Ruprecht-Karls-Universität Heidelberg
Lehrstuhl für Software Engineering
Prof. Dr. Barbara Paech

Nachvollziehbare Entscheidungsprozesse in der industriellen Softwareentwicklung durch Rationale Management

Ansätze zur Umsetzung bei der SAP AG

Master's Thesis

Eingereicht von

Philipp Häfele
Matrikelnummer 2222770
E-Mail PHaefele@web.de

Betreut von

Prof. Dr. Barbara Paech (Universität Heidelberg)

Externer Betreuer

Dipl.-Math. Markus Gebhard (SAP AG)

Abgabetermin

30.04.2007

SOFTWARE ENGINEERING HEIDELBERG

Danksagung

Danken möchte ich Frau Prof. Dr. Paech für die betreuenden Gespräche und die Begleitung während der Arbeit und die Ideen und Ratschläge, mit denen sie mir stets – auch während meines Studiums – hilfreich zur Seite stand. Dank gebührt auch Markus Gebhard, der mich mit seinem Wissen und seinen Erfahrungen jederzeit unterstützte und dessen Diskussionsfreude und konstruktive Kritik wesentlich zum Gelingen der Arbeit beitrugen. Anne danke ich für ihre Geduld, den Ausgleich und die motivierenden Worte. All dies zusammen hat ebenfalls zum Erfolg der Arbeit beigesteuert. Nicht zuletzt bedanke ich mich bei meinen Eltern dafür, dass sie mir mein Studium ermöglicht und mir somit die besten Voraussetzungen für die Zukunft geschaffen haben.

Inhalt

Abkürzungsverzeichnis

AG	Aktiengesellschaft
AP	Application Platform (Anwendungsplattform)
ARIS	Architektur integrierter Informationssysteme
BPP	Business Process Platform
Dev	Development (Entwicklung)
DRL	Decision Representation Language
et al.	et alii (und andere Verfasser)
f.	folgende
ff.	fortfolgende
Hrsg.	Herausgeber
HTML	Hypertext Markup Language
IBIS	Issue-Based Information System
IDE	Integrated Development Environment (integrierte Entwicklungsumgebung)
InfoDev	Information Development
ISO	International Organization for Standardization (Internationale Organisation für Normung)
MD(S)D	Model Driven (Software) Development
MDA	Model Driven Development
MDD	siehe MD(S)D
NASA	National Aeronautics and Space Administration
NFR	Nonfunctional Requirement (nichtfunktionale Anforderung)
o.J.	ohne Jahr
ORB	Rationale for organizing bodies of knowledge
PHI	Procedural hierarchy of issues
PIL	Product Innovation Lifecycle
PIL@AP	Product Innovation Lifecycle at Application Platform
PPC	Product Portfolio Council
PPR	Project and product rationale
PRIO	Priority (Priorität)
QOC	Questions, Options, and Criteria
RE	Requirements Engineering
REPARE	Requirements Engineering Pattern Repository
REPI	Requirements Engineering Process Improvement
REQ	Requirement (Anforderung)
RM	Rationale Management
SAP	Systeme, Anwendungen und Produkte in der Datenverarbeitung
SE	Software Engineering
SEURAT	Software Engineering Using RATionale
SM	Solution Management
SME	Small and Midsize Enterprises
SOA	Service-oriented Architecture
SRS	Software Requirements Specification
SSO	Single Sign-On
SW	Software
TCO	Total Cost of Ownership
UI	User Interface (Benutzungsschnittstelle)
UML	Unified Modeling Language
vgl.	vergleiche
XML	Extensible Markup Language

"The traditional and onerous goal of generating (and then not using) more system documentation, has to be overtaken by the objective of creating better documentation."

John M. Carroll, [Carr06, S. VIII]

Kapitel 1

Einleitung

Das Software Engineering ist heute geprägt von schnelllebigen Technologien und Trends, verteilten, arbeitsteiligen Projekten mit vielen Interessenvertretern (Stakeholdern) und einem umkämpften globalen Markt. In diesem Umfeld wird es zur Herausforderung, gewonnene Erkenntnisse festzuhalten und daraus als gesamte Organisation zu lernen, um Fehler nicht zu wiederholen oder gar ganz zu vermeiden. Information und Wissen sowie Kommunikation und Wissenstransfer zusammen mit wohl reflektierten Entscheidungen werden daher immer mehr zum kritischen Erfolgsfaktor einer Organisation.
Rationale Management (RM) (engl. Begründungsmanagement) ist ein Ansatz, um diese Erfolgsfaktoren auf allen Ebenen einer Organisation einzubeziehen. Durch das Aufzeigen der Gestaltungsalternativen (*design space analysis*) sollen die Entscheidungsträger (gegebenenfalls in einem kollaborativen Prozess) anhand von adäquaten Kriterien zu konsistenten und argumentativ begründeten Entscheidungen gelangen. Über eine geeignete Erfassung, Aufbereitung und Nutzung des Rationale soll eine Verbesserung der Entscheidungskommunikation, des Wissenstransfers und der Informationsgewinnung über individuelle, organisatorische und funktionale Grenzen der Organisation hinweg erreicht werden.

1.1 Problemstellung

Wie die Erfahrung der vergangenen Jahre zeigt, werden Softwareprojekte heute immer größer, finden geographisch an verteilten Orten statt und müssen sich mit sich ständig weiterentwickelnden Technologien auseinandersetzen. Um diese Projekte zu einem erfolgreichen Abschluss zu bringen, muss die große Zahl der Interessenvertreter effizient miteinander kommunizieren und richtig koordiniert werden.
Eine Möglichkeit, dies zu unterstützen, ist die Etablierung von Softwareprozessmodellen (beziehungsweise Vorgehensmodellen) [Somm01, S. 24], die einen gemeinsamen Rahmen zur Verfügung stellen, um alle Interessenvertreter z.B. in Hinblick auf deren Rollen, Aktivitäten und Werkzeuge, sowie weitere normative Vorgaben für das Projekt auf ein gleiches Verständnisniveau zu heben. Softwareprozessmodelle richten sich dabei verstärkt nach einem ingenieurmäßigen und modellgetriebenen Vorgehen aus. Dies bedeutet, dass wichtige Entscheidungen häufig von einer kleineren Anzahl von Softwaredesignern und -architekten getroffen werden, sich dann in Spezifikationen niederschlagen und von einer größeren Zahl Programmieren umgesetzt werden, welche sich teils geographisch entfernt befinden können. Die Kommunikation von Entscheidungen und deren Verständlichkeit wird somit umso wichtiger. So sollen die getroffenen Entscheidungen möglichst viele Realisierungsalternativen

berücksichtigt haben und rational begründet sein. Das Interesse der Organisation sollte darin liegen, dass das Wissen über Produkt, Projekt und Domäne unabhängig vom einzelnen Individuum geteilt und Entscheidungen verständlich und nachvollziehbar gemacht werden. Mit dem Softwareprozessmodell einhergehen muss als kritischer Erfolgsfaktor also ein transparenter und überzeugender Entscheidungsprozess, der alle Interessenvertreter bei der Entscheidungsfindung, der Kommunikation der Entscheidung und deren Umsetzung unterstützt [DMMP06a, S. IX].
In der Praxis kann beobachtet werden, dass Entscheidungen intuitiv und ohne vorherige Betrachtung von Alternativen gefällt und dann ohne ausführliche Begründung kommuniziert werden. Hier stellt sich die Frage, (1) inwieweit die Entscheidung wirklich die Bedürfnisse der Organisation oder die Anforderungen an ein Produkt widerspiegelt. Weiterhin resultieren solche Entscheidungen bei den Umsetzenden oft in (2) wiederkehrenden Diskussionen, aufgrund von Spekulationen über deren Hintergründe oder aufgrund von mangelnder Akzeptanz oder mangelnder Verständlichkeit. Häufig kann auch festgestellt werden, dass einmal getroffene Entscheidungen wegen fehlender Dokumentation (3) nicht wieder rekapitulierbar sind. Man kann sich nicht erinnern, warum eine Architekturentscheidung so gefällt wurde, warum eine zu erledigende Aufgabe (*Action Item*) in einem Gesprächsprotokoll festgehalten wurde und was zu tun ist. Oder man weiß nicht mehr, wie und warum man zu einer finalen Entscheidung bei einem Treffen gekommen ist. Die fehlende Dokumentation erschwert beispielsweise auch (4) eine Entscheidungsrevision bei Änderung des ursprünglichen Entscheidungskontextes beziehungsweise (5) den Wissenstransfer zwischen verschiedenen Projektbeteiligten und -phasen (z.B. Verantwortlichkeitswechsel).
Diese Probleme, die sich in der genannten Form auf produktspezifische System- (z.B. Spezifikationen) und Prozessentscheidungen (z.B. Projektplan) beziehen, gelten auch für organisationsspezifische System- (z.B. Entwurfsmuster, Domänenmodelle) und Prozessentscheidungen (z.B. Prozessmodelle, Best Practices) und sind in diesem Umfeld von gleicher Wichtigkeit.[1]

1.2 Zielsetzung

Die Anwendung von Rationale Management ist in der Praxis noch nicht weit verbreitet und geht über die prototypische Anwendung kaum hinaus. Das **Ziel** der Arbeit ist es, die in der Wissenschaft diskutierten Konzepte auf die Anforderungen eines in der Praxis etablierten Softwareprozessmodells zu übertragen und die dabei gewonnenen Erkenntnisse festzuhalten.

Die Arbeit liefert in diesem Zusammenhang eine Zusammenstellung der Konzepte des Rationale Management, um

- die Grundlagen für die weiteren Ausführungen zu schaffen und auf Basis dieser Grundlagen eine Anwendung zu diskutieren,
- dem interessierten Leser aus der industriellen Praxis die Ideen und Intentionen von Rationale Management näher zu bringen, um diese mit deren Erfahrung und Expertise anwenden zu können und
- die Konzepte dann auf das Umfeld des Softwareprozessmodells übertragen zu können.

Anwendungsgegenstand für die Übertragung bildet das in einer Teilorganisation der SAP AG entwickelte und eingesetzte Softwareprozessmodell, welches die dort verwendeten Prozesse der Produktentwicklung bündelt.

Um die Zielsetzung der Arbeit zu erreichen dienen vier **Fragen als Leitfaden**:

1. Was sind die grundlegenden Konzepte und Ideen von Rationale Management?

[1] Zu einer vertiefenden Darstellung von Wissens- und Entscheidungsbereichen im Software Engineering vgl. Kapitel 2.3.

2. Welche Umsetzungsvorschläge für die Praxis gibt es für diese Konzepte und Ideen?
3. Wie kann eine Umsetzung der Konzepte und Ideen im Umfeld eines konkreten, in der Praxis etablierten Softwareprozessmodells aussehen?
4. Was sind die bei der Bearbeitung des Themas gewonnenen Erkenntnisse, insbesondere bei der Übertragung von Rationale Management in das vorhandene Praxisumfeld?

Die Arbeit **umfasst keine** konkrete Implementierung eines Werkzeugs oder ein unverändert übernehmbares Gesamtkonzept für die Integration von Rationale Management in den Kontext des Softwareprozessmodells. Der Grund hierfür ist, dass ein solches Prozessmodell und dessen gelebte Umsetzung im Rahmen der Masterarbeit nicht so detailliert erfasst werden kann. Daher stellt die Arbeit die Thematik eher in der Breite dar und zeigt verschiedene Anwendungsmöglichkeiten auf, als an einer Stelle zu sehr in die Tiefe abzutauchen. Damit soll Anwendern in der Praxis ein Ausgangspunkt für die weitere Beschäftigung mit der Thematik des Rationale Management aufgezeigt werden.

1.3 Vorgehen und Struktur

Bei der Erstellung der Arbeit wurde in **drei Schritten** vorgegangen:

1. Zur Schaffung der theoretischen Grundlage und zur Gewinnung eines Überblicks über das Themengebiet Rationale Management wurde eine Literaturanalyse durchgeführt. Dabei wurde als Ausgangspunkt vor allem die in Kapitel 1.4 genannte Literatur verwendet.
2. In einem weiteren Schritt wurde eine gründliche Einarbeitung in das Anwendungsgebiet bei der SAP AG absolviert. Diese setzte sich sowohl aus Literatur- und Dokumentenanalyse von SAP Materialien und Quellen als auch aus Gesprächen mit Mitarbeitern der SAP AG und Beobachtungen vor Ort bei der SAP AG zusammen.
3. Der letzte Schritt, zu dem ebenfalls Diskussionen mit einem Mitarbeiter der SAP AG durchgeführt wurden, bestand in der Synthese dieser beiden Gebiete.

Der erste Schritt diente der Behandlung der ersten und zweiten Leitfrage (Kapitel 1.2), der zweite und dritte ermöglichten aufbauend auf dem ersten die Adressierung der letzen beiden Fragestellungen.

Die weitere Struktur der Arbeit ist wie folgt:

Kapitel 2 führt die Terminologie und Grundlagen von Rationale Management ein, auf die im Laufe der Arbeit zurückgegriffen wird. Dabei wird sowohl auf statische Aspekte der Rationalerepräsentation, als auch auf dynamische Aspekte der Prozessimplementierung von Rationale Management mit Aufgaben und Rollen eingegangen. Durch letztere wird auch der Zusammenhang von Rationale Management und Wissensmanagement aufgezeigt und Rationalefragestellungen in den verschiedenen Wissensbereichen des Software Engineering (als dessen Anwendungsbereiche) beschrieben. Die Wissensbereiche werden in Kapitel 3 weiter zur Gliederung der Vorstellung der einzelnen Rationale Ansätze genutzt.

In Kapitel 3 werden – strukturiert nach den verschiedenen Wissensbereichen – Rationale Ansätze vorgestellt, die für den Einsatz in der Praxis diskutiert werden. Hiermit soll ein Eindruck vermittelt werden, wie unterschiedlich deren Ausprägung je nach Anwendungsgebiet ist.

Kapitel 4 beschäftigt sich mit der Anwendung von Rationale Management im Kontext des Softwareprozessmodells PIL@AP. Hierzu werden die für die weiteren Ausführungen wichtigen Aspekte des Prozessmodells eingehend beschrieben. Für die Übertragung der Konzepte des Rationale Management auf den Anwendungsbereich, werden die Ziele, die mit Rationale Management dort erreicht werden sollen, festgelegt (Zieldefinition). Weiterhin werden Kriterien zur Bewertung von Rationale Ansätzen aufgestellt, die für die Diskussion

der Vorschläge herangezogen werden. Anschließend werden die Vorschläge diskutiert und vorgestellt und entsprechendes Rationale für die Entscheidungen angegeben.

Kapitel 5 fasst die Ergebnisse der Arbeit zusammen und gibt einen Ausblick auf zukünftige Arbeitsbereiche. Die Zusammenfassung enthält die während der Arbeit gewonnenen Erkenntnisse und adressiert damit im Besonderen die vierte Leitfrage (Kapitel 1.2).

