise die Belegung von Variationsmöglichkeiten von Mustern) über das Durchlaufen jeder Aktion im Rahmen der Evaluation zusammen mit der Aufnahme von Ergebnissen, bis hin zur Erzeugung eines Rahmens für den Evaluationsbericht.

6.2.2. Aufbau einer umfassenden Wissensbasis

Existierende Musterbeschreibungen beinhalten Informationen über den Zusammenhang zwischen Qualitätsmerkmalen und Mustern sowie über den Zusammenhang zwischen Qualitätsmerkmalen und möglichen Ausprägungen der Muster

(Konfigurationen). Diese Informationen sind jedoch nur implizit in den Beschreibungen enthalten. Um diese Informationen für eine Evaluation nach POSAAM nutzbar zu machen, muss die Information in das erarbeitete Wissensmodell übergeführt und dort explizit hinterlegt werden. Auch Informationen zu Architekturmustern, die in den bereits existierenden Musterbeschreibungen nicht existieren, müssen bei der Überführung in das Wissensmodell ergänzt werden (z.B. der Zusammenhang zu Prinzipien).

Der Aufbau einer Wissensbasis durch die Überführung von Informationen, die in existierenden Musterbeschreibungen vorhanden sind, kann durch den Aufbau der Wissensbasis im Rahmen der Durchführung von Evaluationen ergänzt werden.

6.2.3. Langfristige umfassende Validierung

Die in Kapitel 5 durchgeführte Fallstudie zeigt die grundsätzliche Anwendbarkeit der durch POSAAM vorgeschlagenen Konzepte. Da jedoch bei der Durchführung der Fallstudie keine umfassende Wissensbasis vorlag und keine Werkzeugunterstützung existierte, war eine umfassende Validierung der Methode nicht möglich. Bei vorliegen einer geeigneten Werkzeugunterstützung und nach dem Aufbau einer umfassenden Wissensbasis kann die Methode POSAAM in der Form angewendet werden, nach welcher POSAAM konzipiert wurde. Sind diese Voraussetzungen gegeben, kann auch eine Validierung vorgenommen werden, die umfassender ist, als die Validierung, die in Kapitel 5 vorgenommen wurde. Im Rahmen einer umfassenden Validierung können zusätzlich zur grundsätzlichen Anwendbarkeit der Konzepte POSAAMs noch folgende Punkte untersucht werden:

1. In den Abschnitten 4.6.1 und 4.6.2 wird die Methode POSAAM u.a. bezüglich der Kriterien „Erfahrungen und Kenntnisse des Evaluationsteams“, „Systematik und Vorhersehbarkeit bzw. Wiederholbarkeit“, „Nachvollziehbarkeit“, „Weiterentwicklung und Pflege“ und „Einbettung in den Softwareentwicklungsprozess“ argumentativ eingeordnet. Ist die Methode POSAAM länger im Einsatz kann die argumentative Einordnung mit empirischen Daten unterlegt werden.
2. Des Weiteren kann Untersucht werden, welche Aussagekraft die von POSAAM erzeugten Ergebnisse haben. Dieser Punkt untergliedert sich in zwei weitere Punkte:
    a) Es kann untersucht werden, inwieweit die von POSAAM erzeugten Ergebnisse bei Beachtung innerhalb des Softwareentwicklungsprozesses einen Mehrwert für das Projekt darstellen. In anderen Worten, ob und inwieweit durch POSAAM die Qualität der Architektur des zu

entwickelnden Systems gesteigert oder Kosten bei der Entwicklung des Systems gesenkt werden können.

b) Ein weiterer Punkt, der bezüglich der Aussagekraft von POSAAM untersucht werden kann, ist, inwieweit durch die Anwendung von POSAAM potentielle Missstände der Architektur nicht aufgedeckt werden.

Um die Aussagekraft der durch POSAAM erzeugten Ergebnisse zu untersuchen (Punkt 2), können Architekturbewertungen von Systemen durchgeführt werden, die sich bereits im Einsatz befinden und von denen bekannt ist, dass sie den Qualitätsanforderungen nicht entsprechen. Die Ergebnisse solcher Evaluationen können Aufschluss darüber geben, inwieweit durch POSAAM die Missstände einer Architektur aufgedeckt werden (Punkt 2b). Um an einem solchen System zu untersuchen, inwieweit die durch die Evaluation gelieferten Ergebnisse (zu einem durch POSAAM identifizierten potentiellen Missstand werden Alternativen in Form von Mustern oder Prinzipien vorgeschlagen) dazu beitragen können, die Qualität des Systems zu verbessern, müssten die durch POSAAM vorgeschlagenen Alternativen umgesetzt werden. Nach der Umsetzung können die Ausprägungen des Qualitätsmerkmals, welches verbessert wurde, mit den Ausprägungen des Qualitätsmerkmals vor der Verbesserung verglichen werden.

Nachfolgend werden die Möglichkeiten der Validierung bezüglich der Kriterien aus Punkt eins einzeln betrachtet.

Systematik und Vorhersehbarkeit bzw. Wiederholbarkeit

Da für die Durchführung einer POSAAM-Evaluation die Architekturspezifikation sowie die Anforderungsspezifikation bereits vor Durchführung der Evaluation vorliegen müssen, ist es möglich, dieselbe Evaluation (mit denselben Eingaben) mehrmals durchzuführen. Dadurch ließen sich empirische Erkenntnisse bezüglich der Wiederholbarkeit gewinnen. Die Wiederholbarkeit ist ein Merkmal einer Architekturbewertungsmethode, welche den Grad der Überdeckung der Ergebnisse bei mehrmaliger Durchführung der gleichen Evaluation repräsentiert. Für POSAAM wäre ein geeignetes Maß für den Grad der Überdeckung die Anzahl derselben durch unterschiedliche Evaluationsteams[1] identifizierten Architekturmuster und -konfigurationen, die Anzahl derselben durch unterschiedliche Evaluationsteams identifizierten Architekturmechanismen (welchen dasselbe Prinzip zugrunde liegt) sowie die Anzahl derselben durch unterschiedliche Evaluationsteams identifizierten fehlenden Architekturmuster, Musterkonfigurationen oder Architekturmechanismen.

[1] Wobei die Evaluationsteams für die Ermittlung der Wiederholbarkeit über möglichst Ähnliches Expertenwissen verfügen sollten.

Um die Wiederholbarkeit von POSAAM bezüglich der Wiederholbarkeit von ATAM zu positionieren, muss ein Maß für die Wiederholbarkeit von ATAM ermittelt werden. Da bei einer ATAM-Evaluation Stakeholder des zu Evaluierenden Systems beteiligt sein müssen, ist die mehrmalige Durchführung der gleichen ATAM-Evaluation unter gleichen Voraussetzungen nicht möglich, außer wenn Stakeholder gleicher Natur durch verschiedene Personen repräsentiert würden (z.B. verschiedene Personen aus der Marketing-Abteilung). Alternativ wäre eine Aufzeichnung der ersten Schritte (z.B. durch Aufzeichnung von Bild und Ton) von ATAM denkbar, so dass lediglich der vierte, sechste und achte Schritt von ATAM mit dem Vorgehen von POSAAM verglichen wird. Obwohl in diesem Fall nicht die gesamte Methode durchgeführt würde, ist ein Vergleich der beiden Methoden dennoch sinnvoll, da POSAAM keine Erhebung der Architektur- und Anforderungsspezifikation vorsieht und somit eher den Schritten vier, sechs und acht von ATAM entspricht. Das Maß für die Wiederholbarkeit würde bei ATAM auf die Zahl derselben identifizierten Architekturmechanismen reduziert werden, da eine explizite Zurückführung auf Architekturmuster in ATAM nicht vorgesehen ist.

Erfahrungen und Kenntnisse des Evaluationsteams

Auch den Grad der Erfahrungen und Kenntnisse die notwendig sind, um eine POSAAM-Evaluation durchzuführen ließe sich über Grad der Überdeckung der Evaluationsergebnisse ermitteln. Für eine empirische Validierung müsste eine mehrmalige POSAAM-Evaluation mit denselben Eingaben durch unterschiedlich erfahrene Evaluationsteams durchgeführt werden. Die Erfahrung der Evaluationsteams könnte an der Anzahl der entworfenen Architekturen/durchgeführten Architekturbewertungen, an der Anzahl der entworfenen Architekturen/durchgeführten Architekturbewertungen von Architekturen aus der gleichen Domäne, an der Anzahl der entworfenen Architekturen/durchgeführten Architekturbewertungen zu dem/den geforderten Qualitätsmerkmal(en) sowie an der Anzahl der bekannten Architekturmuster/-mechanismen zu dem/den geforderten Qualitätsmerkmal(en) differenziert werden. Werden Erfahrung und Kenntnisse des Evaluationsteams zum Grad der Überdeckung in Relation gestellt, kann daraus ein Maß für die Expertenunabhängigkeit gewonnen werden.

Der Vergleich mit der Expertenunabhängigkeit von ATAM muss aufgrund der Einbeziehung von Stakeholdern in die ATAM-Evaluation analog zum Vergleich bezüglich der Wiederholbarkeit geschehen. Die Erfahrung des Evaluationsteams kann hingegen sowohl für ATAM als auch für POSAAM in gleicher Weise erfolgen.

Weiterentwicklung und Pflege

Der Vergleich der Weiterentwicklung und Pflege zwischen ATAM und POSAAM kann nur im Sinne einer Klassifikation erfolgen (vgl. hierzu Abschnitt 4.6.2), da die Weiterentwicklung und Pflege in POSAAM im Rahmen der Wissensbasis erfolgt, die in ATAM nicht existiert.

Nachvollziehbarkeit

In [CKK02, S. 82, 83] schlagen *Clements et al.* vor, längere Zeit (in etwa sechs Monate) nach Abschluss einer ATAM Evaluation Umfragen durchzuführen, mit denen der Nutzen der Evaluation durch beteiligte Architekten und Entwickler rückblickend geschätzt werden kann. Um das Kriterium der Nachvollziehbarkeit zu messen, ist ein ähnliches Verfahren denkbar. Durch Umfragen kann geprüft werden, ob die Architekten der Architekturen, die evaluiert wurden, die Ergebnisse der Evaluation nachvollziehen können (die Architekten verstehen, weshalb das Ergebnis produziert wurde und sind damit einverstanden, dass das produzierte Ergebnis zurecht produziert wurde).

Einbettung in den Softwareentwicklungsprozess

Analog zur Messung und zum Vergleich der Nachvollziehbarkeit kann auch die Einbettung in den Softwareentwicklungsprozess über Umfragen ermittelt werden. Allerdings müssten Befragungen von Projektleitern erfolgen, die bereits Erfahrungen mit beiden Evaluationsmethoden gemacht haben. Metriken, die sich zur Messung der Einbettung in den Softwareentwicklungsprozess eignen, sind die Anzahl der im Laufe des Softwareentwicklungsprozesses verwendeten Ergebnisse aus dem Evaluationsbericht.

6.2.4. Erweiterungen des Wissensmodells

Für die vorliegende Arbeit wurde das Wissensmodell so gestaltet, dass die Zusamenhänge zwischen Qualitätsmerkmalen, Prinzipien und Mustern hergestellt werden können. Die Hinterlegung zusätzlicher, im Rahmen einer Architekturevaluation potentiell nützlichen Information wurde nicht betrachtet bzw. an geeigneten Stellen lediglich angedeutet. Im Anschluss wird motiviert, welche zusätzlichen Informationen im Wissensmodell hinterlegt werden können und wie sie in den Evaluationsvorgang Eingang finden könnten.

Informationen für quantitative Analysen

In Abschnitt 4.5.2 wurde im Rahmen des Übergangs zu quantitativen Architekturbewertungsmethoden bereits vorgeschlagen, für jedes Muster Informationen für die Anwendung von quantitativen Architekturbewertungsmethoden (ähnlich den Verfahren in den Attribute-Based Architectural Styles [KK99, KKB+99]) im Wissensmodell zu hinterlegen. Da Muster in POSAAM aus wiederkehrenden sowie variierenden Anteilen bestehen, können im Wissensmodell für jedes Muster Modelle hinterlegt werden, wie sich die Ausprägungen der Qualitätsmerkmale, die durch das Muster beeinflusst werden, in Abhängigkeit der Belegungen der Variationsmöglichkeiten und der Schätzungen über Ausgangswerte (z.B. Dauer einer Berechnung) verändern.

Informationen für die weitere Nutzung im Entwicklungsprozess

Wie in Abschnitt 4.5.2 bereits erwähnt, gibt es Musterbeschreibungen, in welchen Informationen über geeignete Formen des Entwurfs und der Implementierung des Musters enthalten sind. In Abschnitt 2.3.1 wurde erläutert, dass im Rahmen der vorliegenden Arbeit keine Muster betrachtet würden, die auf Paradigmen (z.B. Objektorientierung) ausgerichtet sind. Im Rahmen einer Erweiterung des Wissensmodells mit Informationen, die für den weiteren Entwurf von Systemen genutzt werden können nachdem eine grobe Architektur des Systems bereits fest steht, ist eine Betrachtung paradigmenabhängiger Muster nützlich. Ein Beispiel dafür sind die Muster aus [AMC03], in welchen die Verwendung von allgemeinen Architekturmustern in Bezug auf die Technologie J2EE und weitere Muster zur besseren Umsetzung unter dieser Technologie demonstriert werden.

Informationen über Verfahren zur Prüfung der korrekten Implementierung von Architekturmustern (z.B. durch Tests) sind in gängigen Musterbeschreibungen nicht enthalten. Eine Erweiterung des Wissensmodells bezüglich dieser Informationen kann bei der Integration der Architekturevaluation in den weiteren Entwicklungsprozess von Nutzen sein. Denkbar ist die Integration von Tests, mit welchen geprüft werden kann, ob die Qualitätsmerkmale, die durch den Einsatz des Musters beeinflusst werden sollen, in gewünschter Weise beeinflusst werden.

Informationen über Technologien

Der Einsatz von Technologien in Architekturen kann Auswirkungen auf die Ausprägungen von Qualitätsmerkmalen des Systems haben. Es gibt Technologien, die Architekturmuster einsetzen (z.B. sind Java RMI oder Corba Instanzierungen des Broker-Musters [BMR+96]) und Technologien, die den Einsatz von Mu-

stern vorsehen (beispielsweise wird in Java AWT der Einsatz von MVC vorgesehen). Werden Technologien in das Wissensmodell von POSAAM aufgenommen und mit Mustern und Prinzipien in Beziehung gesetzt, kann der Einsatz von Technologien im Rahmen einer Architekturbewertung einfacher und nachvollziehbarer auf Qualitätsmerkmale zurückgeführt werden.

ANHANG A

Zusammenfassung der „Tactics“ nach Bass et al.

Die von *Bass et al.* als „Tactics“ bezeichneten Mechanismen zur Beeinflussung von Qualitätsmerkmalen durch Gestaltung der Architektur werden in diesem Anhang zusammenfassend wiedergegeben. Für detaillierte Erläuterungen der Arbeitsweisen der Taktiken wird auf [BCK03, S. 102 ff] verwiesen.

Inhalt

A.1. Verfügbarkeit

Zur Steigerung der Verfügbarkeit nennen *Bass et al.* Taktiken aus drei Kategorien: „Erkennung von Fehlern“ *(engl. Fault Detection)*, „Fehlerbehebung“ *(engl. Fault Recovery)* und „Fehlervermeidung“ *(engl. Fault Prevention)*. Wobei die Kategorie „Fehlerbehebung“ noch weiter in die Kategorien „Vorbereitung und Behebung“ *(engl. Preparation and Repair)* sowie „Wiedereinführung“ *(engl. Reintroduction)* unterteilt ist.

Unter der Kategorie „Erkennung von Fehlern“ führen *Bass et al.* die Taktiken „Ping/echo“, „Heartbeat“ und „Exceptions“ auf. Bei den ersten beiden Taktiken wird ein Fehler einer kritischen Komponente anhand einer Überwachung durch weitere Komponenten erkannt, die durch periodische Abfragen den Zustand der kritischen Komponenten erfragen. Bei diesen Taktiken gibt es mehrere Varianten. Bei der Taktik der „Exceptions“ reagiert die Komponente selbst auf einen (vorher definierten und somit bekannten und erwarteten) Fehlerzustand.

Zur Kategorie „Fehlerbehebung“ gehören mehrere Taktiken, die mit unterschiedlichen Formen von Redundanz arbeiten. Sie unterscheiden sich sowohl durch die Form der Redundanz (z.B. redundante Systeme oder redundante Daten) als auch durch die Form des Einsatzes der Redundanz (z.B. sofortige Übernahme des Systems zum aktuellen Zustand oder Übernahme des Systems zu einem bestimmten Checkpoint).

Taktiken der Kategorie „Fehlervermeidung“ werden verwendet, um Zustände, bei denen die Verfügbarkeit beeinträchtigt wird, nicht eintreten zu lassen. *Bass et al.* nennen hier Transaktionen, Prozessmonitore und die Möglichkeit Komponenten vom System abzukoppeln, um Aktivitäten durchzuführen, die ein Fehlverhalten der Komponente unwahrscheinlicher machen (z.B. einen Neustart der Komponente, um Speicherlecks zu vermeiden).

Eine Übersicht der Taktiken zur Verfügbarkeit wird in Abb. A.1 gegeben.

A.2. Änderbarkeit

Bass et al. kategorisieren die Taktiken zur Änderbarkeit nach den Zielen, die durch den Einsatz der Taktiken verfolgt werden. Danach ermitteln *Bass et al.* drei Kategorien: „Änderungen lokal begrenzen“ *(engl. Localize Modifications)*, „Dominosteineffekt vermeiden“ *(engl. Prevent Ripple Effects)* und „Bindungen verzögern“ *(engl. Defer Binding Time)*.

Unter die Kategorie „Änderungen lokal begrenzen“ fallen die Taktiken, die zum Ziel haben, das System in einer Form zu untergliedern, so dass vorhersehbare

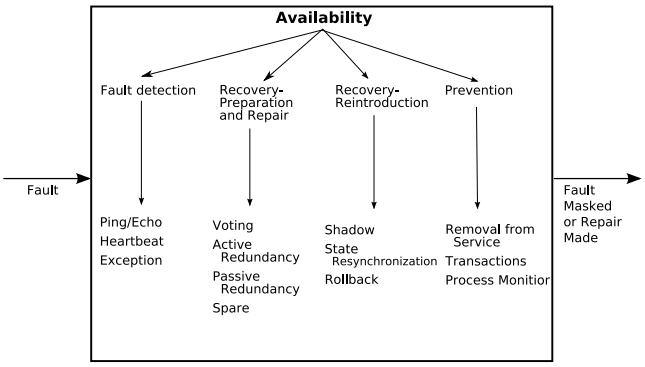

Abbildung A.1.: Mechanismen (Taktiken) zur Beeinflussung der Verfügbarkeit (aus [BCK03])

Änderungen einen Einfluss auf einen möglichst kleinen Teilbereich des Systems haben. *Bass et al.* sehen hierzu u.a. folgende Möglichkeiten: Die generische Gestaltung von Modulen, die Erhöhung der semantischen Kohärenz von Modulen und die Vorhersage von Änderungen. Durch die generische Gestaltung von Modulen können Änderungen evtl. durch veränderte Eingaben an das Modul umgesetzt und müssen nicht durch eine Veränderung der Implementierung des Moduls eingebracht werden. Durch die Erhöhung der semantischen Kohärenz von Modulen haben Änderungen aus der Problemdomäne, die im System nachgezogen werden müssen, lediglich lokale Auswirkungen. Und die Vorhersage von Änderungen ermöglicht die Kapselung von den Teilen, die einer Änderung unterzogen werden müssen, in separaten Modulen.