1.4 Literatur und Quellenlage

Im Folgenden soll kurz vorgestellt werden, welche Quellen primär für die Arbeit herangezogen wurden und welche Literatur bei weiterer Beschäftigung mit gewissen Teilbereichen von Rationale einen ersten Ausgangspunkt bietet.
Die für die Arbeit verwendete Literatur zum Thema Rationale besteht vor allem aus einzelnen Veröffentlichungen in Zeitschriften, beziehungsweise aus den beiden Sammelbänden „Design Rationale“ [MoCa96a] und „Rationale Management in Software Engineering“ [DMMP06a]. Die Sammelbände präsentieren eine ausführliche Zusammenstellung der Forschungsfragestellungen und des Forschungsstandes. Der erste Sammelband erschien 1996 und umfasst vor allem konzeptuelle und methodische Analysen, Diskussionen über Notationen und Ontologien, kleinere Experimente und begrenzte Feldstudien [Carr06, S. VIII]. Letzterer erschien 2006 als aktualisierter Überblick über das Forschungsgebiet. Er bietet einen guten Ausgangspunkt zur vertieften Beschäftigung mit der Thematik, da er zu Anfang einen breiten Abriss über Rationale Ansätze im Allgemeinen und innerhalb des Software Engineering im Speziellen gibt. Ausführlicher behandelt werden dann in weiteren Teilen des Werkes die Probleme von Rationale Management und deren Lösungsmöglichkeiten, Anwendungsmöglichkeiten im Requirements Engineering, während der Architekturdefinition und zur Organisation und Wiederverwendung von Wissen in Softwareentwicklungsorganisationen. Eine Reihe von Werkzeugen zur Erfassung von Rationale werden in diesem Zusammenhang ebenfalls vorgestellt.
Eine kurze und prägnante Einführung in die Thematik, deren Verbindung zu Knowledge Management und einige Beispiele für den Einsatz von Rationale in der Praxis bietet auch der Beitrag über Rationale Management im „Handbook of Software Engineering and Knowledge Engineering“ [DuPa01].
Oft wird versucht, die einzelnen Rationale Ansätze und deren Schemata zu vergleichen. Dies ist auch Teil einer Veröffentlichung von Louridas und Loucopoulos [LoLo00], die einige prominente Rationale Ansätze mit dem Ziel der Entwicklung eines generischen Ansatzes gegenüberstellt.
Weiterhin beinhaltet diese Veröffentlichung – wie auch das Einführungskapitel des 2006 erschienen Sammelbandes – eine Zusammenstellung der Charakteristika der Rationale Ansätze, die über eine präzise Begriffsbildung und Differenzierung die wissenschaftliche Diskussion und Klassifizierung der Ansätze unterstützt.
Da Rationale Management an der Schnittstelle von Wissensmanagement und Entscheidungstheorie angesiedelt ist, empfiehlt sich weiterhin die Lektüre von Rus, Lindvall und Sinha [RuLS01] beziehungsweise Howard [Howa68] und Howard et al. [HoRM02].
Als allgemeine Literatur zum Thema Softwareprozessmodelle wurden für die Arbeit Standardwerke im Bereich des Software Engineering von Balzert [Balz98; Balz00] und Sommerville [Somm01] verwendet.
Die Quellen im Zusammenhang mit dem in dieser Arbeit untersuchten Anwendungsgebiet für Rationale Management bei der SAP AG (Kapitel 4) sind vorwiegend interne Dokumente der SAP AG. Somit sind die Quellen für dieses Kapitel nicht zugänglich, werden aber dennoch im Literaturverzeichnis mit dem Vermerk „SAP Dokument“ angegeben. Es existiert allerdings eine potentiell für Externe zur Verfügung stehende Prozessbeschreibung des bei der SAP AG praktizierten Standardprozesses für Softwareentwicklung „Product Innovation Lifecycle“ (PIL) [DISZ04].

Kapitel 2

Begriffsdefinitionen und Grundlagen

In diesem Kapitel sollen die Grundlagen vorgestellt werden, auf die im Laufe der Arbeit zurückgegriffen wird.
Kapitel 2.1 klärt hierzu die wichtigsten Termini im Umfeld von Rationale Management und stellt die am häufigsten verwendeten Konzepte und Ideen vor. Kapitel 2.1.1 beschäftigt sich zu diesem Zweck mit Entscheidungen und Entscheidungsfindung als Ausgangspunkt für die Erfassung von Rationale. Nach der Klärung der Begriffe Rationale, Rationale Management und Rationale Ansätze werden Charakteristika der Rationale Ansätze vorgestellt, die in der Diskussion über Rationale häufig Verwendung finden und eine grobe Einordnung der Ansätze ermöglichen (Kapitel 2.1.2 und Kapitel 2.1.3). Die weiteste Verbreitung genießen argumentations-basierte Rationale Ansätze, die ein semi-formales Schema verwenden, um das Rationale zu erfassen. Die wichtigsten stellt Kapitel 2.1.4 vor, wobei auch ein Ausblick auf nicht argumentations-basierte Ansätze gegeben wird.
In Kapitel 2.2 steht die prozessorientierte, dynamische Sichtweise auf Rationale Management im Vordergrund: Was ist der erwartete Nutzen beim Einsatz von Rationale Management im Software Engineering (Kapitel 2.2.1), wie sieht die Durchführung von Rationale Management in Hinsicht auf Aufgaben und Rollen aus und was sind die Herausforderungen, denen bei der Anwendung von Rationale Management begegnet werden muss (Kapitel 2.2.2).
In Kapitel 2.3 wird der Zusammenhang von Entscheidungen in den verschiedenen Wissensbereichen des Software Engineering und dem zugehörigen Rationale hergestellt. Hierzu werden die Wissensbereiche vorgestellt, anhand derer in Kapitel 3 auch die Vorstellung der Rationale Ansätze in der praktischen Anwendung stattfindet.
Eine Zusammenfassung der Ausführungen schließt das Kapitel ab (Kapitel 2.4).

2.1 Rationale und Rationale Management

Im Folgenden sollen die Termini **Rationale** und **Rationale Management** definiert und beschrieben werden, um ein grundlegendes Verständnis für die weiteren Ausführungen zu erhalten. Um auf die Begriffe hinzuführen, soll zunächst kurz auf menschliche Entscheidungen eingegangen werden, denn hinter Rationale verbirgt sich – ganz allgemein – die Motivation beziehungsweise Begründung hinter einer Entscheidung [DMMP06b, S. 1].
Rationale Ansätze zielen in diesem Zusammenhang auf die Erfassung, Repräsentation und Pflege (*maintenance*) von Aufzeichnungen des „Warum“ einer Entscheidung zusammen mit ihrem Ergebnis [DuPa01, S. 787]. Sie bezwecken, Rationale in der alltäglichen Arbeit handhabbar zu machen. Die grundlegenden Charakteristika und die Repräsentationsformen für Rationale, die diese Ansätze vorschlagen, sowie einige verbreitete Ansätze werden ebenfalls vorgestellt.

2.1.1 Entscheidungen und Entscheidungsfindung

Eine **Entscheidung** ist der „Akt der verbindl[ichen] Wahl einer von mehreren Möglichkeiten“ (Alternativen, Optionen) [Meye73]. Nach [Herr06] kann eine Entscheidung auf verschiedene Weisen durch einen oder mehrere **Entscheidungsträger** zustande kommen. Sie kann unbewusst oder bewusst gefällt werden, spontan, emotional, zufällig oder rational, subjektiv oder objektiv. Sie basiert auf der impliziten oder expliziten Bewertung des aktuellen Kontextes (Problem, Alternativen, Informationen, Ziele etc.), in dem die Entscheidung zu treffen ist. Eine Entscheidung wird als rational getroffen bezeichnet, wenn sie sich an einem oder mehreren Entscheidungskriterien orientiert, gegen die die Alternativen bewertet werden. Anhand von dieser Bewertung wird dann die „beste“ Alternative gewählt, d.h. die Entscheidung gefällt.
Die Wahl einer Alternative zielt auf deren Umsetzung, welche daraufhin in einer bestimmten Weise Auswirkungen auf die Umgebung hat (Konsequenzen, Folgen). Werden diese wahrgenommen und sind diese (eindeutig) auf die Entscheidung zurückzuführen, so kann der **Erfolg der Entscheidung** bestimmt werden. Die **Güte** der Entscheidung ist ein Maß der Erfüllung der angesetzten Kriterien, welche der Entscheidung zugrunde gelegt wurden.
Wurde die Entscheidung rückblickend analysiert und bewertet, so lassen sich Entscheidungsmuster für gleich gelagerte Entscheidungsfälle in der Zukunft ableiten. Aber auch ähnliche Problemstellungen können über Analogiebildung beziehungsweise Transferleistungen bearbeitet werden. Diese Anwendung von **Erfahrungen** aus vergangenen Entscheidungen spiegelt den **Lernprozess** wider.
Herrmann und Paech [HePa06] haben ein Modell entwickelt (Abbildung 1), welches die Aktivitäten (ovale Elemente in Abbildung 1) der rationalen Entscheidungsfindung,[2] deren Zusammenhang mit entscheidungsrelevanten Erfahrungen (Wissen) und dem Lernprozess darstellt (Elemente mit abgerundeten Ecken in Abbildung 1). Nach diesem Modell kann die zu treffende Entscheidung generell als Frage (-stellung) repräsentiert werden. Daraufhin werden Alternativen erzeugt, die als Wahlmöglichkeiten für die Entscheidung in Frage kommen. Ist der Entscheidungskontext bekannt (oder ähnlich zu einem bekannten), so kann vorhandenes Wissen (gegebenenfalls durch Analogiebildung/Transferleistungen) angewendet werden. Dazu wird der Fragen- und/oder Lösungspool herangezogen, der entsprechende Fragestellungen beziehungsweise Lösungsalternativen aus der Vergangenheit (angeeignetes Wissen, Erfahrung) enthält.
Nachdem Klarheit über die alternativen Wahlmöglichkeiten besteht, folgt deren Bewertung bezüglich der aufgestellten Kriterien. Auch hierfür greift man auf Wissen und Erfahrungen zurück (in Form von anwendbaren Bewertungskriterien, [3] Regeln und Richtwerten). Ähnlichem Wissen bedient man sich auch beim Treffen der Entscheidung, wobei hier Entscheidungskriterien anstatt Bewertungskriterien eine Rolle spielen. Der Unterschied besteht darin, dass für die finale Entscheidung möglicherweise andere Gewichtungen, Aggregationen, Kombinationen beziehungsweise Projektionen der Kriterien angewendet werden.
Auf die Entscheidung folgt die Umsetzung der gewählten Alternative sowie die Bewertung der Entscheidung und deren Folgen. Hier setzt der Lernprozess durch retrospektive Betrachtung der Entscheidungen ein. Über die durchgeführte Bewertung kann das Wissen – in Form von Bewertungs-, und Entscheidungskriterien, Regeln und Richtwerten, Fragen und Lösungen – gegebenenfalls korrigiert (revidiert und ergänzt) werden. Gründe für eine Korrektur können sich aus (Miss-) Erfolg der Entscheidung, neuen Schwerpunkten beziehungsweise Perspektiven oder gegebenenfalls der Notwendigkeit ergeben, weitere/andere Problem- und Fragestellungen beziehungsweise Lösungsmöglichkeiten in Betracht zu ziehen.

---

[2] Herrmann und Paech merken an, dass sich das Modell ebenfalls zur Erklärung einer intuitiven Entscheidungsfindung eignet. Allderings ist dann meist nicht die Vollständigkeit und Nachvollziehbarkeit der Schritte des Modells und/oder des eingeflossenen Wissens gegeben [HePa06, S. 24]
[3] Z.B. Softwaremetriken, betriebswirtschaftliche Größen, ISO 9126 [ISO9126] etc.

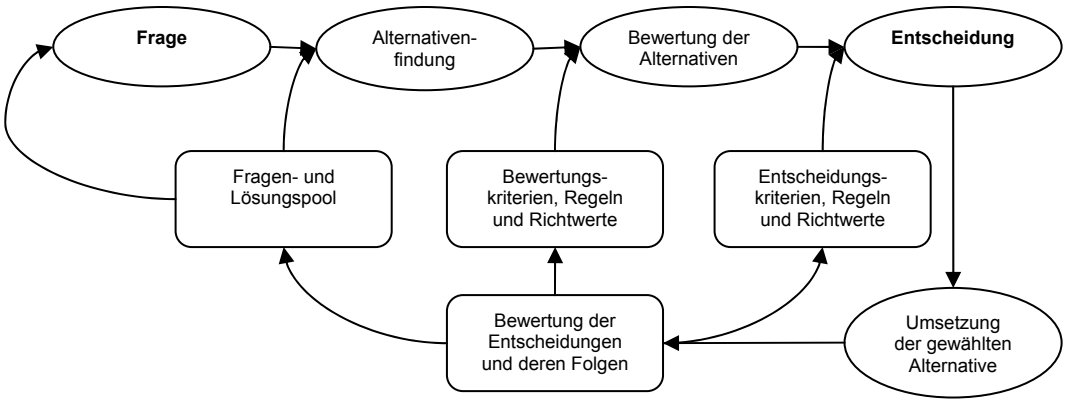

Abbildung 1: Modell einer rationalen Entscheidungsfindung (Quelle: in Anlehnung an [HePa06])

2.1.2 Die Begriffe Rationale und Rationale Management

Der Begriff **Rationale** stammt ursprünglich nicht aus der Domäne der Softwareentwicklung. Er wurde seit den 70er Jahren in verschiedenen Bereichen (z.B. Politik, Ingenieurwesen, Raum- und Gebäudeplanung) angewendet [KuRi70] und fand erst in den 80er Jahren Einzug im Software Engineering [DMMP06b, S. 3].
Anfänglich konzentrierte sich die Forschung auf die Verwendung von Rationale für das Software Design, weshalb sich der Begriff *design rationale* (DR) etablierte. In der Literatur wird der Begriff *design rationale* mit verschiedenen Schwerpunkten verwendet, die von Moran und Carroll [MoCa96b, S. 8] zusammengefasst und als Definitionen festgehalten wurden.[4] Danach ist **design rationale**:

„1. An expression of the relationships between a designed artifact, its purpose, the designer's conceptualization, and the contextual constraints on realizing the purpose.
2. The logical reasons given to justify a designed artifact.
3. A notation for the logical reasons for a designed artifact.
4. A method of designing an artifact whereby the reasons for it are made explicit.
5. Documentation of (a) the reasons for the design of an artifact, (b) the stages or steps of the design process, (c) the history of the design and its context.
6. An explanation of why a designed artifact (or some feature of an artifact) is the way it is."

Design rationale bezeichnet also einerseits die externalisierten **Beziehungen** (und deren Dokumentation) zwischen einem Artefakt und

- dessen Ziel, den zugrunde liegenden Überlegungen des Erschaffers (Konzeption) und dem Kontext der Entstehung (1. Aufzählungspunkt) beziehungsweise
- den Gründen für dessen Ausprägung, den Prozessschritten und Ausprägungsstufen, die letztendlich zu dem Artefakt geführt haben sowie deren Historie (5. Aufzählungspunkt).

Andererseits bezeichnet Rationale die zielgerichtete **Begründung** beziehungsweise Rechtfertigung für eine Entscheidung, um z.B. einen anderen Interessenvertreter zu **überzeugen** (2. Aufzählungspunkt), beziehungsweise um es für andere **interpretierbar** zu machen (6. Aufzählungspunkt). Außerdem wird in einigen Veröffentlichungen unter

[4] Diese Definitionen enthalten die Bedeutungen des Begriffs „design rationale". Wie dieser in Beziehung zu dem allgemeineren Begriff „Rationale" beziehungsweise „software engineering rationale" steht, wird weiter unten erläutert.

Rationale eine **Methode**[5] zum Software Design verstanden, welche auch die Begründungen für ein Artefakt explizit macht (4. Aufzählungspunkt) beziehungsweise eine **Notation**, die die Dokumentation der Begründungen ermöglicht (3. Aufzählungspunkt). Vertiefende Ausführungen zu den verschiedenen Verwendungen des Begriffs findet man direkt bei Moran und Carroll [MoCa96b, S. 8f.].

Auch heute noch dominiert der Begriff *design rationale* [DuPa01, S. 787]. Rationale ist aber nicht auf das Software Design beschränkt und kann auch in weiteren Tätigkeitsbereichen sowohl von den Entwicklern, dem (Projekt-) Management und anderen Interessenvertretern sinnvoll angewendet werden (vgl. Kapitel 3 und Kapitel 4). Somit führen Dutoit et al. [DMMP06b, S. 26] auch den allgemeineren Begriff des **software engineering rationale** (SER) ein, der sich auf Rationale aus den verschiedensten Bereichen des Software Engineering bezieht.
In dieser Arbeit wird sich der Begriffsverwendung von Dutoit et al. [DMMP06b, S. 26] angeschlossen: der Begriff **Rationale** wird gebraucht, wenn sich die Verwendung des Begriffs nicht auf einen bestimmten Bereich während des Software Engineering bezieht, sondern wenn dieser allgemein verwendet wird. Auf diesen allgemeineren Begriff können auch obige Definitionen des Begriffs *design rationale* übertragen werden, wenn als Gegenstand der Entscheidung nicht mehr ausschließlich nur ein Artefakt aus dem Bereich des Software Designs betrachtet wird, sondern vielmehr jeder mögliche Entscheidungsgegenstand.
Unter **Rationale Management** (RM) (engl.: Begründungsmanagement) versteht man die Unterstützung der expliziten Entscheidungsfindung [DMMP06a, S. IX] sowie die Dokumentation der Gründe, die zu einer Entscheidung geführt haben und deren Kommunikation. Es beinhaltet Aufgaben, Fähigkeiten, Methoden und Mittel, die zur Erfassung, Pflege, Verwaltung und Nutzung von Rationale notwendig sind.

2.1.3 Rationale Ansätze

Im Rahmen der Forschungsarbeiten zu Rationale haben sich viele **Rationale Ansätze** (*rationale approaches*) entwickelt, die sich mit der Anwendung von Rationale im Arbeitskontext beschäftigen und dazu Aussagen zu folgenden Bereichen machen:

- Repräsentation von Rationale (**Rationalerepräsentation**), meist verbunden mit einem Schema, welches die einzelnen Elemente von Rationale und deren Beziehungen zueinander aufnimmt
- Integration von Rationale in den Arbeitsablauf (**Prozessimplementierung**), d.h. Einbeziehung der Aufgaben und Aktivitäten des Rationale Management (u.a. Identifikation, Erfassung, Entwicklung und Nutzung des Rationale, vgl. Kapitel 2.2.2.1) in den Softwareentwicklungsprozess

Um die Vielzahl der verschiedenen Ansätze zu strukturieren, bedarf es der Identifikation von Charakteristika zu deren Klassifizierung. In Tabelle 1 sind die Kriterien zur Klassifizierung aufgelistet, welche sowohl in Dutoit et al. [DMMP06b, S. 4ff.] sowie Louridas und Loucopoulos [LoLo00, S. 212ff.] genannt werden.

2.1.4 Repräsentation von Rationale

Die Repräsentation von Rationale kann auf unterschiedliche Weise erfolgen. Dutoit und Paech [DuPa01, S. 790] nennen die Möglichkeit, dies

- als Begründungen in natürlicher Sprache,
- als Regeln in wissensbasierten Systemen oder

[5] Methode hier in der Bedeutung als Oberbegriff von Konzepten, Notation und methodischer Vorgehensweise (vgl. hierzu [Balz00, S. 37])

- als Argumentation in strukturierten Schemata

zu tun. Letzteres wird auch **argumentations-basiertes** (argumentation-based) **Rationale** genannt und ist die am weitesten verbreitete Art, Rationale zu repräsentieren. Diese Art verspricht ein angemessenes Gleichgewicht zwischen Formalität und Offenheit sowie einen großen Fundus an relevanter Literatur, weshalb diese Möglichkeit in der Arbeit weiter verfolgt wird. Die Repräsentation von argumentations-basiertem Rationale resultiert meist[6] in einem **Graph** (vgl. Abbildung 18, S. 65 und Abbildung 21, S. 86), in dem jeder Knoten eine Fragestellung beziehungsweise ein zugehöriges Diskussions-, Argumentations- oder Schlussfolgerungselement repräsentiert [DuPa01, S. 790]. Durch die Kanten zwischen den Knoten werden Eigenschaftsbeziehungen im Rationale ausgedrückt (z.B. positiver/negativer Einfluss).

Charakteristik	**Bedeutung**
Repräsentationsform und Prozessimplementierung (Representation and Process Implementation)	
Repräsentationsform	Art und Weise, in der Rationale erfasst und dargestellt wird. Meist unterscheidet man Repräsentationsformen in ihrem Grad der Formalität: formal, semi-formal, informal.
Prozessimplementierung	Hierunter wird die Integration von Rationale Management in den zugrunde liegenden Prozessablauf verstanden. D.h. wie die Aufgaben von Rationale Management (z.B. Erfassung, Entwicklung, Nutzung) in dem übergeordneten Arbeitsablauf implementiert werden.
Beschreibend/erläuternd/darstellend/deskriptiv vs. verordnend/normativ/Vorschriften machend/präskriptiv (descriptive vs. prescriptive)	
Deskriptiv, beschreibend (descriptive)	Rationale Ansatz, der versucht, die Gedankengänge von Entscheidungsträgern zu erfassen und darzustellen.
Präskriptiv, verordnend (prescriptive)	Rationale Ansatz, der auf eine Verbesserung der Argumentation und Entscheidungsfindung zielt. Der Ansatz versucht durch Vorgaben (z.B. von Kriterien, Vorgehen, Methoden) zu korrekten, konsistenten und besser reflektierten Entscheidungen zu gelangen.
Anmerkung: deskriptiv und präskriptiv schließen sich nicht gegenseitig aus. Ein Rationale Ansatz kann also durchaus mit beiden Aspekten charakterisiert werden.	
Aufdringlichkeit (intrusiveness)	
Aufdringlich (intrusive)	Rationale Ansatz, der den Entwicklungsprozess, beziehungsweise die Art darin zu arbeiten, abändert. Die Aufdringlichkeit eines Ansatzes kann dabei für die verschiedenen Aufgaben (vgl. Kapitel 2.2.2.1) von Rationale variieren.
Nicht aufdringlich (non-intrusive)	Ein Ansatz, der transparent mit wenig Einflussnahme auf den Prozessablauf durchgeführt werden kann. *Anmerkung*: Häufig wird die Aufdringlichkeit eines Ansatzes als die größte Hürde für die Verbreitung und die Nutzung von Rationale in der Praxis gesehen, weshalb deren Reduzierung anzustreben ist.
Produkt- vs. prozessorientiert (product- vs. process-oriented)	
Produktorientiert (product-oriented)	Hierunter wird ein Ansatz verstanden, der darauf abzielt, das Rationale zu den während des Prozesses entwickelten Artefakten zu erfassen. Bei schema-basierten Ansätzen werden hier meist die Elemente des Schemas mit dem Artefakt verlinkt, um eine Verfolgbarkeit (*traceability*) herzustellen.
Prozessorientiert (process-oriented)	Ein Ansatz, der versucht, das Rationale hinter den fortlaufenden Prozessschritten und Entscheidungen zu erfassen.

Tabelle 1: Charakteristika von Rationale Ansätzen

Welche Elemente (Entitäten) im Graph repräsentiert werden, hängt vom zugrunde liegenden **Schema** (Metamodell) des Rationale Ansatzes ab. Dies entsteht **durch Modellbildung** der Entscheidungsdomäne (Kapitel 2.1.1) und dem gewünschten Anwendungskontext des Rationale (z.B. Dokumentation von Diskussionen, Dokumentation von Architekturentscheidungen). Die charakteristischen Merkmale der Entscheidungsdomäne und des Anwendungs-

[6] Es wird hier eine Einschränkung vorgenommen, da einige Ansätze das Rationale nicht explizit und vollständig als Graph repräsentieren sondern z.B. in tabellarischer Form. Sie enthalten allerdings die Elemente des zugrunde liegenden Schemas. Ein Beispiel hierfür sind die in Kapitel 3.2.2 und in Kapitel 3.3.1 vorgestellten Ansätze.

kontexts werden unter Beachtung der drei Modellmerkmale – Abbildungsmerkmal, Verkürzungsmerkmal und pragmatisches Merkmal [Stac73, S. 131ff.] – in ein Modell überführt. Beispielsweise lassen sich somit zur Repräsentation von Entscheidungen (z.B. entsprechend Kapitel 2.1.1) folgende Elemente herleiten:

- die Fragestellung, welche gelöst werden soll,
- die Optionen, die zur Lösung in Betracht gezogen werden können,
- die Argumente für und wider der Optionen,
- die Kriterien, die zur Abwägung der Optionen und damit zur Entscheidungsfindung herangezogen werden,
- die Entscheidung, die getroffen wird.

Diese Elemente bilden das Vorbild (Entscheidung und deren Kontext) ab (Abbildungsmerkmal) und verkürzen es durch Abstraktion auf die wichtigen Elemente (Verkürzungsmerkmal), die für eine bestimmte Zweckerfüllung (in einem Anwendungskontext) geeignet sind (pragmatisches Merkmal).

Welche Elemente einer Entscheidung etablierte argumentations-basierte Rationale Schemata als Modellelemente aufnehmen, soll in den folgenden Teilkapiteln vorgestellt werden. Anhand der behandelten Rationale Ansätze werden auch die Klassifizierungskriterien nochmals aufgegriffen und veranschaulicht.

2.1.4.1 Issue-Based Information System (IBIS)

Issue-Based Information System (IBIS) [KuRi70] gilt als der Vorläufer aller weiteren Rationale Ansätze. Das IBIS Schema enthält folgende Elemente:

Issues: Issues repräsentieren die Fragestellungen/Probleme, die der Diskussion zugrunde liegen und über die entschieden werden soll.
Positions: Positions sind die Lösungsalternativen/-optionen für eine *Issue*. Wird eine Position gewählt, so spricht man von der Lösung (*Resolution*).
Arguments: Arguments unterstützen (*support*) beziehungsweise beanstanden (*reject*) *Positions* und dienen damit zu deren Bewertung.

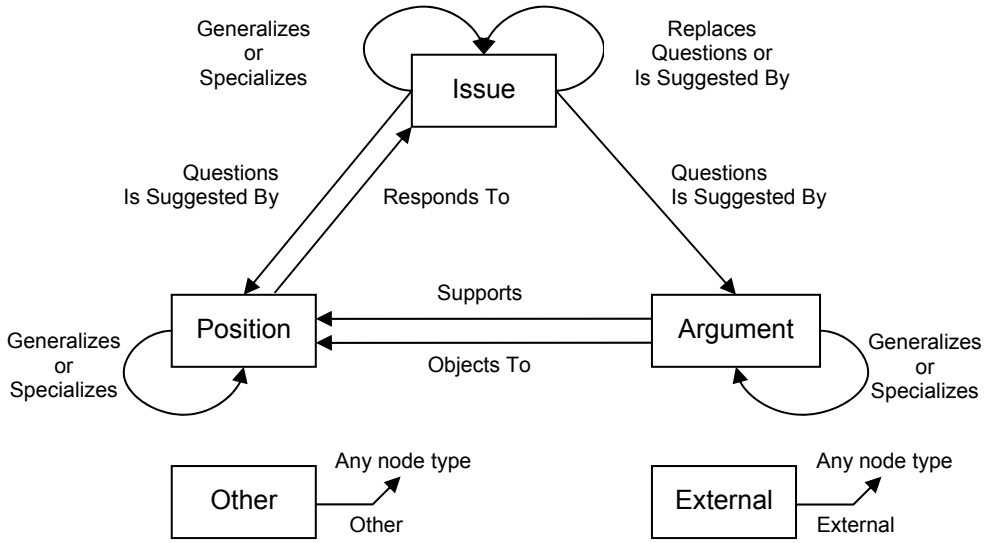

Abbildung 2: Das gIBIS Schema (Quelle: in Anlehnung an [LoLo00, S. 203])

Diese Elemente werden durch Beziehungen miteinander verbunden, die in Abbildung 2 dargestellt sind. Zusätzlich zu den genannten Elementen wurden bei der graphical IBIS (gIBIS) [CoBe88] Variante von IBIS die Entitäten *External* und *Other* hinzugefügt. *External*

repräsentiert eine Referenz auf ein externes Artefakt und *Other* einen Container für nicht von den anderen Entitäten Abgedecktes.
Das Ziel von IBIS – nicht als Schema, sondern als Ansatz, der das IBIS Schema zur Repräsentation des Rationale nutzt – ist vor allem die Unterstützung und Verbesserung von Beratungssitzungen mit mehreren Beteiligten. Instanzen des IBIS Schema sollen hierbei die Beratungen festhalten und eine Erfassung von verschiedenen Standpunkten ermöglichen. Hierzu wird eine so genannte *root-issue* gestellt, von der aus dann die Sammlung von Positionen und zugehörigen Argumenten beginnt. So entsteht eine Baumstruktur, aus deren Positionen und Argumenten weitere Fragestellungen entstehen können. Durch das Aufkommen von ganz neuen Fragen entsteht ein IBIS „Wald“. Am Ende einer Sitzung wird dann eine Einigung angestrebt, die die geführte Diskussion und deren niedergelegte Argumentation als Grundlage für die Entscheidung heranzieht.
Sollte man IBIS gemäß o.g. Charakteristika für Rationale Ansätze klassifizieren, so besitzt der Ansatz ein einfaches semi-formales Schema als Repräsentationsform und müsste nach Dutoit et al. [DMMP06b, S. 8] als vordergründig präskriptiv und aufdringlich gelten.[7] Präskriptiv ist er in der Hinsicht, dass er versucht,

1. durch das Aufzeigen verschiedener Sichtweisen (Positionen) auf die Fragestellung und
2. durch die Sammlung von Argumenten für diese Positionen

die Entscheidungsfindung der Interessenvertreter zu steuern und zu beeinflussen.
Aufdringlich ist er deshalb, weil die Erfassung, Strukturierung und Präsentation der einzelnen Schemaelemente auf den eigentlichen Ablauf einer Diskussion in einer Sitzung sehr stark einwirkt.
Es sei angemerkt, dass das IBIS Schema auch in weiteren Rationale Ansätzen Verwendung findet, in denen diese Bewertung durchaus anders aussehen kann.