Die Taktiken der Kategorie „Dominosteineffekt vermeiden“ haben zum Ziel, dass Änderungen an einem Teil des Systems keine notwendigen Änderungen an weiteren Teilen des Systems nach sich ziehen. Dieser Fall tritt immer dann ein, wenn zwischen den Teilen des Systems (z.B. Modulen) Abhängigkeiten bestehen. Mit den Taktiken wird versucht, die Abhängigkeiten zwischen den Teilen des Systems zu reduzieren. Bass et al. zählen zu dieser Kategorie u.a. folgende Taktiken: Informationen kapseln, Schnittstellen bei Änderungen beibehalten und Nutzung eines Intermediärs. Die Kapselung von Informationen (bzw. von Daten) reduziert Abhängigkeiten, da keine Abhängigkeiten zu Daten bestehen kann, die von Außen nicht sichtbar sind. Die Schnittstellen eines Moduls bei einer Änderung beizubehalten, hat den Effekt, dass weitere Module, die von der (syntaktischen) Schnittstelle des veränderten Moduls abhängen, nicht verändert werden müssen. Eine Änderung der Schnittstelle zu vermeiden kann auf verschiedene Wege erreicht werden. Beispielsweise kann ein Adapter innerhalb des zu verändernden Moduls eingebaut werden. Durch den Einsatz eines Intermediärs zwischen zwei Modulen können die Änderungen, die durch eine Abhängigkeit nötig werden im

Intermediär abgefangen werden. Beispielsweise können Datenformate zwischen zwei Modulen durch einen Intermediär transformiert werden.

Taktiken der Kategorie „Bindungen verzögern“ haben zum Ziel, die Änderungen am System zu einem späteren Zeitpunkt als zum Entwicklungszeitpunkt zu ermöglichen. Dazu zählen nach *Bass et al.* beispielsweise Konfigurationsmechanismen, die Möglichkeit verschiedene Komponenten zum Programmstart einzubinden oder die Einhaltung von genormten Protokollen, was einen Wechsel von Interaktionen mit unterschiedlichen Einheiten zur Laufzeit ermöglicht.

Eine Übersicht der Taktiken zur Änderbarkeit wird in Abb. A.2 gegeben.

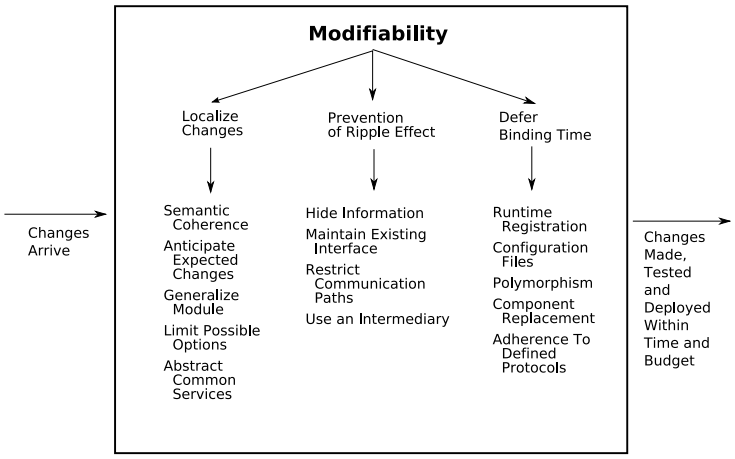

Abbildung A.2.: Mechanismen zur Beeinflussung der Änderbarkeit (aus [BCK03])

A.3. Performanz

Bass et al. sehen Taktiken zur Beeinflussung der Performanz eines Systems als Mittel, um die Zeit zu verringern, die ein System benötigt, um auf ein Ereignis zu reagieren. *Bass et al.* kategorisieren die Taktiken zur Beeinflussung der Performanz nach „Ressourcenbedarf“ *(engl. Resource Demand)*, „Management von Ressourcen“ *(engl. Resource Management)* und „Ressourcenzuweisung“ *(engl. Resource Arbitration)*.

Unter die Kategorie „Ressourcenbedarf“ fallen Taktiken, die verwendet werden, um den Bedarf an Ressourcen zu reduzieren. Implizit führen *Bass et al.* unter dieser Kategorie noch drei weitere Subkategorien. Der Ressourcenbedarf kann demnach reduziert werden, indem

1. der Ressourcenbedarf, den ein einzelnes, vom System zu verarbeitendes Ereignis benötigt, reduziert wird. *Bass et al.* nennen hier als Beispiel die Reduktion des Ressourcenbedarfs für eine Ressource durch Verlagerung auf andere Ressourcen (z.B. können Zwischenberechnungen im Speicher gehalten oder wenn benötigt erneut berechnet werden). Des Weiteren fallen unter diese Kategorie die Reduktion von Zusatzberechnungen wie z.B. Berechnungen zur Synchronisation mehrerer Prozesse oder Zugriffskontrollprüfungen. Die Reduktion dieser Berechnungen stellen Tradeoffs dar und müssen an die Anforderungen angepasst erfolgen.
2. die Anzahl der Ereignisse, die vom System zu verarbeiten sind reduziert werden. Bei Systemen, mit Sensoren, die Messungen über die Umwelt des Systems vornehmen, kann z.B. die Frequenz, mit der die Sensoren Ereignisse generieren, verringert werden. Auch können Ereignisse u.U. ignoriert werden. In beiden Fällen, führt die Taktik evtl. zu einem Verlust der Qualität der Berechnungen. Auch bei diesen Taktiken sind also die Anforderungen zu berücksichtigen.
3. der Zeit für die Ressourcennutzung eine Obergrenze gesetzt wird. Erneut ist bei dieser Taktik ein Abwägen zwischen dem Qualitätsmerkmal der Performanz und anderen Qualitätsmerkmalen zu erkennen.

Taktiken der Kategorie „Management von Ressourcen“ sind z.B. die Erhöhung der verfügbaren Ressourcen (leistungsfähigere / mehr Prozessoren / Speicher / usw.), die Verwaltung der verfügbaren Ressourcen, so dass die Ressourcen besser ausgelastet werden können (Stichwort „Load balancing“) oder auch Verfahren mit denen die Nutzung gleichartiger Ressourcen geregelt wird (Stichwort „Caching“).

In die Kategorie „Ressourcenzuweisung“ fallen Taktiken, bei denen konkurrierende Zugriffe auf Ressourcen geregelt werden (Stichwort „Scheduling“).

Eine Übersicht der Taktiken zur Performanz wird in Abb. A.3 gegeben.

A.4. Informationssicherheit

Taktiken zur Beeinflussung des Qualitätsmerkmals der Informationssicherheit werden von Bass et al. in die drei Kategorien „Angriffen standhalten“ *(engl. Resisting Attacks)*, „Angriffe identifizieren“ *(engl. Detecting Attacks)* und „Umgang mit erfolgten Angriffen“ *(engl. Recovering From Attacks)* eingeteilt.

Unter die erste Kategorie „Angriffen standhalten“ fallen die Taktiken, die die Beeinflussung der Qualitätsteilmerkmale des Qualitätsmerkmals „Informationssicherheit“ verfolgen. Darunter fallen z.B. die Identifikation, Authentifikation

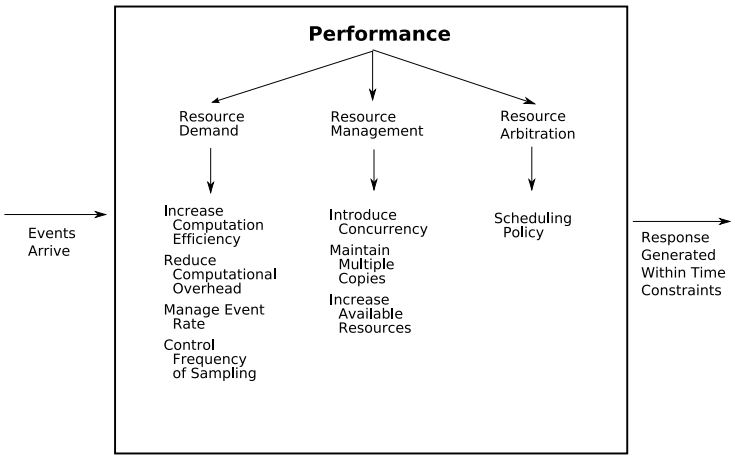

Abbildung A.3.: Mechanismen zur Beeinflussung der Effizienz bzw. Performanz (aus [BCK03])

und Autorisation von Benutzern, Taktiken zur Erhaltung von Vertraulichkeit und Integrität sowie Taktiken, um die Verbreitung des Zugriffs von kompromittierten Komponenten auf weitere Komponenten des Systems zu unterbinden.

Der Kategorie „Angriffe identifizieren“ gehören alle Arten von Intrusion Detection Systemen an.

Die Kategorie „Umgang mit erfolgten Angriffen“ wird von *Bass et al.* in die zwei Kategorien „Zustand wiederherstellen“ *(engl. Restoration)* und „Identifikation“ *(engl. Identification)* weiter untergliedert. Der ersten Kategorie gehören Taktiken an, mit denen der sichere Betrieb des Systems wiederhergestellt werden kann. Diese Taktiken sind den Taktiken der Verfügbarkeit ähnlich. Unterschiede bestehen in der besonderen Handhabung der Wiederherstellung von sicherheitsrelevanten Teilen des Systems, wie z.B. Kennwort- oder Zugriffskontrolldaten. Taktiken, mit denen die Aktionen der Benutzer eines Systems verfolgt und aufgezeichnet werden, gehören der Teilkategorie „Identifikation“ an. Zweck ist z.B. die Straf- oder Zivilrechtliche Verfolgung von Angreifern.

Eine Übersicht der Taktiken zur Informationssicherheit wird in Abb. A.4 gegeben.

A.5. Testbarkeit

Bass et al. unterteilen die Taktiken zur Testbarkeit in die beiden Kategorien „Ein-/Ausgabe“ *(engl. Input/Output)* und „Interne Überwachung“ *(engl. Internal Monitoring)*.

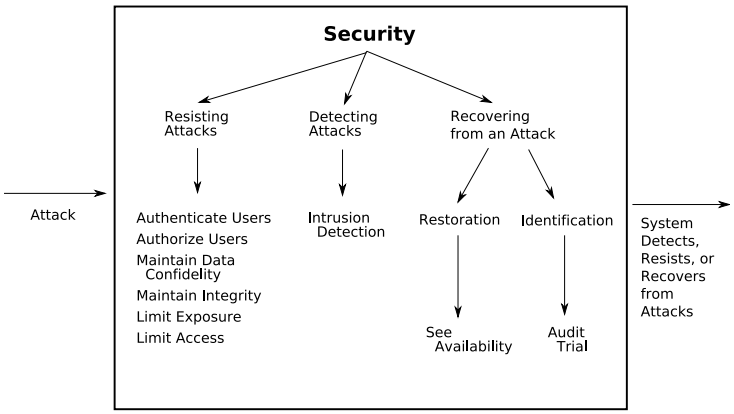

Abbildung A.4.: Mechanismen zur Beeinflussung der Informationssicherheit (aus [BCK03])

Bei den Taktiken zur Kategorie „Ein-/Ausgabe“ handelt es sich um Taktiken, mit denen die Ein- und Ausgabe für die zu testende Komponente kontrolliert werden kann. Beispielsweise können die Ein- und Ausgabe aus dem regulären Betrieb des Systems aufgezeichnet und für zukünftige Tests wiederverwendet werden. Weitere Taktiken dieser Kategorie sind die Verwendung von Komponenten, die besondere Eingaben an die umgebenden Komponenten liefern oder die Nutzung besonderer Zugriffsschnittstellen, mit denen die Zustände der betreffenden Komponenten explizit verändert werden können.

Zur Kategorie „Interne Überwachung“ werden Taktiken zugeordnet, bei denen die innerhalb der betreffende Komponente Informationen gesammelt werden, die über spezielle Zugriffsschnittstellen zur Verfügung gestellt werden (z.B. Information über Speicherbedarf oder Auslastung).

Eine Übersicht der Taktiken zur Testbarkeit wird in Abb. A.5 gegeben.

A.6. Benutzbarkeit

Bass et al. unterteilen die Taktiken zur Benutzbarkeit in die beiden Kategorien „Laufzeit“ *(engl. Runtime Tactics)* und „Entwurfszeit“ *(engl. Design-Time Tactics).*

Die Kategorie „Laufzeit“ wird von *Bass et al.* weiter in die Teilkategorien „Unterstützung der Nutzerinitiative“ *(engl. Support User Initiative)* und „Unterstützung der Systeminitiative“ *(engl. Support System Initiative)* unterteilt. Unter Taktiken zur Unterstützung der Nutzerinitiative verstehen *Bass et al.* das

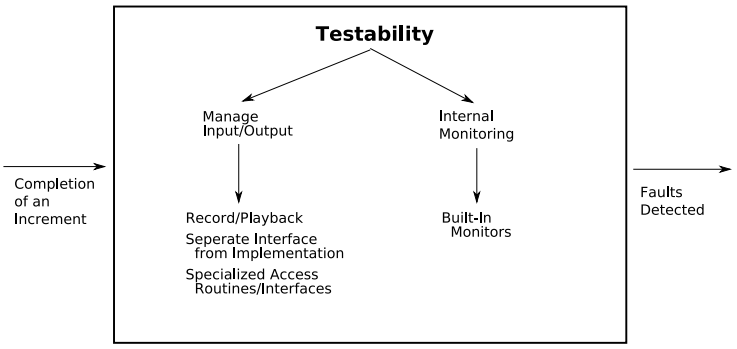

Abbildung A.5.: Mechanismen zur Beeinflussung der Testbarkeit (aus [BCK03])

Hinzufügen von Funktionalität, die dem Nutzer die Möglichkeit geben bezüglich der Bedienung des Systems aus eigener Initiative aktiv zu werden. Solche Funktionen sind u.a. der Abbruch einer Operation oder das Zurücksetzen auf einen Stand vor einer Operation. Solche Operationen sind nur durch eine entsprechende Berücksichtigung in der Architektur des Systems zu erreichen. Bei den Taktiken zur Unterstützung der Systeminitiative werden Modelle der Umgebung oder des aktuelle Systemzustands geschaffen, die dem System ermöglichen, Funktionen zu initiieren, die dem Benutzer die Bedienung des Systems erleichtern könnten. Explizit nennen Bass et al. ein System-, ein Benutzer- und ein Aufgabenmodell.

Unter den Taktiken der Kategorie „Entwurfszeit“ nennen Bass et al. lediglich die Taktik die Benutzerschnittstelle vom Rest der Applikation zu trennen, da davon auszugehen ist, dass die Benutzerschnittstelle im Laufe des Systemlebenszyklus oft zu überarbeiten ist. Es fällt auf, dass es sich hierbei um eine Taktik des Qualitätsmerkmals Änderbarkeit handelt.

Eine Übersicht der Taktiken zur Benutzbarkeit wird in Abb. A.6 gegeben.

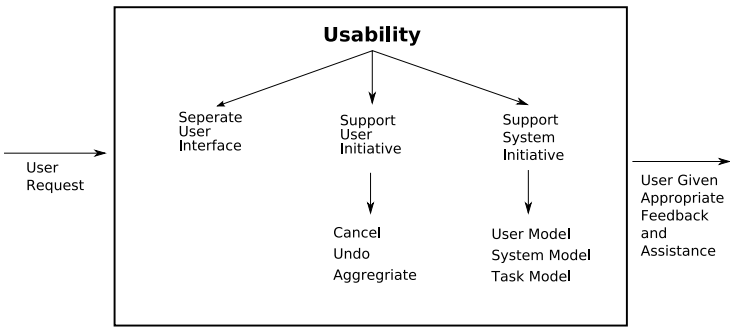

Abbildung A.6.: Mechanismen (Taktiken) zur Beeinflussung der Benutzbarkeit (aus [BCK03])

ANHANG B

Beispieleinträge für die Wissensbasis

Für ein besseres Verständnis werden in diesem Anhang einige beispielhafte Einträge nach den Vorgaben des Wissensmodells vorgenommen.

Inhalt

B.1. Beispieleinträge für Architekturmuster

B.1.1. Caching

In Tabelle B.1 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Architekturmuster „Caching“ [KJ04] vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben. Der beispielhafte Eintrag in die Wissensbasis gleicht nicht ganz der Beschreibung aus [KJ04]. Die Beziehung zwischen den Elementen *Resource User* und *Resource Provider* wird im Eintrag in die Wissensbasis nicht in der Topologie der wiederkehrenden Anteile aufgenommen, da es eine Möglichkeit der Variation des Musters gibt, bei welcher diese Beziehung nicht auftritt. Da aber die wiederkehrenden Anteile eines Musters in jeder Variationsform des Musters vorkommen müssen, sind die wiederkehrenden Anteile eines Musters auf die Anteile beschränkt, die allen Mustervarianten gleich sind.

Tabelle B.1.: Eintrag in die Wissensbasis für das Architekturmuster Caching

Quelle	P. Jain und M. Kircher: Pattern-Oriented Software Architecture: Patterns for Resource Management, Volume 3, John Wiley & Sons, 2004.	
Wiederkehrende Anteile		
	Elemente	Es existieren vier Elemente unter den wiederkehrenden Anteilen. In Anlehnung an die Beschreibung aus der oben genannten Quelle werden die englischen Bezeichnungen verwendet: *Resource User*, *Resource*, *Resource Cache* und *Resource Provider*.
	Topologie	Die Topologie der Elemente ist ein vollständiger Graph vermindert um die Kante, die das Element *Resource User* mit dem Element *Resource Provider* verbindet.
	Interaktionsform	Die Interaktion zwischen den Elementen beginnt mit dem Abrufen einer *Resource* beim *Resource Cache* durch den *Resource User*. Ist die *Resource* nicht bereits im *Resource Cache* abgelegt, wird die *Resource* zunächst vom *Resource Provider* abgerufen. Anschließend wird die *Resource* dem *Resource User* zurückgeliefert. Die *Resource* wird dann vom *Resource User* verwendet. Nach der Verwendung der *Resource* wird sie wieder dem *Resource Cache* zugeführt. Alle weiteren Zugriffe auf die *Resource* durch den *Resource User* finden über den *Resource Cache* statt. Detailliertere Beschreibungen zur Interaktionsform sind in der genannten Quelle hinterlegt.
	Einschränkungen	-

Variationsmöglichkeiten		
	Strategie zur Haltung von Daten im Cache	Das Zwischenspeichern von Ressourcen belastet andere Ressourcen. Typischerweise wird durch einen Cache die Speichereffizienz vermindert. Bei der Belegung dieser Variationsmöglichkeit wird ein Ausgleich zwischen dem Effizienzgewinn der zwischengespeicherten Ressource und der dazu belasteten Ressource (üblicherweise Speicher) geschaffen, indem im Cache zwischengespeicherte Ressourcen wieder freigegeben werden. Typische Strategien zur Haltung bzw. Entfernung von Ressourcen aus dem Cache sind „Least Recently Used“ oder „Least Frequently Used“. Es können beliebige Strategien verwendet werden. Auch die Strategie, dass Ressourcen nie aus dem Cache gelöscht werden, ist eine gültige Belegung der vorliegenden Variationsmöglichkeit. Bei dieser Variationsmöglichkeit wird die Verantwortlichkeit des Elements *Resource Cache* erweitert. Bei Einsatz des „Evictor“ Architekturmusters [KJ04] werden zusätzliche Elemente eingefügt.
	Form der Synchronisation zwischen Original und Cache	Es existieren Ressourcen, die ihren Zustand verändern können. Werden Ressourcen dieser Art in einem Cache zwischengespeichert, können Inkonsistenzen zwischen den zwischengespeicherten und den originalen Ressourcen auftreten. Bei Ressourcen dieser Art ist also eine Form der Synchronisation zwischen den Originalressourcen und den zwischengespeicherten Ressourcen durchzuführen. Die Form der Synchronisation kann von Fall zu Fall variieren. Die Variationen unterscheiden sich hinsichtlich des Zeitpunkts, wann eine Änderung an einer Ressource im Zwischenspeicher an der entsprechenden Originalressource durchgeführt wird und wann eine Änderung an einer Originalressource an entsprechenden zwischengespeicherten Ressourcen durchgeführt werden. Übliche Strategien sind das sofortige „durchschreiben“ von Änderungen an einem Zwischenspeicher in die entsprechende Originalressource sowie das Registrieren von Caches an Originalressourcen, so dass eine Änderung einer Originalressource sofort an registrierte Caches gemeldet und von diesen nachvollzogen wird. Andere Möglichkeiten sind periodische Updates, Update bei nächstem Zugriff oder gar kein Update. Bei dieser Variationsmöglichkeit werden sowohl die Verantwortlichkeiten der Elemente *Resource Cache* und *Resource Provider* als auch die Interaktionsform zwischen diesen Elementen verändert.
Qualitätsmerkmale		
	Positiv	Performanz, Verfügbarkeit
	Negativ	Speichereffizienz
	Die Qualitätsmerkmale Performanz und Speichereffizienz werden maßgeblich durch die Variationsmöglichkeit „Strategie zur Haltung von Daten im Cache“ beeinflusst.	