2.1.4.2 Questions, Options, and Criteria (QOC) und design space analysis

Questions, Options, and Criteria (QOC) [MYBM91] bezeichnet ein Rationale Schema mit vier Elementen, welche mit ihren Beziehungen auch in Abbildung 3 dargestellt sind:

Questions: *Questions* repräsentieren Probleme, die gelöst werden sollen.
Options: *Options* sind die Alternativen, die als Lösung für das Problem vorgeschlagen werden.
Criteria: *Criteria* dienen der Evaluation der Optionen und bestimmen die Problemlösung (gewählte Option). Als Kriterien, die bei einer Entscheidung berücksichtigt werden müssen, können z.B.

- Anforderungen an Artefakte,
- Ziele für ein Produkt (z.B. die einzelnen Elemente der ISO 9126 [ISO9126]),
- Ziele des Projektmanagements (z.B. Zeit- oder Budgeteinhaltung) oder
- Kontexteinflüsse (z.B. momentane technische Gegebenheiten, legale oder firmeninterne Vorgaben)

eine Rolle spielen.
Arguments: *Arguments* dienen der Argumentation für und wider den weiteren Elementen des Schemas. Sie können also mit jedem der vorher genannten Elemente in Beziehung gesetzt werden.

[7] Klassifiziert man Rationale Ansätze nach o.g. Charakteristiken, so müssen zwei Dinge unterschieden werden. Das Schema (Metamodell) an sich und der Ansatz (vgl. Kapitel 2.1.3), der das Schema nutzt und Aussagen zur Prozessimplementierung des Rationale Management macht. Ein Schema an sich kann erst einmal nicht hinsichtlich der Charakteristika klassifiziert werden, ein Ansatz sehr wohl. Dies ist vor allem bei Ansätzen/Schemata zu beachten, bei denen Ansatz und Schema dieselbe Bezeichnung besitzen.

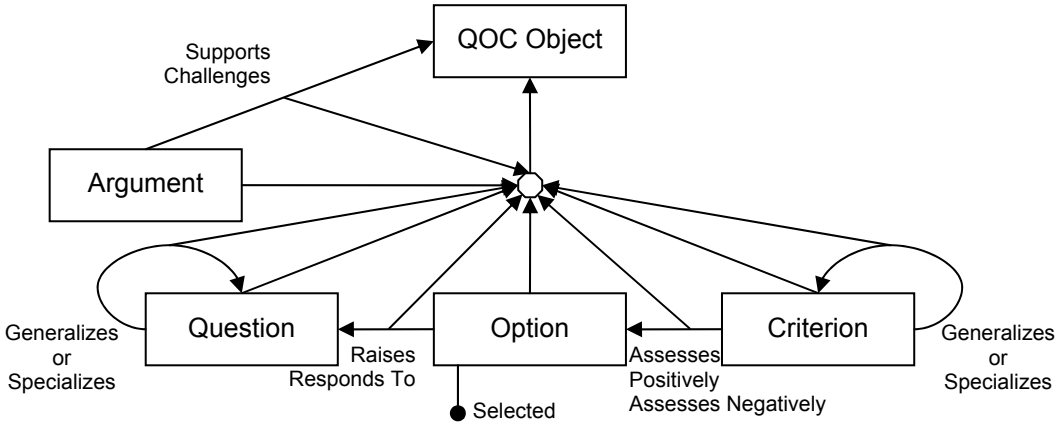

Abbildung 3: Das QOC Metamodell (Quelle: in Anlehnung an [LoLo00, S. 207])

QOC als ein Rationale Ansatz (**design space analysis**)[8] zielt darauf ab, den Gestaltungsraum (**design space**) über die Frage und deren Lösungsoptionen explizit zu machen (*design space focus*) und dadurch dessen Analyse zu ermöglichen (*design space analysis*). Damit will er auch die Kommunikation unter den Beteiligten fördern und Kreativität wecken (finden von Optionen). Sind die Optionen klar, werden sie gegen Kriterien abgewogen. Es wird folglich festgestellt, inwiefern die Kriterien die Optionen unterstützen beziehungsweise hemmen. Die Kriterien sind ein zentraler Punkt bei QOC (*focus on criteria*). Sie sollen die Ziele, die letztlich mit einer Entscheidung erreicht werden, in den Fokus rücken. Dabei kann ein Kriterium negative oder positive Auswirkungen in verschiedenen graduellen Abstufungen auf eine Option haben. Über die explizite Evaluation soll der Anwender von QOC darin unterstützt werden, die beste Alternative für die Problemlösung unter Berücksichtigung der untersuchten Kriterien zu wählen. Über Argumente kann jedes Element von QOC entweder in Frage gestellt oder unterstützt werden. Damit lassen sich die anderen Elemente anreichern, was eine argumentative Diskussion ermöglicht (*argument based*). Mit am wichtigsten sind die Argumente hinter den Evaluationen, da erst diese die Evaluation nachvollziehbar machen.
Ein ausführliches Beispiel für die Anwendung von QOC findet man bei MacLean et al. [MYBM91], der ursprünglichen Veröffentlichung zu QOC, sowie in Kapitel 4.4.1 dieser Arbeit.
Der QOC Ansatz ist auf der einen Seite deskriptiv. Er will die Überlegungen und Schlussfolgerungen der Entscheidungsträger explizit darlegen, um später mit dieser Entscheidung konfrontierte Interessenvertreter darüber zu informieren, z.B. falls eine Änderung der Entscheidung ansteht. Der Fokus auf die *design space analysis* ist auf der anderen Seite präskriptiv ausgerichtet. Und geht man davon aus, dass man für bestimmte wiederkehrende Entscheidungen im Entwicklungsprozess eine Mengen von Kriterien vorgibt, die beim Fällen einer Entscheidung zu beachten sind, so wird dieser präskriptive Charakter noch weiter ausgebaut. Weiterhin ist QOC aufdringlich, da eine QOC Aufzeichnung immer zusätzlich zu einer Entscheidung und dem eigentlichen Vorgehen erstellt werden muss (*rationale as coproduct*).

2.1.4.3 Decision Representation Language (DRL)

Die Decision Representation Language (DRL) [LeLa91] entstand aus der Intention, ein ausdrucksstärkeres Modell zur Verfügung zu stellen, um generisch eine Vielzahl von Entscheidungen repräsentieren zu können (*decision rationale*). Das Resultat ist ein

[8] MacLean et al. [MYBM91, S. 53] unterscheiden in ihren Ausführungen zu QOC das Schema (QOC) und den Ansatz (design space analysis) auch begrifflich.

umfangreiches Modell, das zur groß angelegten Anwendung Werkzeugunterstützung benötigt, die ebenfalls von den Autoren entwickelt wurde (SYBIL) [Lee90].
Zur Erfassung von Entscheidungen sind in dem Modell folgende Basiselemente vorhanden:

Decision Problem: Dieses Element repräsentiert das Problem, welches eine Lösung verlangt.
Alternative: Eine Alternative stellt eine Lösungsmöglichkeit dar, die als Lösung für das *Decision Problem* in Betracht gezogen wird.
Goal: Ein Ziel repräsentiert eine gewünschte Eigenschaft des Zustandes nach der Entscheidung und wird verwendet, um die Optionen zu evaluieren. Die Autoren sprechen auch von „umformulierten" Kriterien.
Claim: *Claims* repräsentieren Behauptungen und sind in DRL das Mittel zur Argumentation.

In Abbildung 4 sind noch einige zusätzliche Modellelemente dargestellt, auf die hier nicht weiter eingegangen wird. Es seien aber noch zwei Dinge zu dem Modell erwähnt:

- Die in der Abbildung eingezeichneten Striche stellen „*is-a*" Beziehungen dar.
- Daraus ergibt sich, dass das „*Is Related to*" Element, welches in DRL Beziehungen zwischen Elementen darstellt, ebenfalls als Behauptung (*Claim*) angesehen wird.

Die Beziehungen haben verschiedene Bedeutungen (z.B. *Achieves, Supports, Denies* etc.) und die Elemente, die sie in Beziehung setzen, sind in der Abbildung in den Klammern dahinter vermerkt.

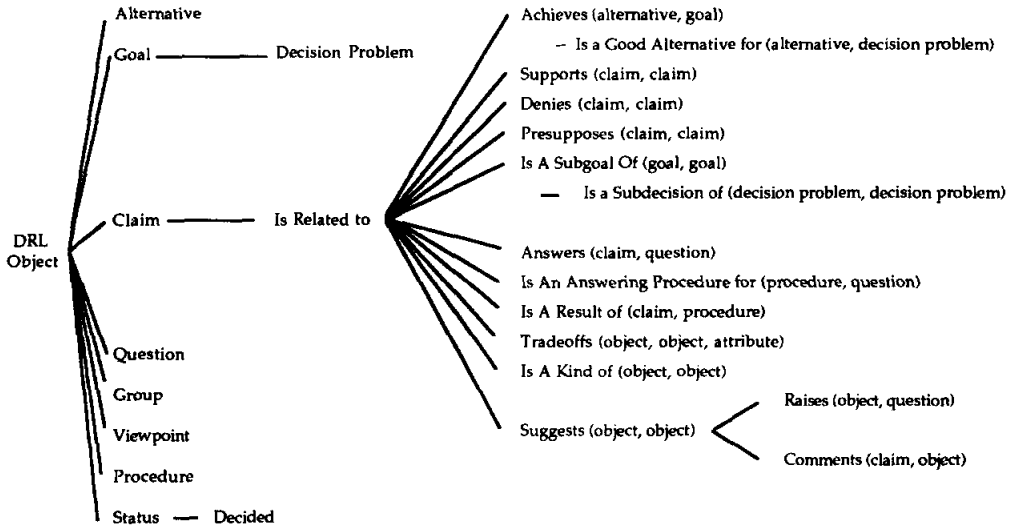

Abbildung 4: Die Modellelemente der DRL (Quelle: [LeLa91, S. 265])

Das Vorgehen eines Anwenders von DRL weist Ähnlichkeiten mit QOC auf. Begonnen wird mit der Erfassung des Entscheidungsproblems (*Decision Problem*), für das dann zu erreichende Ziele und mögliche Lösungsalternativen angegeben werden. Die Alternativen werden darauf gegen die Ziele abgewogen, was mittels der *Achieve*-Beziehung geschieht, die wiederum einen *Claim* darstellt. *Claims* können mit der Wahrscheinlichkeit, mit der sie gelten, (*plausibility*) und dem Ausmaß, mit dem die Behauptung gilt (*degree*), attributiert werden. So sagt z.B. das Ausmaß der *Achieve*-Beziehung aus, inwieweit die Alternative das Ziel erreicht (Gewichtung). Ein ausführliches Beispiel und die Erläuterung der weiteren Elemente von DRL geben Lee und Lai [LeLa91].
Der DRL Ansatz versucht durch ein besonders ausdrucksstarkes und damit auch umfangreiches Schema das Rationale einer Entscheidung umfassend und in erster Linie deskriptiv zu erfassen. Allerdings ist die Anwendung von DRL durch den Aufbau eines

Modells einer Entscheidung mit vielen Überlegungen und Diskussionen verbunden (explizite Ziele/Kriterien, generierte Alternativen, Beziehungssetzung und Evaluation). Findet diese Diskussion vor der Entscheidung statt, kann der Ansatz ebenfalls als präskriptiv bezeichnet werden. Durch die Komplexität des Modells und die Notwendigkeit der Werkzeugunterstützung ist der Ansatz als aufdringlich einzustufen.

2.1.4.4 Weitere Rationale Ansätze

In der Literatur finden sich noch viele weitere argumentations-basierte Ansätze, wie z.B.

- Representation and Maintenance of Process Knowledge (REMAP) [RaDh92; LoLo00, S. 204],
- Procedural hierarchy of issues (PHI) [McCa91] oder
- das „Potts and Bruns Metamodell“ [PoBr88] beziehungsweise dessen Weiterentwicklung das „Potts Metamodell“ [Pott89].

In den meisten Veröffentlichungen zum Thema Rationale werden von diesen Ansätzen die **Schemata** und grundlegenden Ideen **verwendet** und **adaptiert**, um weitere Rationale Ansätze zu entwickeln. Da man heute davon ausgeht, dass beim Einsatz von Rationale die **Werkzeugunterstützung** eine **bedeutende Rolle** spielt [DuPa01, S. 808; Lee97], um die Aufdringlichkeit von Rationale und dessen Verwaltungsaufwand zu reduzieren, finden sich diese Schemata auch in vielen unterschiedlichen Werkzeugen wieder, z.B. beim Compendium Werkzeug[9] [BSS+06] (IBIS Schema), bei SEURAT[10] [BuBr06] (Adaption des DRL Schemas) und bei Sysiphus[11] [WoDu04] (QOC Schema).
Zu den Rationale Ansätzen, die **kein Schema** zur Erfassung verwenden, gehört der von Schneider [Schn06] beschriebene Ansatz, der mittels des FOCUS Werkzeugs Rationale erfasst. Die Erfassung geschieht über Videoaufnahmen von Entwicklern während der Prototypentwicklung oder deren Diskussion mit anderen Entwicklern. Die Aufnahmen können dann z.B. mit verschiedenen Schlagwörtern versehen oder mit den Codeelementen verknüpft werden.

2.2 Nutzen und Durchführung von Rationale Management

2.2.1 Nutzen von Rationale Management

Der in der Literatur diskutierte Nutzen von Rationale ist vielfältig und steht in starker Wechselwirkung mit dem Kontext (vgl. Kapitel 2.3), in dem Rationale angewendet wird. Dutoit et al. [DMMP06b, S. 16ff.] geben eine Übersicht über die Vorteile, von denen hier einige kurz vorgestellt werden sollen:

Unterstützung der Zusammenarbeit *(supporting collaboration)*
Hierunter fallen verschiedene Aspekte, wie die Nutzung von Rationale, um

- die Arbeit anderer Teammitglieder zu verstehen und nachzuvollziehen,
- verschiedene Standpunkte (z.B. während Teamtreffen) offen zu legen (vgl. IBIS Ansatz)
- gemeinsam eine kritische Reflexion des Problems/der Entscheidung zu erreichen und
- einen Konsens zwischen den Teammitgliedern zu schaffen.