Zugrunde liegende Prinzipien		
	Performanz und Speichereffizienz	Die Auswirkungen auf die Qualitätsmerkmale Performanz und Speichereffizienz lassen sich durch das Prinzip „Verlagerung von Ressourcenbedarf“ begründen. Die Originalressource, die durch das Muster in Kopie in einem Cache zwischengespeichert wird, ist eine knappe Ressource. Durch die Zwischenspeicherung wird eine alternative Ressource belastet. Die Verwendung der knappen Ressource wird reduziert.
	Verfügbarkeit	Die Auswirkungen auf das Qualitätsmerkmal Verfügbarkeit lässt sich durch das Prinzip „Nutzung redundanter Ressourcen“ begründen. Werden durch den Cache wichtige Ressourcen zwischengespeichert, liegen diese auch zur Nutzung bereit, wenn der Zugriff auf die Originalressource nicht möglich ist.
Qualitätseinfluss von Variationsmöglichkeiten		
	Strategie zur Haltung von Daten im Cache	**Tradeoff Point:** Performanz, Speichereffizienz. Je mehr Ressourcen im Cache zwischengespeichert werden, desto höher ist die Performanz und desto niedriger ist die Speichereffizienz. **Zugrunde liegende Prinzipien** Der Tradeoff Point zwischen den Qualitätsmerkmalen Performanz und Speichereffizienz lässt sich durch das Prinzip „Verlagerung von Ressourcenbedarf“ begründen. Die Originalressource, die durch das Muster in Kopie in einem Cache zwischengespeichert wird, ist eine knappe Ressource. Durch die Zwischenspeicherung wird eine alternative Ressource belastet. Die Verwendung der knappen Ressource wird reduziert.
	Form der Synchronisation zwischen Original und Cache	**Sensitivity Point:** Performanz. Je weniger Updates durchgeführt werden, desto performanter verhält sich die Anwendung. **Zugrunde liegende Prinzipien** Der Sensitivity Point lässt sich durch das Prinzip „Entlastung einer Ressource“ begründen. Die knappe Ressource ist CPU-Rechenzeit. Werden weniger Updates durchgeführt, müssen weniger Berechnungen (Belastung der CPU-Rechnenzeit) durchgeführt werden.
Beziehungen		
	verfeinert	-
	besteht aus	-
	verwendet	Evictor [KJ04]

B.1.2. Pooling

In Tabelle B.2 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Architekturmuster „Pooling“ [KJ04] vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

Tabelle B.2.: Eintrag in die Wissensbasis für das Architekturmuster Pooling

Quelle	P. Jain und M. Kircher: Pattern-Oriented Software Architecture: Patterns for Resource Management, Volume 3, John Wiley & Sons, 2004.	
Wiederkehrende Anteile		
	Elemente	Es existieren vier Elemente unter den wiederkehrenden Anteilen. In Anlehnung an die Beschreibung aus der oben genannten Quelle werden die englischen Bezeichnungen verwendet: *Resource User*, *Resource*, *Resource Pool* und *Resource Provider*.
	Topologie	Die Topologie der Elemente ist ein vollständiger Graph vermindert um die Kante, die das Element *Resource User* mit dem Element *Resource Provider* verbindet.
	Interaktionsform	Die Interaktion zwischen den Elementen beginnt damit, dass der *Resource Pool* eine definierte Menge von *Resourcen* beim *Resource Provider* erwirbt. Diese *Resources* stehen anschließend im *Resource Pool* zur Verfügung. Bei Bedarf wird eine *Resource* durch den *Resource User* beim *Resource Pool* abgerufen. Der *Resource User* muss die *Resource* dann ggf. geeignet initialisieren bzw. belegen und kann die *Resource* anschließend nutzen. Sind alle *Resources* im *Resource Pool* in Gebrauch, werden mehr *Resources* durch den *Resource Pool* beim *Resource Provider* abgerufen. Nach Nutzung einer *Resource* wird sie in den *Resource Pool* zurückgeführt. Dazu muss die Ressource wieder von Belegungen befreit werden. Dies kann durch den *Resource User* oder den *Resource Pool* durchgeführt werden. Detailliertere Beschreibungen zur Interaktionsform sind in der genannten Quelle hinterlegt.
	Einschränkungen	-
Variationsmöglichkeiten		
	Anzahl der Ressourcen im Pool	Die Anzahl der *Resources*, die im *Resource Pool* gehalten werden, muss so angepasst werden, dass für den Großteil aller Anfragen durch *Resource User Resources* im *Resource Pool* bereitstehen. Gleichzeitig sollte die Anzahl der *Resources*, die im *Resource Pool* gehalten werden und nicht benötigt werden möglichst gering sein. Die Anzahl der *Resources*, die im *Ressource Pool* gehalten werden, kann statisch festgelegt sein oder dynamisch während der Laufzeit geändert werden. Bei dieser Variationsmöglichkeit wird die Verantwortlichkeit des Elements *Resource Pool* verändert.

Qualitätsmerkmale		
	Positiv	Performanz
	Negativ	Speichereffizienz
Zugrunde liegende Prinzipien		
	Performanz und Speichereffizienz	Die Auswirkungen auf die Qualitätsmerkmale Performanz und Speichereffizienz lassen sich durch das Prinzip „Verlagerung von Ressourcenbedarf“ begründen. Die Ressource, die durch das Muster vorzeitig in einem Pool bereitgestellt wird, ist eine knappe Ressource. Durch die Bereitstellung wird eine alternative Ressource belastet. Die Verwendung der knappen Ressource wird reduziert und geschieht zu einem anderen Zeitpunkt.
Qualitätseinfluss von Variationsmöglichkeiten		
	Anzahl der Ressourcen im Pool	**Tradeoff Point:** Performanz, Speichereffizienz. Je mehr Ressourcen im Pool gelagert werden, desto höher ist die Performanz und desto niedriger ist die Speichereffizienz. **Zugrunde liegende Prinzipien** Der Tradeoff Point zwischen den Qualitätsmerkmalen Performanz und Speichereffizienz lässt sich durch das Prinzip „Verlagerung von Ressourcenbedarf“ begründen. Die Ressource, die durch das Muster vorzeitig in einem Pool bereitgestellt wird, ist eine knappe Ressource. Durch die Bereitstellung wird eine alternative Ressource belastet. Die Verwendung der knappen Ressource wird reduziert und geschieht zu einem anderen Zeitpunkt.
Beziehungen		
	verfeinert	-
	besteht aus	-
	verwendet	Leasing [KJ04]

B.1.3. Eager Acquisition

In Tabelle B.3 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Architekturmuster „Eager Acquisition“ [KJ04] vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

Tabelle B.3.: Eintrag in die Wissensbasis für das Architekturmuster Eager Acquisition

Quelle	P. Jain und M. Kircher: Pattern-Oriented Software Architecture: Patterns for Resource Management, Volume 3, John Wiley & Sons, 2004.

Wiederkehrende Anteile		
	Elemente	Es existieren vier Elemente unter den wiederkehrenden Anteilen. In Anlehnung an die Beschreibung aus der oben genannten Quelle werden die englischen Bezeichnungen verwendet: *Resource User*, *Resource*, *Provider Proxy* und *Resource Provider*.
	Topologie	Die Topologie der Elemente ist ein vollständiger Graph vermindert um die Kante, die das Element *Resource User* mit dem Element *Resource Provider* verbindet.
	Interaktionsform	Die Interaktion zwischen den Elementen beginnt damit, dass der *Provider Proxy* nach einer festzulegenden Strategie (vgl. Variationsmöglichkeiten) *Resourcen* beim *Resource Provider* erwirbt. Bei Bedarf wird eine *Resource* durch den *Resource User* beim *Provider Proxy* abgerufen. Ist die vom *Resource User* benötigte Ressource nicht im Voraus durch den *Provider Proxy* erworben worden, muss die *Resource* durch den *Provider Proxy* beim *Resource Provider* zum Zeitpunkt der Anfrage durch den *Resource User* abgerufen werden. Detailliertere Beschreibungen zur Interaktionsform sind in der genannten Quelle hinterlegt.
	Einschränkungen	-
Variationsmöglichkeiten		
	Strategie des Ressourcenerwerbs	Um durch den Einsatz dieses Musters einen positiven Effekt auf die Performanz zu erzielen, muss der Ressourcenerwerb durch den *Provider Proxy* so geschehen, dass 1. die *Resource* zum Zeitpunkt der Anfrage zum Erwerb der *Resource* durch den *Resource User* bereits beim *Provider Proxy* vorliegt. 2. die Ressource für einen möglichst geringen Zeitraum (Zeitpunkt des vorzeitigen Erwerbs der *Resource* durch den *Provider Proxy* bis zum Zeitpunkt der Anfrage zum Erwerb der *Resource* durch den *Resource User*) zwischengespeichert werden muss. 3. der Ressourcenerwerb zu einem Zeitpunkt geschieht, zu dem der Verlust an Performanz, der durch den Erwerb der Ressource entsteht, hinnehmbar ist. Zu diesem Zweck ist eine Strategie für den vorzeitigen Erwerb von Ressourcen durch den *Provider Proxy* zu wählen, durch die zu geeigneten Zeitpunkten (z.B. bei geringer Rechenauslastung des Systems) solche Ressourcen erworben werden, für welche absehbar ist, dass diese innerhalb eines absehbaren Zeitraums durch einen *Ressource User* angefragt werden.