[9] http://www.compendiuminstitute.org/
[10] http://www.users.muohio.edu/burgeje/SEURAT/
[11] http://sysiphus.in.tum.de/

Qualitativ bessere Entscheidungen
Über die Offenlegung von Diskussionen (z.B. mit Zielen, Argumenten, Alternativen, Kriterien und Bewertungen) werden Entscheidungen stärker durchdacht und abgewogen (vgl. rationale Entscheidungen Kapitel 2.1.1), sind für mehrere Interessenvertreter transparent und damit leichter zu überprüfen. Dies kann zu einer Qualitätsverbesserung der Entscheidungen beitragen.

Unterstützung von Änderungen und Wiederverwendung
In der Softwareentwicklung besteht häufig die Notwendigkeit, Änderungen in einer Software vorzunehmen. Hier versprechen sich Befürworter von Rationale – vor allem von deskriptiven Ansätzen – die Erleichterung der Arbeit für Rollen, die die Änderungen vornehmen müssen. Denn man erhofft sich, dass die Rollen die Auswirkungen der Änderungen mit Hilfe von Rationale besser verstehen und abschätzen können. Gegebenenfalls werden durch das bessere Verständnis auch geplante Änderungen wieder zurückgezogen, da erkannt wird, dass die Änderung nicht den ursprünglich gewünschten Effekt beziehungsweise ungewollte „Nebenwirkungen" erzielen würde.
Ähnliches gilt für die Wiederverwendung von Artefakten, die ein gutes Verständnis derselben voraussetzt.

Unterstützung von Wissenstransfer
In diesem Zusammenhang kann die Erfassung von Rationale dazu dienen, aus der Vergangenheit zu lernen, indem man die Entscheidungen retrospektiv analysiert und daraus Schlüsse zieht, inwieweit Dinge beim nächsten Mal anders beziehungsweise gleich gemacht werden sollten.
Dieses Wissen kann man z.B. in Form von Mustern (*pattern*) [GHJV95, HHLP06, REPARE] wieder verwenden und anderen zur Verfügung stellen. Gleiches gilt für *Lessons Learned* Analysen, die durch Rationale unterstützt werden können.
Weiterhin kann die Erfassung von Rationale die Einführung neuer Teammitglieder unterstützen und einem Projekt Wissen (auch über die physische Präsenz von Teammitgliedern hinaus) zur Verfügung stellen.

2.2.2 Durchführung von Rationale Management

Die Durchführung von Rationale Management sieht je nach gewünschter Zielerreichung und **Anwendungskontext** verschieden aus. In einem Projekt, welches unter hohem Zeitdruck durchgeführt werden muss, wird die Erfassung von Rationale gegebenenfalls weniger stattfinden beziehungsweise erst versetzt zu einem Zeitpunkt mit weniger Druck. Bei einem Forschungsprojekt, welches im Nachhinein ausgiebig evaluiert werden und einen hohen Erkenntnisgewinn und Lernzuwachs für die Beteiligten erzielen soll, kann die Erhebung mit Sicherheit verstärkt erfolgen. Auch wenn die Umsetzung von Rationale Management unterschiedlich ausfällt, können generische Aufgaben des Rationale Management identifiziert und je nach **Bereitstellung von Ressourcen** für diese Aufgaben verschiedene („Reife-") **Stufen** bei der Durchführung von Rationale Management unterscheiden werden.

2.2.2.1 Aufgaben des Rationale Management

Dutoit und Paech [DuPa01, S. 793ff.] identifizieren die **generischen Aufgaben** des Rationale Management ausgehend von der Domäne des Wissensmanagements (*Knowledge Management*) und damit **als spezielle Aufgaben des Wissensmanagements**, da Rationale ebenfalls eine bestimmte Art des Wissens repräsentiert (vgl. Kapitel 2.3).
Grundsätzlich wird unterschieden zwischen strategischen und operativen Aufgaben, die auch in Abbildung 5 dargestellt sind:

Strategische Aufgaben:

Identifikation von Rationale Zielen *(Identifying rationale goals)*. Hier werden die Ziele festgelegt, die mit der Anwendung von Rationale Management in der Organisation erreicht werden sollen. Dafür kann vom allgemeinen Nutzen ausgegangen werden, welchen Rationale verspricht (vgl. Kapitel 2.2.1). Dieser kann dann – auf spezifischere Ziele in einem Anwendungsfeld herunter gebrochen – die konkret vorhandenen Probleme in der Organisation adressieren.

Rationale Messung *(Rationale measurement)*. Diese Aufgabe ist eine typische Qualitätssicherungsaufgabe für den Prozess und soll gewährleisten, dass Rationale Management auch den gewünschten Nutzen erbringt (z.B. über die Definition von Metriken etc.).

Operative Aufgaben:

Rationaleidentifikation *(Rationale identification)*. Diese Aufgabe identifiziert die Entscheidungen, für welche Rationale erfasst werden soll und bestimmt damit die Menge an Rationale, die letztendlich erfasst wird. Je nach Zieldefinition werden hier andere Entscheidungen in den Fokus der Identifikation rücken.

Rationaleerfassung *(Rationale acquisition)*. Diese Aufgabe beinhaltet die Aktivitäten, die nötig sind, Rationale offen zu legen, d.h. implizit vorhandenes Rationale explizit zu machen. Je nach Prozessimplementierung kann diese Aufgabe parallel beziehungsweise zeitnah mit anstehenden Entscheidungen oder aber rückwirkend geschehen.

Rationaleentwicklung *(Rationale development)*. Rationaleentwicklung ist die Aufgabe, das nun explizit vorhandene Rationale zu dokumentieren, indem es in eine geeignete Form gebracht und strukturiert wird. Dies geschieht z.B. in Form der o.g. Schemata. Dutoit und Paech [DuPa01, S. 797] bemerken, dass diese Aufgabe von der Rationaleerfassung oft nicht scharf abzugrenzen ist.

Rationaleverteilung *(Rationale distribution)*. Um Rationale nutzbar zu machen, muss es den verschiedenen Beteiligten zur Verfügung stehen und zugreifbar sein. Dies zu ermöglichen ist Aufgabe der Rationaleverteilung (z.B. über identifizierbare Entscheidungen und deren Rationale, Rationale welches mit dem Artefakt verknüpft ist, Filter-, Navigations- oder Suchfunktionen etc.).

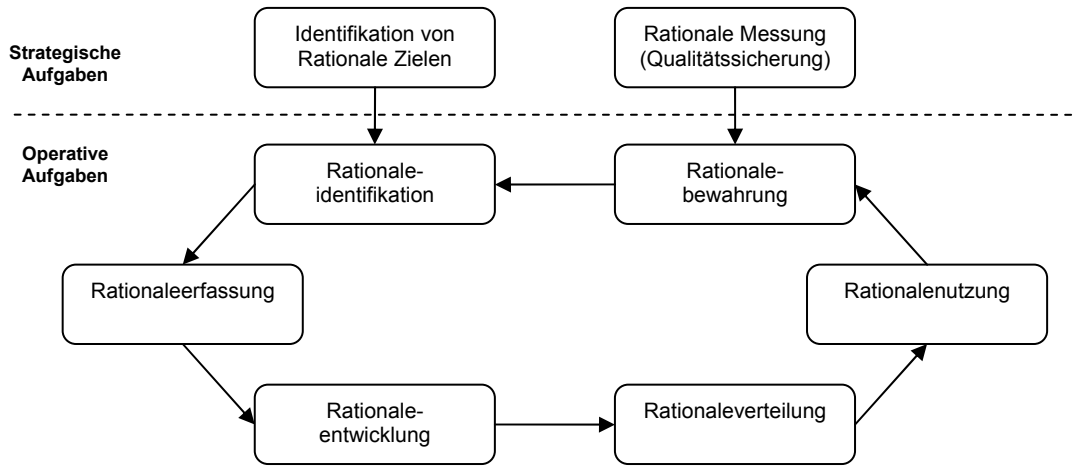

Abbildung 5: Aufgaben des Rationale Management (Quelle: modifiziert in Anlehnung an [DuPa01, S. 794])

Rationalenutzung *(Rationale use)*. Diese Aufgabe beinhaltet die Nutzung von Rationale, z.B. um bei einer Änderungsanfrage eine bessere Entscheidung zu treffen oder ein neues Teammitglied in ein System einzuführen und es ihm zu erklären.

Rationalebewahrung *(Rationale preservation)*. Diese Aufgabe umfasst die Pflege und Bewahrung von Rationaleaufzeichnungen. Rationale muss abgelegt und gleichzeitig auf einem aktuellen Stand gehalten werden. Dies muss so geschehen, dass der Zugriff auf das Rationale weiterhin effizient möglich ist, wodurch gegebenenfalls auch überfälliges Rationale gelöscht werden sollte. Hierfür wird in einigen Veröffentlichungen eine eigene Rolle vorgeschlagen, die die Pflege und Bewahrung des Rationale vornimmt (*Rationale Maintainer*).

2.2.2.2 Rollen des Rationale Management

Bei Vorstellung der Aufgabe „Rationalebewahrung" wurde bereits angemerkt, dass Dutoit und Paech [DuPa01, S. 797 beziehungsweise 805ff.] eine eigene Rolle (*Rationale Maintainer*) hierfür vorschlagen, deren Einsatz auch für weitere Aufgaben wie der „Rationaleerfassung" sinnvoll sei. Sie beschreiben die Aufgaben und Tätigkeiten dieser Rolle im Kontext des „Use Case Driven Software Development" ausführlicher in [DuPa01, S. 804ff.; DuPa00]. Auch weitere Autoren schlagen **spezialisierte Rollen** für einige Rationale Aufgaben vor. Buckingham et al. [BSS+06, S. 121] schreiben z.B. einem so genannten *Dialogue Mapper* zentrale Bedeutung zu. Dieser trägt der Tatsache Rechnung, dass es nicht einfach ist, aus Gesprächen beziehungsweise Diskussionen (*dialogue*), die oft nicht streng strukturiert geführt werden, „*on the fly*" Rationale zu extrahieren. Die Stellung der Rolle soll auch deren Ansehen und die Bedeutung von Rationale unterstreichen. Damit stellen sich die Autoren gegen die Delegation der wichtigen „Rationaleerfassung" an die so genannte *DR Scribe*-Rolle (*Design Rationale Scribe*), die in der Praxis meist auf einen einfachen Protokollant hinausläuft.[12]
Man sieht also, dass bei der Einführung von Rationale Management immer auch überlegt werden muss, ob es sinnvoll ist, eigene Rollen für gewisse Aufgaben vorzusehen. Unterstützt wird dies auch durch die Tatsache, dass man **Rationale Management als eine Variante des Wissensmanagement** betrachten kann und dort **ebenfalls spezialisierte Rollen** eingesetzt werden (z.B. *Experience Broker, Experience Communicator*) [RuLS01, S. 40]. Allerdings ist dies nicht immer notwendig und muss **anwendungsspezifisch** untersucht werden.

2.2.2.3 Stufen bei der Durchführung von Rationale Management

Die Durchführung von Rationale Management kann auf verschiedenen **(„Reife-") Stufen** geschehen. Diese werden nach dem Grad, in dem **Ressourcen** für die Aufgaben des Rationale Management zur Verfügung gestellt werden, unterschieden. Dutoit und Paech [DuPa01, S. 792] nennen vier Stufen:

Keine explizite Rationaleerfassung *(No explicit rationale capture)*. In dieser Stufe werden keine Ressourcen verwendet, um Rationale explizit zu erfassen oder dies zur Verfügung zu stellen. Dadurch befindet sich Rationale nur implizit in Dokumenten wie E-Mails, Protokollen oder Präsentationen.

Rekonstruktion von Rationale *(Rationale reconstruction)*. In dieser Stufe werden Ressourcen verwendet, um nach dem Abschluss eines Projektes die wichtigsten Ziele, Entscheidungen und Probleme zu dokumentieren. Zumeist werden dann z.B. Optionen und Argumente nicht gesammelt und niedergelegt.

[12] Es soll hier nicht der Eindruck erweckt werden, dass die Tätigkeit eines Protokollanten nicht von Bedeutung ist, sondern im Gegenteil einige Fähigkeiten abverlangt. In der Praxis wird dieser Aufgabe aber nicht immer genügend Bedeutung zugeschrieben.

Rationaleerfassung *(Rationale capture)*. In dieser Stufe werden Ressourcen zur Verfügung gestellt, um Rationale parallel mit den Entscheidungen zu erfassen und durch die Anwendung von Rationale Ansätzen auch die Entscheidungsfindung zu verbessern. Nach Abschluss des Projektes wird das Rationale strukturiert und komplettiert.

Rationaleintegration *(Rationale integration)*. Diese Stufe stellt die höchste Stufe dar, in der Rationale während des Projektes aufgezeichnet und aufbereitet wird, in den Arbeitsablauf integriert ist und das Rationale Management möglichst von Werkzeugen unterstützt wird. Hier sollte eine übergreifende Verknüpfung von Rationale und den Entscheidungsgegenständen und eine komfortable Nutzung des Rationale möglich sein.

2.2.2.4 Herausforderungen bei der Durchführung von Rationale Management

Auch wenn die Durchführung von Rationale Management in verschiedenen Bereichen des Software Engineering Vorteile verspricht und verschiedenste Ansätze und mittlerweile auch Werkzeuge für das Rationale Management zur Verfügung stehen (vgl. Kapitel 3), ist die Anwendung im industriellen Kontext nicht üblich. Die zögerliche Verbreitung in der Praxis kann man auf folgende vier zentrale Problembereiche zurückführen [HoAt06, S. 79ff.; DMMP06b, S. 20ff.]:

Kognitive Grenzen *(Cognitive Limitations)*. Menschen haben beschränkte kognitive Kapazitäten, um Informationen zu verarbeiten. Deshalb tendieren sie dazu, Probleme/ Entscheidungen nicht ausführlich zu überdenken und geben sich auch mit nahe liegenden – gegebenenfalls suboptimalen – Lösungen zufrieden. Die ausführliche Darlegung vieler/aller Ideen, Lösungsalternativen, Kriterien etc. führt zu höherer Komplexität und einer größeren Fülle von Informationen, deren Er- und Verarbeitung Zeit und zusätzliche kognitive Ressourcen benötigen.