Qualitätsmerkmale		
	Positiv	Performanz
	Negativ	Speichereffizienz
Zugrunde liegende Prinzipien		
	Performanz und Speichereffizienz	Die Auswirkungen auf die Qualitätsmerkmale Performanz und Speichereffizienz lassen sich durch das Prinzip „Verlagerung der zeitlichen Auslastung von Ressourcenbedarf“ begründen. Die Verwendung einer knappen Ressource geschieht zu einem anderen Zeitpunkt, zu welchem angenommen wird, dass zu diesem Zeitpunkt die Auslastung der Ressource geringer ist.
Qualitätseinfluss von Variationsmöglichkeiten		
	Strategie des Ressourcenerwerbs	**Tradeoff Point:** Performanz, Speichereffizienz. Werden Ressourcen früh erworben sind sie schnell verfügbar (Steigerung der Performanz). Werden die Ressourcen jedoch lange Zeit nicht benötigt und müssen zwischengespeichert werden, belastet dies die Speichereffizienz. **Zugrunde liegende Prinzipien** Der Tradeoff Point zwischen den Qualitätsmerkmalen Performanz und Speichereffizienz lässt sich durch das Prinzip „Verlagerung von Ressourcenbedarf“ begründen. Die Ressource, die durch das Muster frühzeitig erworben wird, ist eine knappe Ressource. Muss diese Ressource zwischengespeichert werden, so lange sie nicht zur Nutzung abgerufen wird, belastet das Zwischenspeichern der Ressource die Speichereffizienz.
Beziehungen		
	verfeinert	-
	besteht aus	-
	verwendet	Reflection (vgl. [KJ04])

B.2. Beispieleinträge für Prinzipien

B.2.1. Kapselung von sich wahrscheinlich ändernden Teilen

In Tabelle B.4 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Prinzip „Kapselung von sich wahrscheinlich ändernden Teilen“ vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

B.2.2. Entlastung einer Ressource

In Tabelle B.5 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Prinzip „Entlastung einer Ressource“ (vgl. Abschnitt A.3 oder [BCK03, Kapitel 5]) vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

B.2.3. Verlagerung von Ressourcenbedarf

In Tabelle B.6 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Prinzip „Verlagerung von Ressourcenbedarf“ (vgl. Abschnitt A.3 oder [BCK03, Kapitel 5]) vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

B.2.4. Verlagerung der zeitlichen Auslastung von Ressourcen

In Tabelle B.7 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Prinzip „Verlagerung der zeitlichen Auslastung von Ressourcen“ vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

B.2.5. Nutzung redundanter Ressourcen

In Tabelle B.8 wird ein beispielhafter Eintrag in eine Wissensbasis nach dem POSAAM-Wissensmodell für das Prinzip „Nutzung redundanter Ressourcen“ vorgenommen. Wie der Eintrag vorzunehmen ist, wird in Abschnitt 4.2.2 ab Seite 81 vorgeschrieben.

Tabelle B.4.: Eintrag in die Wissensbasis für das Prinzip „Kapselung von sich wahrscheinlich ändernden Teilen“

Prinzip	Gestaltung der Architektur, so dass Teile des durch die Architektur beschriebenen System, die sich wahrscheinlich Ändern, vom Rest des Systems derart entkoppelt sind, dass erwartete Änderungen an diesen Teilen des Systems keine Änderungen am restlichen System nach sich ziehen.	
Eigenschaften, die durch Prinzip geregelt werden		
	Wahrscheinlichkeit der Änderung von Teilen	Der Einsatz des Prinzips beruht auf dem Vorliegen von Informationen darüber, wie Wahrscheinlich sich welche Teile des Systems verändern. Die Wahrscheinlichkeit der Änderung eines Teils eines Systems muss nicht exakt vorliegen, aber die Änderungswahrscheinlichkeiten müssen in einer Ordnung zueinander liegen, so dass es möglich ist, die Teile eines Systems zu identifizieren, die sich wahrscheinlicher Ändern als andere Teile eines Systems.
	Zu erwartende Änderungen von Teilen	Zusätzlich zur Wahrscheinlichkeit von Änderungen im Allgemeinen beruht der Einsatz des Prinzips auch auf der Art der zu erwartenden Änderungen.
	Kopplung zwischen Teilen	Die Eigenschaft, die durch den Einsatz des Prinzips geregelt wird, ist die Kopplung zwischen den Teilen des Systems. Die Verschiedenen Formen der Kopplung wurden von *Stevens et al.* in [SMC74] erörtert.
Qualitätseinfluss von Prinzip		
	Wartbarkeit	Die Wartbarkeit wird durch Anwendung dieses Prinzips positiv beeinflusst, wie in [Par72] erläutert.
Beziehung zu anderen Prinzipien		
	befolgt regeln	-
	setzt voraus	-

Tabelle B.5.: Eintrag in die Wissensbasis für das Prinzip „Entlastung einer Ressource“

Prinzip	Gestaltung der Architektur, so dass eine Ressource, die einen Engpass der Effizienz darstellt, möglichst selten verwendet werden muss. Vergleiche [BCK03, Kapitel 5].	
Eigenschaften, die durch Prinzip geregelt werden		
	Vorliegende knappe Ressource	Der Einsatz des Prinzips ist abhängig vom Vorliegen knapper Ressourcen. Hierbei handelt es sich üblicherweise um Rechenleistung (Prozessor), Speicher oder Ein-/Ausgabegeräte. Denkbar sind jedoch auch alle anderen Arten von Ressourcen (z.B. Mensch oder Energie).
	Verwendung der Ressource	Durch den Einsatz des Prinzips wird die Verwendung der knappen Ressource verändert, indem die Häufigkeit der Verwendung reduziert wird.
Qualitätseinfluss von Prinzip		
	Effizienz	Da die Ressource weniger verwendet wird, reduziert sich die „Knappheit“ der Ressource.
Beziehung zu anderen Prinzipien		
	befolgt regeln	-
	setzt voraus	-

Tabelle B.6.: Eintrag in die Wissensbasis für das Prinzip „Verlagerung von Ressourcenbedarf“

Prinzip	Gestaltung der Architektur, so dass eine Ressource, die einen Engpass der Effizienz darstellt, möglichst selten verwendet werden muss, indem eine andere Ressource, die keinen oder einen geringeren Engpass darstellt, belastet wird. Beispielsweise, indem Berechnungsergebnisse in einem Speicher gehalten werden. Vergleiche [BCK03, Kapitel 5].	
Eigenschaften, die durch Prinzip geregelt werden		
	Vorliegende knappe Ressource	Der Einsatz des Prinzips ist abhängig vom Vorliegen und der Nutzung (sowohl Art als auch Häufigkeit) knapper Ressourcen. Hierbei handelt es sich üblicherweise um Rechenleistung (Prozessor), Speicher oder Ein-/Ausgabegeräte. Denkbar sind jedoch auch alle anderen Arten von Ressourcen (z.B. Mensch oder Energie).
	Verwendung der Ressource	Durch den Einsatz des Prinzips wird die Verwendung der knappen Ressource verändert, indem die Häufigkeit der Verwendung reduziert wird. Die Verwendung einer oder mehrerer anderer Ressourcen wird verändert (in Häufigkeit und evtl. auch Art der Verwendung).
	Vorliegende alternative Ressource(n)	Durch den Einsatz des Prinzips werden alternative Ressourcen in ihrer Verwendung verändert, u.U. werden alternative Ressourcen durch den Einsatz des Prinzips erzeugt.
Qualitätseinfluss von Prinzip		
	Effizienz	Da die knappe Ressource weniger verwendet wird, reduziert sich die „Knappheit“ der Ressource.
	Effizienz	Die „Knappheit“ der alternativen Ressource wird durch die Anwendung des Prinzips erhöht.
	Da bei Anwendung des vorliegenden Prinzips die Verlagerung der Verwendung von einer knappen Ressource auf die Verwendung einer nicht knappen oder weniger knappen Ressource erfolgt, kann die Effizienz insgesamt gesteigert werden. Zudem können verschiedene Formen der Effizienz gegeneinander abgewogen werden. So kann z.B. die Speichereffizienz vermindert werden, um die Performanz zu steigern.	
Beziehung zu anderen Prinzipien		
	befolgt regeln	Entlastung einer Ressource
	setzt voraus	-

Tabelle B.7.: Eintrag in die Wissensbasis für das Prinzip „Verlagerung der zeitlichen Auslastung von Ressourcen“

Prinzip	Gestaltung der Architektur, so dass Zugriffe auf Ressourcen, die einen Engpass der Effizienz darstellen, so geregelt werden, dass die Zugriffe zeitlich so verteilt vorgenommen werden, dass die Ressource einer gleichmäßigen Auslastung unterliegt.	
Eigenschaften, die durch Prinzip geregelt werden		
	Vorliegende knappe Ressource	Der Einsatz des Prinzips ist abhängig vom Vorliegen und der Nutzung (sowohl Dauer als auch Häufigkeit als auch gleichzeitige Zugriffe) knapper Ressourcen. Hierbei handelt es sich üblicherweise um Rechenleistung (Prozessor), Speicher oder Ein-/Ausgabegeräte. Denkbar sind jedoch auch alle anderen Arten von Ressourcen (z.B. Mensch oder Energie).
	Zeitliche Verwendung der Ressource	Durch den Einsatz des Prinzips wird die zeitliche Verwendung der knappen Ressource verändert, indem der Zeitpunkt der Verwendung der Ressource verlagert wird.
Qualitätseinfluss von Prinzip		
	Effizienz	Da die Verwendung der knappen Ressource so verändert wird, dass die Zugriffe auf die Ressource zu Zeitpunkten stattfinden, zu denen die Auslastung der Ressource gering ist, wird vermieden, dass die Auslastung der Ressource die Kapazitäten der möglichen Auslastung der Ressource übersteigt.
Beziehung zu anderen Prinzipien		
	befolgt regeln	Verlagerung von Ressourcenbedarf
	setzt voraus	-

Tabelle B.8.: Eintrag in die Wissensbasis für das Prinzip „Nutzung redundanter Ressourcen“

Prinzip	Gestaltung der Architektur, so dass Ressourcen, die für den Betrieb des Systems wichtig sind, und der Zugriff auf solche Ressourcen redundant verfügbar gehalten werden. Vergleiche [BCK03, Kapitel 5].	
Eigenschaften, die durch Prinzip geregelt werden		
	Vorliegende wichtige Ressource	Der Einsatz des Prinzips wird durch das Vorliegen und die Verwendung von für die Verfügbarkeit des Systems wichtiger Ressourcen geleitet. Wichtige Ressourcen für die Verfügbarkeit von Systemen können z.B. Rechenleistung (Prozessor), Speicher oder Ein-/Ausgabegeräte oder aber auch ganze Subsysteme wie z.B. Datenbanken oder Zertifizierungsinstanzen sein. Denkbar sind jedoch auch alle anderen Arten von Ressourcen (z.B. Mensch oder Energie).
	Verwendung der Ressource	Der Einsatz des Prinzips wird durch das Vorliegen und die Verwendung von für die Verfügbarkeit des Systems wichtiger Ressourcen geleitet. Die Verwendung wichtiger Ressourcen und auch die Verwendung redundant vorliegender Ressourcen wird durch Einsatz des Prinzips verändert.
	Redundanz von Ressource(n)	Durch den Einsatz des Prinzips werden ggf. neue, redundante Ressourcen erzeugt. Die Verwendung wichtiger als auch redundant vorliegender Ressourcen wird durch den Einsatz des Prinzips verändert.
Qualitätseinfluss von Prinzip		
	Verfügbarkeit	Wenn die betroffene Ressource oder der Zugriff auf die betroffene Ressource nicht mehr verfügbar ist, kann dennoch ein Zugriff auf die Redundant hinterlegte Ressource möglich sein.
	Effizienz	Ressourcen müssen redundant verfügbar gemacht werden. Dies geht zu Lasten der Effizienz.
Beziehung zu anderen Prinzipien		
	befolgt regeln	-
	setzt voraus	-

Literaturverzeichnis

[ABC+97] Abowd, Gregory, Len Bass, Paul Clements, Rick Kazman, Linda Northrop und Amy Zaremski: *Recommended Best Industrial Practice for Software Architecture Evaluation.* Technischer Bericht CMU/SEI-96-TR-025, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Januar 1997.

[ADMC85] Alexander, Christopher, Howard Davis, Julio Martinez und Donald Corner: *The Production of Houses*, Band 4 der Reihe *Center for Environmental Structure Series.* Oxford University Press, New York, 1985.

[AIS+77] Alexander, Christopher, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King und Shlomo Angel: *A Pattern Language: Towns – Buildings – Construction*, Band 2 der Reihe *Center for Environmental Structure Series.* Oxford University Press, New York, 1977.

[AKK01] Asundi, J., R. Kazman und M. Klein: *Using Economic Considerations to Choose Among Architecture Design Alternatives.* Technischer Bericht CMU/SEI-2001-TR-035, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Dezember 2001.

[Ale79] Alexander, Christopher: *The Timeless Way of Building*, Band 1 der Reihe *Center for Environmental Structure Series.* Oxford University Press, New York, 1979.

[Ale81] Alexander, Christopher: *The Linz Café/Das Linz Café*, Band 5 der Reihe *Center for Environmental Structure Series*. Oxford University Press, New York, 1981.

[AMC03] Alur, Deepak, Dan Malks und John Crupi: *Core J2EE Patterns: Best Practice and Design Strategies*. Prentice Hall, zweite Auflage, Mai 2003.

[ASA+75] Alexander, Christopher, M. Silverstein, S. Angel, S. Ishikawa und D. Abrams: *The Oregon Experiment*, Band 3 der Reihe *Center for Environmental Structure Series*. Oxford University Press, New York, NY, 1975.

[Bab04] Babar, Muhammad Ali: *Scenarios, quality attributes, and patterns: capturing and using their synergistic relationships for product line architectures*. In: *11th Asia-Pacific Software Engineering Conference (APSEC'04).*, Seiten 574–578, Dezember 2004.

[Bal98] Balzert, H.: *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum Akademischer Verlag, 1998.

[BB98] Bengtsson, PerOlof und Jan Bosch: *Scenario-based Software Architecture Reengineering*. In: *International Conference on Software Reuse. Proceedings.*, Seiten 308–317, Juni 1998.

[BBK+78] Boehm, Barry W., John R. Brown, Hans Kaspar, Myron Lipow, Gordon J. MacLeod und Michael J. Merritt: *Characteristics of software quality*. TRW series of software technology. North-Holland Publishing Company, 1978.

[BC87] Beck, Kent und Ward Cunningham: *Using Pattern Languages for Object-Oriented Programs*. Technischer Bericht CR-87-43, Computer Research Laboratory, Tektronix Inc., September 1987.

[BC89] Beck, K. und W. Cunningham: *A laboratory for teaching object oriented thinking*. In: *OOPSLA '89: Conference proceedings on Object-oriented programming systems, languages and applications*, Seiten 1–6, New York, NY, USA, 1989. ACM.

[BCK03] Bass, L., P. Clements und R. Kazman: *Software Architecture in Practice*. Addison-Wesley, zweite Auflage, 2003.

[Ben08] Beneken, Gerd Hinrich: *Logische Architekturen: Eine Theorie der Strukturen und ihre Anwendung in Dokumentation und Projektmanagement*. Doktorarbeit, Institut für Informatik der Technischen Universität München, Boltzmannstr. 3, 85748 Garching, Juli 2008.

[BFJK99] Bergey, John K., Matthew J. Fisher, Lawrence G. Jones und Rick Kazman: *Software Architecture Evaluation with ATAM in the DoD System Acquisition Context.* Technischer Bericht CMU/SEI-99-TN-012, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, September 1999.

[BHS07a] Buschmann, Frank, Kevlin Henney und Douglas C. Schmidt: *Past, Present, and Future Trends in Software Patterns.* IEEE Software, 24(4):31–37, Juli - August 2007.

[BHS07b] Buschmann, Frank, Kevlin Henney und Douglas C. Schmidt: *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*, Band 4 der Reihe *Wiley Series in Software Design Patterns.* John Wiley & Sons, Inc. New York, NY, USA, 2007.

[BHS07c] Buschmann, Frank, Kevlin Henney und Douglas C. Schmidt: *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages*, Band 5 der Reihe *Wiley Series in Software Design Patterns.* John Wiley & Sons, Ltd, 2007.

[BLBvV00] Bengtsson, PerOlof, Nico Lassing, Jan Bosch und Hans van Vliet: *Analyzing software architectures for modifiability.* Technischer Bericht HK/R-RES—00/11—SE, Department of Software Engineering and Computer Science, University of Karlskrona/Ronneby, Sweden, S-372 25 Ronneby, Sweden, 2000.

[BLBvV04] Bengtsson, P., N. Lassing, J. Bosch und H. van Vliet: *Architecture-level modifiability analysis (ALMA).* Journal of Systems and Software, 69(1-2):129–147, Januar 2004.

[BMR+96] Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad und M. Stal: *Pattern-oriented software architecture: a system of patterns*, Band 1 der Reihe *Wiley Series in Software Design Patterns.* John Wiley & Sons, Inc. New York, NY, USA, 1996.

[Boe81] Boehm, Barry W.: *Software Engineering Economics.* Prentice Hall PTR Upper Saddle River, NJ, USA, 1981.

[Boo07] Booch, Grady: *The Well-Tempered Architecture.* IEEE Software, 24(4):24–25, 2007.

[BR08] Böhme, Rainer und Ralf Reussner: *Validation of Predictions with Measurements*, Band 4909/2008 der Reihe *Lecture Notes in Computer Science*, Kapitel 3, Seiten 14–18. Springer, Mai 2008.

[BW99] Barbacci, Mario R. und William G. Wood: *Architecture Tradeoff Analyses of C4ISR Products*. Technischer Bericht CMU/SEI-99-TR-014, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Juni 1999.

[BZJ04] Babar, Muhammad Ali, Liming Zhu und Ross Jeffery: *A framework for classifying and comparing software architecture evaluation methods*. In: Zhu, L. (Herausgeber): *Australian Software Engineering Conference, 2004. (ASWEC'04)*, Seiten 309–318, 2004.

[CBB+02] Clements, P., F. Bachmann, L. Bass, D. Garlan, J. Ivers und R. Little: *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2002.

[CKK02] Clements, P., R. Kazman und M. Klein: *Evaluating Software Architectures: Methods and Case Studies*. Addison-Wesley, 2002.

[Cle00] Clements, Paul C.: *Active Reviews for Intermediate Designs*. Technischer Bericht CMU/SEI-2000-TN-009, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, August 2000.

[CS95] Coplien, James O. und Douglas C. Schmidt (Herausgeber): *Pattern languages of program design*, Band 1 der Reihe *Software Patterns Series*. ACM Press/Addison-Wesley Publishing Co., 1995.

[dCP08] Cruz, David Bettencourt da und Birgit Penzenstadler: *Designing, Documenting, and Evaluating Software Architecture*. Technischer Bericht TUM-INFO-06-I0818-0/1.-FI, Technische Universität München, Institut für Informatik, Boltzmannstr. 3, 85748 Garching, GERMANY, Juni 2008.

[DIN95] DIN e.V.: *8402: Qualitätsmanagement Begriffe*, 1995.

[DN02] Dobrica, L. und E. Niemelä: *A survey on software architecture analysis methods*. Transactions on Software Engineering, 28(7):638–653, 2002.

[EA08] Erfanian, Aida und Fereidoun Shams Aliee: *An ontology-driven software architecture evaluation method*. In: *SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge*, Seiten 79–86, New York, NY, USA, 2008. ACM.

[Eck01] Eckert, Claudia: *IT-Sicherheit: Konzepte – Verfahren – Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, 2001.

[EHM07] Eicker, Stefan, Christian Hegmanns und Stefan Malich: *Auswahl von Bewertungsmethoden für Softwarearchitekturen.* In: *ICB-Research Report*, Nummer 14 in *ICB-Research Reports*. Universität Duisburg Essen, März 2007.

[Gar84] Garvin, David A.: *What does product quality really mean.* MIT Sloan Management Review, 26(1):25–43, 1984.

[GHJV95] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Design patterns, Elements of Reusable Object-Oriented Software.* Addison-Wesley Reading, Mass, 1995.

[Han07] Hanmer, Robert S.: *Patterns for Fault Tolerant Software.* Wiley Series in Software Design Patterns. John Wiley & Sons, Inc. New York, NY, USA, 2007.

[Has06] Hasselbring, W.: *Software-Architektur - Das aktuelle Schlagwort.* Informatik-Spektrum, 29(1):48–52, 2006.

[HNS99] Hofmeister, Christine, Robert L. Nord und Dilip Soni: *Applied Software Architecture.* Object Technology Series. Addison-Wesley, 1999.

[IEE00] The Institute of Electrical and Electronics Engineers, Inc.: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (ANSI/IEEE-Std-1471)*, IEEE-SA Standards Board Auflage, September 2000.

[Joi01] Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software engineering: *ISO/IEC 9126-1:2001 Software engineering – Product quality – Part 1: Quality model.* International Organization for Standardization and International Electrotechnical Commission, Juni 2001.

[Jon98] Jones, T.C.: *Estimating software costs.* McGraw-Hill, Inc. Hightstown, NJ, USA, 1998.