Das Erfassungsproblem *(Capture Problem/ Capture Limitations)*.[13] Will man Rationale erfassen, ist man mit einer großen Anzahl von Entscheidungen und deren Begründungen konfrontiert. Es muss entschieden werden, was zu dokumentieren ist, bevor man überhaupt sicher sein kann, ob dieses Rationale erneut genutzt werden wird. Hinzu kommt der Aufwand beziehungsweise die notwendige Disziplin für die Erfassung und die Tatsache, dass der Zeitdruck bei den Tätigkeiten, in denen das meiste Rationale anfällt, am höchsten ist. Außerdem profitiert häufig die Rolle, die diesen Aufwand investiert, später nicht selbst vom dokumentierten Rationale. Weiterhin gibt es politische oder individuelle Gründe, sich nicht in die „Karten schauen zu lassen". Steht die Erfassung von Rationale fest, so ist es oft auch nicht einfach, eine Entscheidung oder geführte Diskussion in eine (semi-) formale Darstellung zu bringen, die von den meisten Rationale Ansätzen verlangt wird.

Das Abfrageproblem *(Retrieval Problem/ Retrieval Limitations)*. Dieses Problem beinhaltet die Herausforderungen in Verbindung mit dem Wiederfinden von Rationale. Denn wird Rationale erfasst, so steigt die Menge an Informationen stetig an. Hierfür muss eine geeignete Navigations-, Filter- oder Suchsystematik bereitstehen. Auch Fragestellungen,

- wie Rationale repräsentiert werden muss, um es effektiv abfragen (*retrieve*) zu können beziehungsweise
- wie die Notation für Rationale aussehen muss und
- in welchem Ausmaß (Kontext, Vollständigkeit) man Rationale erfassen muss, damit auch fremde Personen das Rationale verstehen und nachvollziehen können

[13] Schneider bezeichnet dieses Problem auch als das „Rationale Paradox" (The Rationale Paradox: „When most rationale is created, chances to capture it are lowest."), das es zu überwinden gilt [Schn06, S. 93].

müssen beantwortet werden. Dutoit et al. [DMMP06b, S. 20] merken hierzu an, dass bislang dieses Problem noch wenig untersucht wurde, da die Erfassung von Rationale noch nicht in großem Umfang stattfände und das Erfassungsproblem die noch größte Hürde darstelle.

Das Nutzungsproblem *(Usage Problem/ Usage Limitations)*. Dieses Problem besteht darin, dass es häufig Entscheidungen zu Fragestellungen gibt, die einmalig in der Art auftreten und deshalb nicht auf andere Probleme übertragen werden können. Diese erzeugen nur einen Mehraufwand, sowohl während der Erfassung, während der Pflege als auch während der Suche nach anwendbarem Rationale.

Jeder Rationale Ansatz muss auf diese Herausforderungen eigene Antworten finden, sei es durch weniger aufdringliche Ansätze, informale Erfassung des Rationale oder verstärkte Kommunikation zwischen den Projektbeteiligten. Misst man die Güte der Antworten auf obige Problemstellung an der Verbreitung der Ansätze in der Industrie, muss davon ausgegangen werden, dass die bisherigen Antworten noch nicht adäquat sind.

Zu diesen Problembereichen, die sich auf die operativen Aufgaben des Rationale Management beziehen, wird hier noch ein weiterer hinzugefügt, der die Einführung von Rationale Management in einer Organisation adressiert:

Die Einführungsproblematik. Soll Rationale in der Praxis angewendet werden, so muss dieses speziell auf die Ziele (vgl. Aufgabe „Identifikation von Rationale Zielen“, Kapitel 2.2.2.1) abgestimmt und in den gelebten Prozess einer Organisation eingepasst werden. Denn einfach nur einen vorhandenen Ansatz wie QOC vorzugeben, wird wenig Erfolg erzielen, ohne z.B. die Entscheidungen einzugrenzen, die zu dokumentieren sind und ohne sinnvolle Kriterien aufzustellen, wogegen die Optionen abgewogen werden sollen.
Weiterhin kann die Frage gestellt werden, inwieweit die Anwendung von Rationale in einem größeren Rahmen von der in der Organisation vorhandenen und gelebten Prozessreife abhängig ist. Denn ist das Prozessmanagement noch im Aufbau begriffen, stehen meist andere Prozesse noch im Vordergrund. Auch ist eine Integration von Rationale in Prozesse meist erst strukturiert möglich, wenn man sich klar ist, wie man vorgeht, was man eigentlich tut und was bei den zu treffenden Entscheidungen wichtig ist. Hierzu können ein reicher Erfahrungsschatz und etablierte Prozesse notwendig sein.

2.3 Wissens- und Entscheidungsbereiche im Software Engineering

Die genannten Konzepte und Rationale Ansätze sind zwar nicht auf die Durchführung von Rationale Management im Bereich des Produkt-/Systemwissens (vgl. Tabelle 2) innerhalb eines Projektes beschränkt, werden aber – teils aus historischen Gründen (vgl. Kapitel 2.1.2) – heute häufig dafür eingesetzt. Sie werden daher im Verlauf eines Entwicklungsprojektes meist zur Begründung von Entscheidungen bezüglich Systemartefakten (z.B. in der Anforderungsspezifikation, Architekturspezifikation, Testspezifikation etc.) verwendet.
Dieser Bereich stellt aber nur einen von vier Wissensbereichen (respektive Entscheidungsbereiche) im Software Engineering dar [DuPa01, S. 788ff.]. Wenn man also die Integration von Rationale Management in einer Softwareentwicklungsorganisation untersuchen möchte, müssen auch die anderen Wissensbereiche Beachtung finden. Diese werden nun in Verbindung mit deren Entscheidungsgegenständen und dem zugehörigen Rationale dargestellt. Bei Dutoit und Paech [DuPa01, S. 788ff.] findet man folgende Unterscheidung der Wissensbereiche innerhalb des Software Engineering:

Systemwissen *(System Knowledge)*. Dieser Wissensbereich umfasst das Wissen, welches mit dem zu entwickelnden System verbunden ist. Dieses Wissen findet sich für ein konkretes System z.B. in den Spezifikationsdokumenten unter den verschiedenen

Perspektiven auf das System (Anforderungsspezifikation, Architekturdefinition, Testfallspezifikation etc.).
Wird das Wissen zu einem System verallgemeinert, ist es nicht mehr nur für das zu entwickelnde System gültig, sondern kann auch für weitere Systeme wieder verwendet werden und fließt dort in die Entscheidungen mit ein. Ein Beispiel für solches Wissen sind die Entwurfsmuster (*design patterns*) [GHJV95].

Prozesswissen *(Process Knowledge)*. Wissen über Rollen, Aufgaben, Dokumente, Werkzeuge, Ressourcen und deren Anwendung, Ablauf und Verbindungen spiegeln das Prozesswissen wider. Prozesswissen regelt die tägliche Arbeit und die Verantwortlichkeiten der einzelnen Rollen.
Wie auch beim Systemwissen kann man das Prozesswissen eines konkreten Projektes verallgemeinern und in Erfahrungsberichte, Best Practices und Prozessmodelle aufnehmen, die dann über das konkrete Projekt hinaus Anwendung finden.

Wie man in den beiden Wissensbereichen gesehen hat, kann man jeweils nochmals eine Unterscheidung treffen zwischen Wissen, welches für das konkrete Projekt spezifisch ist und Wissen, welches über das konkrete Projekt hinaus anwendbar ist. Dies führt zusätzlich zu folgender Klassifizierung:

Produktwissen *(Product Knowledge)*. Dieser Wissensbereich enthält System- und Prozesswissen, welches bei der Entwicklung eines Produktes in einem Projekt gewonnen wird und Anwendung findet. Auf Systemebene sind dies vor allem die Anforderungen, Systemmodelle und Testfälle in den Spezifikationen und anderen Dokumenten, auf Prozessebene die Strategien, der Projektplan mit eingesetzten Ressourcen und weitere Rahmenbedingungen bei der Entwicklung des konkreten Produktes.

Organisationswissen *(Organizational Knowledge)*. Organisationswissen wird projektübergreifend eingesetzt. Auf Ebene der Organisation besteht das Systemwissen aus generalisierten Erkenntnissen und Erfahrungen, die sich z.B. in Architekturmustern, Blueprints, Domänenmodellen und (Produkt-) Standards widerspiegeln. Prozesswissen findet man generalisiert z.B. in Prozessmodellen, Best Practices, Checklisten und Templates.

		System Knowledge	**Process Knowledge**
Represents		System	Work Roles Resources
Product Knowledge	What	Specification System Design Object Design Source code Test plans and results	Project plan Budget Expenditures Policies & Procedures
	Why (rationale)	Justification behind the system, including development goals and criteria, alternatives evaluated, and their evaluation	Justification behind the task plan, including risk assessments, contingency plans, management goals & criteria
Organizational Knowledge	What	Domain models System architectures Design patterns	Process models Best practices Experiences
	Why (rationale)	Justification behind the generalized model, such as forces and trade-offs	Justification behind the generalized model, such as success factors associated with practices

Tabelle 2: Wissensbereiche im Software Engineering (Quelle: [DuPa01, S. 790])

In jedem der genannten Wissensbereiche kann Rationale Management angewendet werden. Während das Wissen in den jeweiligen Bereichen auf die Bereichsgegenstände wie System (Systemwissen) oder Arbeitsablauf (Prozesswissen) – entweder projekt-/produktbezogen (Produktwissen) oder verallgemeinert (Organisationswissen) – zielt, beschäftigt sich Rationale entsprechend mit den Entscheidungen und deren Elementen, die dieses Wissen erzeugen. In Tabelle 2 sind die generellen rationale-relevanten Fragestellungen zu den jeweiligen Wissensbereichen grau hinterlegt.
Man kann sich vorstellen, dass die jeweilige Repräsentation und Prozessimplementierung von Rationale – und daher der jeweils gewählte Rationale Ansatz – für die einzelnen Bereiche sehr unterschiedlich aussehen kann. Verschiedene Ansätze für die Wissensbereiche werden in Kapitel 3 vorgestellt.

2.4 Zusammenfassung

In diesem Kapitel wurde das grundlegende Verständnis für die weitere Behandlung des Themas Rationale Management geschaffen. Hierfür wurden die Kernkonzepte von Rationale vorgestellt: Rationale als Gründe hinter einer Entscheidung, deren einbezogene Lösungsalternativen, Kriterien und Argumentationen, die zur endgültigen Entscheidung geführt haben. Dies alles wird meist dokumentiert mit einem Schema zur Repräsentation des Rationale. Die Vorschläge für Schemata unterscheiden sich dabei in den einbezogenen Aspekten einer Entscheidung und ermöglichen dadurch eine unterschiedlich differenzierte Darstellung mit unterschiedlichen Schwerpunkten.
Die Anwendbarkeit der Vielzahl von diskutierten Ansätzen zeigt sich allerdings erst anhand konkreter Problemstellungen, für die sie eingesetzt werden sollen. Wie in Kapitel 3 noch sichtbar wird, bedienen sich die verschiedenen Rationale Ansätze in der praktischen Anwendungen häufig der in diesem Kapitel vorgestellten Schemata (IBIS, QOC, DRL etc.) und passen diese entsprechend ihren Bedürfnissen an, integrieren sie in die Prozesse und in geeignete Werkzeuge.
Neben diesen vorwiegend statischen Elementen des Rationale wurde ebenfalls die Integration von Rationale in den Arbeitsablauf während des Software Engineering betrachtet. Je nach Grad der zur Verfügung gestellten Ressourcen unterscheidet man dabei unterschiedliche („Reife-“) Stufen. Im Zusammenhang mit den Aufgaben und Rollen wurde auch die Verbindung von Rationale Management und Wissensmanagement ersichtlich. Denn Rationale kann man als zusätzliches Wissen betrachten, und somit sind dessen Behandlung und die damit verbundenen Aufgaben und Rollen denen des Wissensmanagements sehr ähnlich. Gezeigt wurde außerdem, dass den Vorteilen der Anwendung von Rationale Management ebenfalls größere Herausforderungen entgegenstehen, die bei dessen Einsatz adressiert werden müssen.
Anhand der zum Abschluss vorgestellten Wissensbereiche werden im folgenden Kapitel verschiedene Rationale Ansätze aus der Praxis vorgestellt, welche bestätigen, dass das Rationale Management in den jeweiligen Bereichen durchaus größere Unterschiede aufweist und andere Schwerpunkte setzt.

Kapitel 3

Rationale Ansätze in der Anwendung

In Kapitel 2 wurden die grundlegenden Ideen des Rationale Management beziehungsweise der Rationale Ansätze beleuchtet und der intendierte Nutzen und die Ziele sowie der Durchführungskontext in Form von generischen Aufgaben, Rollen und Herausforderungen dargestellt. Weiterhin hat das Kapitel den Bezug von Rationale zu den verschiedenen Wissens- und Entscheidungsbereichen des Software Engineering aufgezeigt.
Nun werden, aufbauend auf diesen Grundlagen, verschiedene in der Forschung diskutierte Rationale Ansätze vorgestellt. Die Vorstellung der einzelnen Ansätze orientiert sich an der Struktur der Veröffentlichung [DuPa01], in welcher von den in Kapitel 2.2.2.1 identifizierten (operativen) Aufgaben für das Rationale Management ausgegangen wird.[14] Diese Aufgaben werden im Folgenden jeweils unter dem Oberbegriff Prozessimplementierung zusammengefasst. Ergänzend zu diesen Erläuterungen der Prozessimplementierung werden die grundlegende Zielsetzung und die Wahl der Repräsentationsform des Rationale für die Rationaleerfassung und -entwicklung vorangestellt. Mit der Vorstellung der Ansätze anhand der Repräsentationsform und der Prozessimplementierung werden die Hauptaspekte angesprochen, zu denen ein Rationale Ansatz Aussagen macht (Kapitel 2.1.3). Werden in diesem Kapitel unter der Prozessimplementierung keine Angaben zu einzelnen Aufgabenbereichen gemacht, so finden sich für die Ansätze hierzu keine konkreten Aussagen.
Im ersten Abschnitt des Kapitels werden Rationale Ansätze vorgestellt, die zur Unterstützung von Zusammenarbeit und Strukturierung von Treffen/Diskussionen Verwendung finden und deshalb grundsätzlich unabhängig von den Wissensbereichen sind (Kapitel 3.1). Der zweite Abschnitt umfasst Rationale Ansätze aus dem Wissensbereich Produktwissen, dessen Rationale in [Paec06] als Projekt- und Produktrationale (*project and product rationale*, PPR) bezeichnet wird (Kapitel 3.2). Der dritte Abschnitt beschäftigt sich dann mit Rationale Ansätzen des Wissensbereichs Organisationswissen, dessen entsprechendes Rationale in [Paec06] *rationale for organizing bodies of knowledge* (OBR) genannt wird (Kapitel 3.3). Abschließend findet eine Zusammenfassung statt (Kapitel 3.4).

[14] Die Anlehnung an diese Vorstellungsstruktur ermöglicht es, dass bei Interesse weitere vier Rationale Ansätze in der selben Darstellung nachgeschlagen werden können. Die Ansätze in dieser Veröffentlichung adressieren die Unterstützung von Verhandlungen (basierend auf der Theorie W und Win-Win), die Unterstützung der Anforderungserhebung, die Strukturierung und Erfassung von Treffen und die Vorhersage von Änderungsauswirkungen.