[JRvdL00] Jazayeri, Mehdi, Alexander Ran und Frank van der Linden: *Software Architecture for Product Families: Principles and Practice.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 2000.

[KABC96] Kazman, R., G. Abowd, L. Bass und P. Clements: *Scenario-based analysis of software architecture.* IEEE Software, 13(6):47–55, 1996.

[KAK01] Kazman, R., Jai Asundi und M. Klein: *Quantifying the costs and benefits of architectural decisions*. In: *Proceedings of the 23rd International Conference on Software Engineering, 2001. ICSE 2001.*, Seiten 297–306, 2001.

[KBK+99] Kazman, Rick, Mario Barbacci, Mark Klein, S. Jeromy Carrière und Steven G. Woods: *Experience with performing architecture tradeoff analysis*. In: *ICSE '99: Proceedings of the 21st international conference on Software engineering*, Seiten 54–63, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[KBWA94] Kazman, Rick, Len Bass, Mike Webb und Gregory Abowd: *SAAM: a method for analyzing the properties of software architectures*. In: *Proceedings of the 16th international conference on Software engineering*, Seiten 81–90. IEEE Computer Society Press Los Alamitos, CA, USA, 1994.

[KJ04] Kircher, Michael und Prashant Jain: *Pattern-Oriented Software Architecture: Patterns for Resource Management*, Band 3 der Reihe *Wiley Series in Software Design Patterns*. John Wiley & Sons, 2004.

[KK99] Klein, Mark und Rick Kazman: *Attribute-Based Architectural Styles*. Technischer Bericht CMU/SEI-99-TR-022, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Oktober 1999.

[KKB+98] Kazman, R., M. Klein, M. Barbacci, T. Longstaff, H. Lipson und J. Carriere: *The architecture tradeoff analysis method*. In: *Fourth IEEE International Conference on Engineering of Complex Computer Systems. ICECCS '98. Proceedings.*, Seiten 68–78, 1998.

[KKB+99] Klein, M.H., R. Kazman, L. Bass, J. Carriere, M. Barbacci und H. Lipson: *Attribute-Based Architecture Styles*. In: *Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1), San Antonio, TX, 225*, Band 243, 1999.

[KKC00] Kazman, R., M. Klein und P. Clements: *ATAM: Method for Architecture Evaluation*. Technischer Bericht CMU/SEI-2000-TR-004, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, August 2000.

[Kos05] Koschke, R.: *Rekonstruktion von Software-Architekturen*. Informatik-Forschung und Entwicklung, 19(3):127–140, 2005.

[Kru95] Kruchten, Philippe B.: *The 4+ 1 View Model of architecture.* IEEE Software, 12(6):42–50, November 1995.

[LBvVB02] Lassing, Nico, PerOlof Bengtsson, Hans van Vliet und Jan Bosch: *Experiences with ALMA: Architecture-Level Modifiability Analysis.* Journal of Systems and Software, 61(1):47–57, März 2002.

[LC05] Lee, Younbok und Ho-Jin Choi: *Experience of Combining Qualitative and Quantitative Analysis Methods for Evaluating Software Architecture.* In: *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, Seiten 152–157, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[Mal07] Malich, Stefan: *Ein pattern-basiertes Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen.* Doktorarbeit, Universität Duisburg-Essen, Oktober 2007.

[Mal08] Malich, Stefan: *Qualität von Softwaresystemen: Ein pattern-basiertes Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen.* Gabler Edition Wissenschaft, Mai 2008.

[MEH01] Maier, Mark W., David Emery und Rich Hilliard: *Software architecture: Introducing IEEE Standard 1471.* Computer, 34(4):107–109, April 2001.

[MRB97] Martin, Robert, Dirk Riehle und Frank Buschmann (Herausgeber): *Pattern languages of program design 3*, Band 3 der Reihe *Software Patterns Series.* Addison-Wesley Longman Publishing Co., 1997.

[NW86] Niemann, G. und H. Winter: *Maschinenelemente: Band III: Schraubrad-, Kegelrad-, Schnecken-, Ketten-, Riemen-, Reibradgetriebe, Kupplungen, Bremsen, Freiläufe.* Springer Verlag, Berlin Heidelberg New York Tokyo, zweite, völlig neubearbeitete Auflage, 1986.

[NW03] Niemann, G. und H. Winter: *Maschinenelemente: Band 2: Getriebe allgemein, Zahnradgetriebe-Grundlagen, Stirnradgetriebe.* Springer, Berlin Heidelberg New York Tokyo, 2., völlig neubearb. Aufl., 2. ber. Nachdr., korr. Nachdr. Auflage, 2003.

[NWH05] Niemann, G., H. Winter und B.-R. Höhn: *Maschinenelemente: Band 1: Konstruktion und Berechnung von Verbindungen, Lagern, Wellen.* Springer, Berlin Heidelberg New York Tokyo, 4., bearbeitete Auflage, 2005.

[Obj07a] Object Management Group: *Unified Modeling Language (OMG UML), Infrastructure, V2.1.2*. Technischer Bericht formal/2007-11-04, Object Management Group, November 2007.

[Obj07b] Object Management Group: *Unified Modeling Language (OMG UML), Superstructure, V2.1.2*. Technischer Bericht formal/2007-11-02, Object Management Group, November 2007.

[OKK+02] Obbink, Henk, Philippe Kruchten, Wojtek Kozaczynski, Rich Hilliard, Alexander Ran, Herman Postema, Lutz Dominick, Rick Kazman, Will Tracz und Ed Kahane: *Report on Software Architecture Review and Assessment (SARA)*, Februar 2002. Version 1.0.

[Par72] Parnas, D. L.: *On the criteria to be used in decomposing systems into modules*. Communications of the ACM, 15(12):1053–1058, 1972.

[PBG07] Posch, Torsten, Klaus Birken und Michael Gerdom: *Basiswissen Softwarearchitektur: Verstehen, entwerfen, wiederverwenden*. dpunkt.verlag, 2., überarbeitete und erweiterte Auflage, Januar 2007.

[Pet05] Pettersson, Niklas: *Measuring precision for static and dynamic design pattern recognition as a function of coverage*. SIGSOFT Softw. Eng. Notes, 30(4):1–7, Juli 2005.

[Pon01] Pont, M.J.: *Patterns for time-triggered embedded systems: building reliable applications with the 8051 family of microcontrollers*. ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 2001.

[PW85] Parnas, David L. und David M. Weiss: *Active design reviews: principles and practices*. In: *ICSE '85: Proceedings of the 8th international conference on Software engineering*, Seiten 132–136, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press.

[RH06] Reussner, Ralf und Wilhelm Hasselbring (Herausgeber): *Handbuch der Software-Architektur*. dpunkt.verlag, 2006.

[RR06] Robertson, Suzanne und James Robertson: *Mastering the Requirements Process*. Addison-Wesley, Upper Saddle River, NJ, zweite Auflage, 2006.

[SEI08] Software Engineering Institute, Carnegie Mellon University: *Published Software Architecture Definitions*. Web-Publikation, Januar 2008. http://www.sei.cmu.edu/architecture/published_definitions.html.

[SFBH+05] Schumacher, Markus, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann und Peter Sommerlad: *Security Patterns: Integrating Security and Systems Engineering.* John Wiley & Sons, Ltd, 2005.

[SG06] Schneider, Klaus-Jürgen und Alfons Goris: *Bautabellen für Architekten.* Werner Verlag, 17. Auflage, September 2006.

[Sha01] Shaw, M.: *The coming-of-age of software architecture research.* In: *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, Seiten 656–664, 2001.

[SMC74] Stevens, W., G. Myers und L. Constantine: *Structured Design.* IBM Systems Journal, 13(2):115–139, Mai 1974.

[Smi90] Smith, Connie U.: *Performance Engineering of Software Systems*, Band 1 der Reihe *The SEI series in software engineering.* Addison-Wesley, 1990.

[SNH95] Soni, Dilip, Robert L. Nord und Christine Hofmeister: *Software architecture in industrial applications.* In: *ICSE '95: Proceedings of the 17th international conference on Software engineering*, Seiten 196–207, New York, NY, USA, April 1995. ACM.

[SRG95] Sha, Lui, Ragunathan Rajkumar und Michael Gagliardi: *A Software Architecture for Dependable and Evolvable Industrial Computing Systems.* Technischer Bericht CMU/SEI-95-TR-005, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Juli 1995.

[SRSB00] Schmidt, D. C., H. Rohnert, M. Stal und F. Buschmann: *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects*, Band 2 der Reihe *Wiley Series in Software Design Patterns.* John Wiley & Sons, Inc. New York, NY, USA, 2000.

[TA05] Tyree, Jeff und Art Akerman: *Architecture Decisions: Demystifying Architecture.* IEEE Software, 22(2):19–27, 2005.

[TKL08] Tang, Antony, Fei-Ching Kuo und Man F. Lau: *Towards Independent Software Architecture Review.* In: *Proceedings of the Second European Conference on Software Architecture, ECSA 2008 Paphos, Cyprus*, Band 5292 der Reihe *Lecture Notes in Computer Science*, Seiten 66–81. Springer Berlin / Heidelberg, Oktober 2008.

[VAC+05] Vogel, Oliver, Ingo Arnold, Arif Chughtai, Edmund Ihler, Uwe Mehlig, Thomas Neumann, Markus Völter und

Uwe Zdun: *Software-Architektur: Grundlagen - Konzepte - Praxis*. Elsevier, Spektrum, Akademischer Verlag, 2005.

[VCK96] Vlissides, John M., James O. Coplien und Norman L. Kerth (Herausgeber): *Pattern languages of program design 2*, Band 2 der Reihe *Software Patterns Series*. Addison-Wesley Longman Publishing Co., 1996.

[WB99] Woods, Steve G. und Mario Barbacci: *Architectural Evaluation of Collaborative Agent-Based Systems*. Technischer Bericht CMU/SEI-99-TR-025, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, Pennsylvania 15213, Oktober 1999.

[ZBJ04] Zhu, Liming, Muhammad Ali Babar und Ross Jeffery: *Mining Patterns to Support Software Architecture Evaluation*. In: *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, Seiten 25–34, Juni 2004.

[Zim95] Zimmer, Walter: *Relationships between design patterns*. In: *Pattern languages of program design*, Seiten 345–364. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.