3.1 Wissensbereichübergreifendes Rationale – Erfassung von Rationale aus Diskussionen und Treffen

Um Rationale aus Diskussionen und Treffen heraus zu erfassen, existieren mehrere Ansätze. Aufgrund der ursprünglichen Ausrichtung von Rationale auf das Software Design (Kapitel 2.1.2) wird in einigen Publikationen explizit von Design-Treffen gesprochen. Daher beziehen sich auch die meisten Beispiele und Fallstudien in diesen Veröffentlichungen auf diese Treffen. Die Ansätze sind aber so allgemein gehalten, dass sie nicht auf Design-Treffen beschränkt sind und auch darüber hinaus angewendet werden können. Deshalb werden sie auch nicht Kapitel 3.2 „Wissensbereich Produktwissen – project and product rationale (PPR)“ zugeordnet, sondern als wissensbereichsübergreifende Ansätze vorgestellt.
Insbesondere der Compendium Ansatz macht die Anwendbarkeit über das Software Design und die Software Engineering Domäne hinaus deutlich. Dieser Ansatz und das zugehörige Werkzeug finden z.B. auch in Treffen und Workshops zu verschiedenen Themen bei der NASA, zur Konfliktlösung in Organisationen und Gemeinschaften sowie zur Visualisierung und Strukturierung von Verhandlungen und Beratungen über politische Strategien und Taktiken Anwendung.[15]
Im Folgenden werden zwei Ansätze zur Erfassung von Rationale aus Diskussionen und Treffen aufgezeigt: der genannte Compendium Ansatz und der Reasoning Loop Ansatz.

3.1.1 Der Compendium Ansatz

Der **Compendium Ansatz** [BSS+06] ist aus den ursprünglichen Ansätzen wie IBIS und QOC und damit aus dem klassischen argumentations-basierten Rationalegedanken (Kapitel 2.1.4) hervorgegangen. Der Ansatz wird von einer Organisation vorangetrieben (Compendium Institute[16]), die u.a. die (Weiter-) Entwicklung koordiniert, Schulungen anbietet und die (Interessens-) Gemeinschaft (*community*) bündelt und unterstützt. Laut dieses Institutes besteht der Compendium Ansatz aus verschiedenen Dimensionen,[17] z.B.:

1. Compendium als **Methode**, um durch strukturierte, visuelle Repräsentation eines Problems zu einer Lösung desselben zu kommen.
2. Compendium als eine **Technologie**, repräsentiert durch das Compendium Werkzeug, welches verschiedenste Schnittstellen und Im-/Exportfunktionen (XML, HTML etc.) zur Verfügung stellt, um es in Werkzeuglandschaften zu integrieren [BSS+06, S. 118].
3. Compendium als eine (Interessens-) **Gemeinschaft** (*community*) von Benutzern, Entwicklern, Forschern.
4. Compendium als ein **Forschungsgegenstand** für die Bereiche Rationale Management Knowledge Management, computerunterstützte Zusammenarbeit (*Computer Supported Cooperative Work*), Systeme zur Unterstützung kollektiver Entscheidungen (*Group Decision Support Systems*) etc.

Das Werkzeug stellt dabei einen zentralen Bestandteil des Ansatzes dar. Es besteht aus einer Benutzungsoberfläche (ähnlich eines Mind-Mapping Werkzeugs) zur Erfassung der Diskussionen und Entscheidungen, basierend auf den Elementen der Repräsentationsform (Kapitel 3.1.1.2). Weiterhin bietet es Strukturierungs- und Filterungshilfen, eine Navigations- und Suchfunktionalität sowie eine Benutzerverwaltung, um nur einige Funktionen zu nennen. Die gesammelten Daten werden entweder in einer integrierten oder einer beliebig bestimm- und konfigurierbaren externen relationalen Datenbank abgelegt. Um einen Eindruck des

[15] http://www.compendiuminstitute.org/community/showcase.htm am: 13.03.2007
[16] http://www.compendiuminstitute.org/
[17] http://www.compendiuminstitute.org/library/briefings.htm am: 13.03.2007

Compendium Werkzeugs zu erhalten, ist es in Abbildung 6 dargestellt. In dieser Abbildung kann man einige der genannten Funktionalitäten erkennen.

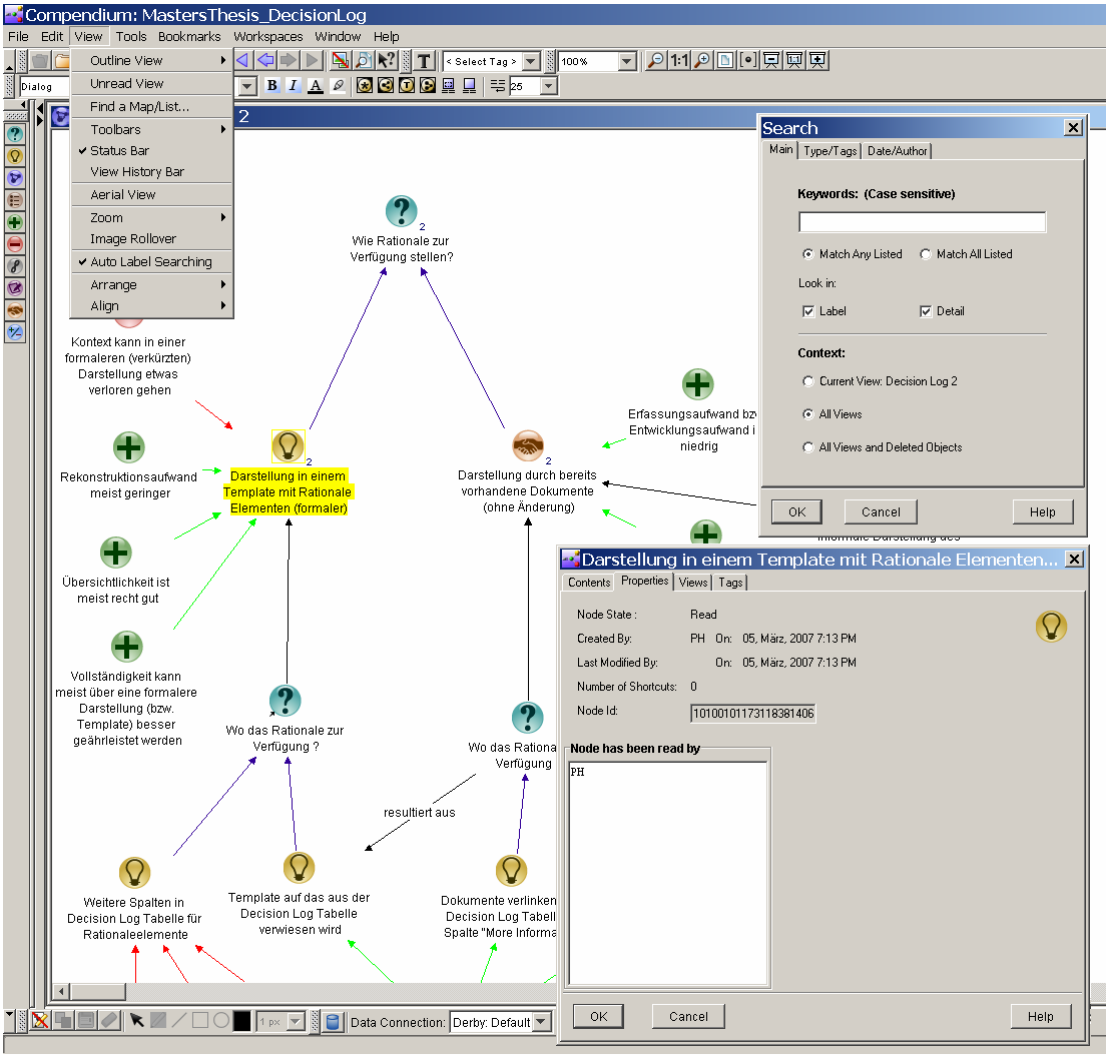

Abbildung 6: Benutzungsoberfläche des Compendium Werkzeugs

3.1.1.1 Ziele des Ansatzes

Der Compendium Ansatz adressiert die Unterstützung der Zusammenarbeit im Hinblick auf die Entscheidungsfindung. Dies geschieht durch Erfassung (und Modellierung) von Diskussionen in kollaborativen Arbeitsumgebungen durch so genanntes Dialogue Mapping. Unter **Dialogue Mapping** versteht man das Abbilden von Diskussionen (mit aufgegriffenen Ideen, Argumenten etc.) in eine graphische (IBIS-) Struktur, um die Ideen und Probleme analysieren zu können und verschiedene Sichtweisen der Interessenvertreter (Stakeholder) offen zu legen. Intention ist es, hierdurch auch das Rationale für die Probleme und die abgeleiteten Entscheidungen mit zu erfassen [BSS+06, S. 120].
Mit dem Ansatz wird außerdem versucht, über die Werkzeugbereitstellung und die Befähigung zum Dialogue Mapping durch Schulungen, die Aufdringlichkeit (*intrusiveness*) als größte Hürde für die Anwendung von Rationale Management zu reduzieren [BSS+06, S. 111].

3.1.1.2 Repräsentation des Rationale

Wie bereits erwähnt, basiert der Compendium Ansatz auf dem argumentations-basierten Rationalegedanken und genauer auf dem IBIS Schema (Kapitel 2.1.4.1). Das Schema wird

etwas abgewandelt, immer aber mit dem Ziel, das Repräsentationsschema so einfach wie möglich zu gestalten, um eine parallele Erfassung zur Diskussion zu vereinfachen. Ergänzte Elemente sind *Decisions* (gewählte Ideen beziehungsweise Antworten auf *Issues*), *Notes* (Anmerkungen zu den anderen Elementen) und *References* (Verweis auf eine Internetadresse oder ein anderes Dokument).
Ein Beispiel für die Erfassung einer Diskussion mit dem Compendium Werkzeug stellt Abbildung 21 (S. 86) dar, die eine Argumentation in Bezug auf eine Entscheidung in Kapitel 4.4.1.2 enthält.[18]

3.1.1.3 Prozessimplementierung des Ansatzes

a) Rationaleidentifikation

Die Fragestellungen und Probleme, die während der Diskussionen mittels des Compendium Ansatzes erfasst werden sollen, müssen von den Teilnehmern beziehungsweise dem Anwendungsgebiet festgelegt werden. Eine Agenda für ein Treffen gibt z.B. den Rahmen vor, welche Probleme zur Rationaleerfassung in Frage kommen. Innerhalb der diskutierten Probleme macht der Compendium Ansatz allerdings keine Vorgaben, zu welchen Problemen das Rationale tatsächlich erfasst werden soll. Er bietet also vor allem das Mittel, um die Aufgaben des Rationale Management durchzuführen, wenn die zu erfassenden Probleme und Fragestellungen identifiziert wurden.
Ist allerdings klar, dass der Compendium Ansatz in einen (Entwicklungs-) Prozess integriert wird und wiederkehrende Probleme damit erfasst werden sollen, so bietet das Werkzeug die Möglichkeit, so genannte *issue templates* zur Verfügung zu stellen [BSS+06, S. 119]. Diese sollen die wichtigsten Aspekte einer bekannten Situation enthalten und damit dem Anwender des Compendium Werkzeugs behilflich sein, alle Aspekte bei der Erfassung einer Diskussion beziehungsweise das Rationale in einer bestimmten Situation zu berücksichtigen. Buckingham Shum et al. nennen das Beispiel der Diskussion eines Geschäftsprozesses [BSS+06, S. 119]. Hierzu wird eine Vorlage (*template*) erstellt, welche die Aspekte vorgibt, die in diesem Zusammenhang zu diskutieren sind (z.B. Inputs, benötigte Ressourcen, Software, Infrastruktur, Outputs etc.). Damit kann dann indirekt auch die Rationaleidentifikation unterstützt werden, denn es wird vorgegeben, was erfasst werden soll.

b) Rationaleerfassung/Rationaleentwicklung

Jedes zu dokumentierende Problem wird als Frage mit seinen Lösungsmöglichkeiten externalisiert und von den Interessenvertretern über (Pro-/Contra-) Argumente angereichert. Weiterhin können Anmerkungen und Referenzen auf externe Ressourcen erfasst werden.
Die Rationaleentwicklung findet integriert mit der Erfassung statt, d.h. parallel zur Diskussion des Problems mittels des Compendium Werkzeugs. Damit soll der Zusatzaufwand reduziert und die Validierung des Rationale durch die gemeinsame Kontrolle aller anwesenden Interessenvertreter erreicht werden. Um den letzten Punkt zu gewährleisten, wird während der Treffen die erfasste Diskussion über das Werkzeug durch Bildschirme beziehungsweise Projektionen allen Interessenvertretern zur Verfügung gestellt. Zur effizienten Erfassung und Entwicklung schlagen Buckingham Shum et al. [BSS+06] auch eine spezialisierte Rolle, den *Rationale Maintainer* (Kapitel 2.2.2.2), vor, die hierfür geschult ist.

c) Rationaleverteilung

Die erfassten Diskussionen und das zugehörige Rationale sind über das Compendium Werkzeug auf vielfältige Weise zugänglich. U.a. über die Benutzungsschnittstelle selbst, aber auch exportiert als Bild, XML-Daten oder interaktive HTML-Seite. Darüber hinaus gibt es sehr viele unterschiedliche Möglichkeiten der Verteilung, da die Daten in einer relationalen Datenbank gespeichert werden. Hierdurch können sie in beliebige Formate exportiert und z.B. auch in anderen Werkzeugen verwendet werden. Innerhalb des Compendium Werkzeugs

[18] Diese Entscheidung wird dort auch genauer erläutert.

besteht für den Zugriff eine Navigations- und Suchfunktion sowie die Möglichkeit, verschiedene Sichten auf die Daten zu definieren.

d) Rationalenutzung

Das erfasste Rationale kann als Anreicherung der Protokolle von Diskussionen und Treffen angesehen werden und hilft, falls nochmals auf die Diskussionen zurückgegriffen werden muss. Es ermöglicht später das Nachvollziehen des Kontextes der Entscheidung, die Überprüfung, welche Aspekte bei einer Entscheidung berücksichtigt wurden, und kann dazu dienen, Interessenvertreter, die nicht an den Diskussionen beteiligt waren, in die Thematik einzuführen. Entscheidungen können somit im Nachhinein auch noch einer „Qualitätskontrolle“ unterzogen werden.
Trotz der Verwendung für diese Zwecke (deskriptive Intention) fokussiert der Ansatz aber vor allem auf die Strukturierung der Diskussion während der Treffen und die bessere Entscheidungsfindung über deren Externalisierung (präskriptive Intention).

3.1.1.4 Ähnliche Ansätze

Einen ähnlichen Ansatz wie den Compendium Ansatz stellt der Wisdom Ansatz dar [RoSP06]. Er basiert ebenfalls auf dem argumentations-basierten Rationalegedanken, nutzt das IBIS Schema und stellt ein entsprechendes Werkzeug zur Verfügung (Wisdom Werkzeug). Weiterhin wird ein Prozess vorgegeben, der durch Problemstrukturierungstechniken (Cognitive Mapping und Dialogue Mapping) die frühen Tätigkeiten des Requirements Engineering unterstützen soll. Das Ziel ist es, damit zu einer vollständigen und alle Interessenvertreteranforderungen entsprechend berücksichtigenden Problemdefinition zu kommen. Cognitive Mapping wird dabei zur Unterstützung der strategischen Entscheidungsfindung in der Domäne verwendet, indem wichtige (kausale) Zusammenhänge graphisch externalisiert werden. Ursache-Wirkung-Beziehungen, von denen die Interessenvertreter bislang implizit ausgegangen sind, werden also explizit diskutiert, niedergelegt und graphisch dargestellt. Dies geschieht zu Beginn des Prozesses und soll eine grobe Übersicht und Strukturierung bieten. Dialogue Mapping wird dann weiter zur Detaillierung der einzelnen Fragestellungen verwendet, um die Diskussion zwischen den Interessenvertretern mit Ideen und Alternativen sowie Argumenten für eine konkrete Problemlösung und damit das Rationale festzuhalten.
Auch die Autoren des Wisdom Ansatzes schlagen eine eigene Rolle zur Erfassung und Entwicklung des Rationale vor, deren Aufgabe es ist, die erstellten Karten (*maps*) mit den Interessenvertretern gemeinsam zu erarbeiten und abzugleichen.

3.1.2 Der Reasoning Loop Ansatz

3.1.2.1 Ziel des Ansatzes

Im Folgenden wird der Reasoning Loop Ansatz [LoLo00] vorgestellt, dessen vordergründiges Ziel eine Verbesserung des Software Designs im Sinne eines reflektierten Designs (*reflective design*) ist.[19] Er weist Ähnlichkeiten zu den vorangegangenen Ansätzen auf, da auch durch ihn implizite Überlegungen während der Diskussionen offen gelegt und somit zugänglich gemacht werden sollen. Dies wird als Basis zur reflektierten Entscheidungsfindung verwendet. Im Gegensatz zu den vorigen Ansätzen ist hierbei jedoch der Verzicht auf eine computerbasierte Werkzeugunterstützung ein weiteres Ziel. Die meist bei jeder Diskussion und jedem Treffen vorhandenen Medien Papier, Tafel, White Board etc. sollen ausreichen.
Neben den prozessbezogenen Zielen sollen bei der Repräsentation des Rationale beim Reasoning Loop Ansatz die (Gemeinsamkeiten der) Konzepte verschiedener ursprünglicher

[19] Auch dieser Ansatz ist unabhängig von dem Kontext des Software Designs in einem anderen Umfeld einsetzbar [LoLo00, S. 199].

Rationale Schemata (QOC, IBIS, DRL etc.) in ein „generisches Modell“ überführt und angewendet werden.

3.1.2.2 Repräsentation des Rationale

Entsprechend der Zielsetzung, ein „generisches Modell“ der argumentations-basierten Rationale Schemata zur Unterstützung der kollaborativen Problemlösung (reflektiertes Design) zu verwenden, werden von Louridas und Loucopoulos die gemeinsamen Konzepte der Rationale Ansätze in ein Modell überführt [LoLo00, S. 215ff.]. Dies resultiert in folgenden Elementen (Abbildung 7):

Goals: Sie repräsentieren die Ziele/Anforderungen, die erreicht und die Probleme/Fragestellungen, die gelöst werden sollen.
Hypotheses: Sie repräsentieren (Lösungs-) Vorschläge und Ideen für die Lösung der Probleme und die Erreichung der Ziele und Anforderungen.
Justifications: Sie repräsentieren die Argumentation zu den diskutierten Hypothesen (Pro/ Contra).
Design Actions: Sie repräsentieren die Aktivitäten zur Lösung der Probleme und Erreichung der Ziele/Anforderungen (z.B. Änderung eines Artefakts).

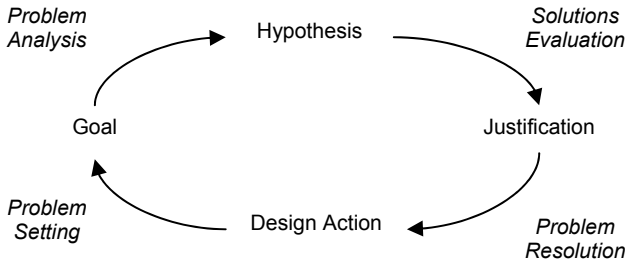

Abbildung 7: Der Reasoning Loop (Quelle: in Anlehnung an [LoLo00, S. 219])

Durch die Integration von Aktivitäten (*Problem setting*, *Problem analysis*, *Solutions Evaluation*, *Problem Resolution*) in das Modell, werden neben den statischen Aspekten auch die dynamischen reflektiert. Damit wollen die Autoren auch einen Rahmen für das Vorgehen beim reflektierten Design beschreiben. Die identifizierten Aktivitäten ähneln den in Kapitel 2.1.1 beschriebenen Aktivitäten zur Entscheidungsfindung von Herrmann und Paech [HePa06] und sind im Grunde auch implizit in den Ansätzen IBIS, PHI, QOC etc. enthalten, werden aber nun explizit in das Modell aufgenommen:

1. Zu lösende Probleme identifizieren (Problemidentifikation, *Problem setting*)
2. Analyse der Problemdomäne um (Lösungs-) Alternativen zu finden (Problemanalyse, *Problem analysis*)
3. Evaluation der vorgeschlagenen Alternativen (Lösungsevaluation, *Solutions evaluation*)
4. Wahl einer Alternative als Lösung (Problemlösung, *Problem resolution*)

Eine Entscheidung (Wahl einer Alternative als Lösung) zieht meist weitere Entscheidungen nach sich, wodurch sich diese Tätigkeiten wiederholen. Louridas und Loucopoulos nennen diese zyklische Abfolge der Tätigkeiten, die sich in deren Modell manifestiert, „Reasoning Loop“.

3.1.2.3 Prozessimplementierung des Ansatzes

a) Rationaleidentifikation

Bezüglich der Rationaleidentifikation ist auch dieser Ansatz so allgemein gehalten, dass er keine konkreten Vorgaben macht, welche Entscheidungen mittels des Reasoning Loops externalisiert werden sollen. Es gilt aber analog zu den vorgenannten Ansätzen, dass im Grunde eine Agenda eines Treffens die Probleme vorgibt, welche (durch den Reasoning Loop) analysiert werden sollten. Allerdings ist dies wenig spezifisch und wird vermutlich auch nicht für jedes Problem praktiziert.

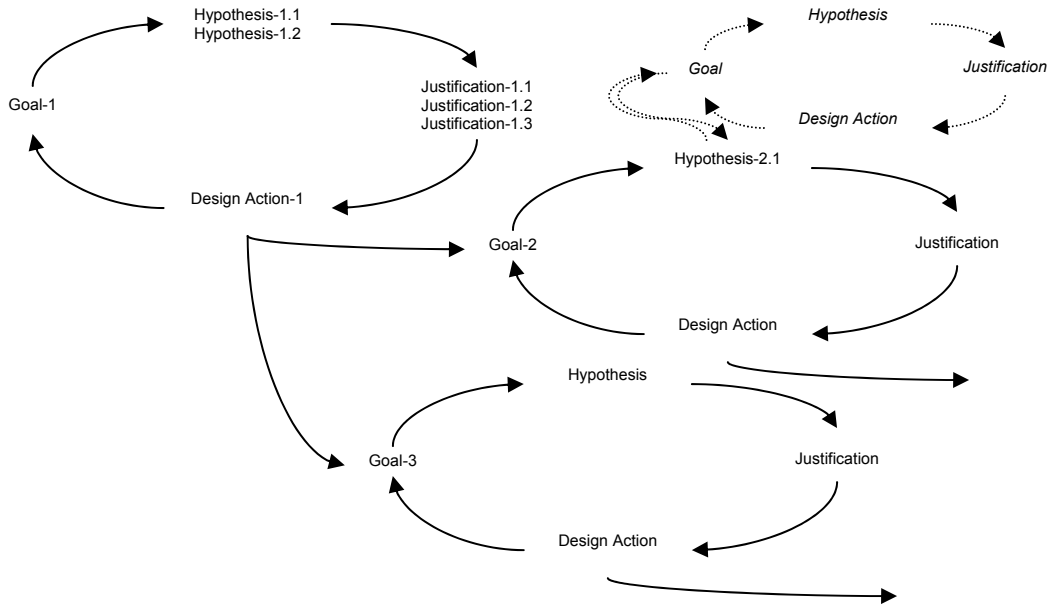

Abbildung 8: Beispiel des „reflektierten Designs“ mit dem Reasoning Loop (Quelle: in Anlehnung an [LoLo00, S. 220])

b) Rationaleerfassung/Rationaleentwicklung

Auch in diesem Ansatz findet die Rationaleerfassung und -entwicklung integriert und während des Treffens beziehungsweise der Diskussion des Problems statt. Dazu wird das in Kapitel „Repräsentation“ angegebene Schema verwendet. Die Erfassung und Entwicklung findet auf Papier, einer Tafel, Flip Chart, White Board oder ähnlichen Medien statt.

An dieser Stelle wird ein abstraktes Beispiel für die Anwendung des Ansatzes gegeben, welches aus der Veröffentlichung [LoLo00] übernommen wurde (Abbildung 8), in der sich auch noch weitere konkrete Beispiele finden:
In Abbildung 8 sind die Hypothesen 1.1 und 1.2 zwei Möglichkeiten, um das während der Problemidentifikation (*Problem setting*) aufgestellte Ziel Goal-1 (als Lösung des Problems) zu realisieren. Die Hypothesen selbst werden während der Problemanalyse (*Problem analysis*) identifiziert. In einem nächsten Schritt (*Solution evaluation*) evaluiert man die Hypothesen und hält die Argumente (Justification 1.1, 1.2 und 1.3) für und wider der Hypothesen fest. Basierend auf dieser Evaluation wird dann eine Umsetzung einer oder mehrerer Lösungsmöglichkeiten beschlossen (*Problem resolution*), welche meist in der Modifikation eines Artefaktes und/oder in der Aufstellung neuer Ziele resultiert (*Design Action*). In Abbildung 8 sind die neuen Ziele Goal-2 und Goal-3, die wieder den Ursprung eines weiteren Reasoning Loop ergeben. Hierdurch entsteht ein verschachteltes Netzwerk aus Reasoning Loops zu verschiedenen Zielen (Problemen), womit die Autoren den Design Prozess repräsentiert sehen [LoLo00, S. 221].

Zusätzlich merken die Autoren an, dass während des Design Prozesses beziehungsweise der Diskussionen nicht nur die Kerndiskussionen zu den Zielen und Problemen geführt werden, sondern auch so genannte Meta-Diskussionen (*meta-discussions*). Als Meta-Diskussionen bezeichnen sie Diskussionen, über die „Instanzen" der statischen Elemente des Reasoning Loop und nicht über das eigentliche Design Problem (*Goal*). Meta-Diskussionen enthalten somit z.B. Diskussionen und Argumentationen, inwieweit ein Hypothese wirklich eine Hypothese ist oder ob eine Begründung in dem betrachteten Kontext überhaupt relevant ist. Die Erfassung und Abbildung solcher Diskussionen ermöglicht der Ansatz, indem er neue Reasoning Loops, ausgehend von den Schemaelementen (außer dem Design Action Element), erlaubt. Diese werden als *secondary loops* bezeichnet. Ein *secondary loop*, der eine Meta-Diskussion über die Hypothese 2.1 enthält, ist in Abbildung 8 dargestellt (Reasoning Loop ausgehend von Hypothesis-2.1 mit gestrichelten Linien).
Damit wird das Netzwerk aber nochmals komplexer, auch wenn die Autoren vorschlagen, dass man sich bei der Betrachtung des Netzwerks auf die Kerndiskussion konzentrieren könne, indem man nur die Reasoning Loops verfolge, die von den Design Actions ausgehen (*primary loops*).

c) Rationaleverteilung/Rationalenutzung

Verteilung und Nutzung des erfassten Rationale werden von diesem Ansatz nicht adressiert. Dies wird verständlich, wenn man folgendes Zitat von Louridas und Loucopoulos betrachtet: „Our interests focus on using DR capture itself for improving the design process, and not in using its outcome, i.e., the rationale, for documentary and historical purposes" [LoLo00, S. 212]. Damit ist der Ansatz also vor allem auf die Anwendung in den Treffen und während den Diskussionen zur Erreichung der reflektierten Entscheidungsfindung (*reflective design*) beschränkt.

3.1.3 Vergleichende Zusammenstellung der Ansätze für den Bereich Rationale in Diskussionen und Treffen

Anhand der drei vorgestellten Ansätze für den Bereich „Wissensbereichübergreifendes Rationale – Erfassung von Rationale aus Diskussionen und Treffen" konnte man sehen, wie implizites Wissen bei kollaborativen Tätigkeiten externalisiert werden kann. Hierzu wurde in den ersten beiden Ansätzen zur Erfassung von Rationaleelementen das IBIS Schema verwendet. Dies resultiert mit aus der Tatsache, dass dieses Schema recht elementar ist und somit eine einfachere Erfassung parallel zu den Tätigkeiten stattfinden kann. Der Reasoning Loop Ansatz bedient sich dagegen mehreren Schemata, identifiziert Gemeinsamkeiten und stellt dann daraus abgeleitete Konzepte in seinem Modell auf.
Alle Ansätze streben allerdings danach, über die Externalisierung während der Diskussion die Entscheidungsfindung zu beeinflussen und zu steuern (präskriptiv) und sind aufgrund ihrer zentralen Stellung in den Treffen als aufdringlich (*intrusive*) anzusehen. Nichtsdestotrotz beansprucht der Compendium Ansatz durch die Einfachheit seines Schemas und die einfache Erfassung mittels des Werkzeugs, die Aufdringlichkeit möglichst stark zu reduzieren.
Eine zentrale Fragestellung ist bei einem Rationale Ansatz dessen Integrationsfähigkeit in bestehende Prozesse und Werkzeuge. Dies versucht vor allem der Compendium Ansatz durch das allgemeine Schema, benutzerspezifische Anpassungsmöglichkeiten (Erstellung von Vorlagen, Anpassung und Erweiterung von Modellobjekte etc.) und zahlreiche Schnittstellen sowie Im- und Exportmöglichkeiten der Daten zu gewährleisten. Die einfache Integration des Reasoning Loop Ansatzes, die Reduzierung der Aufdringlichkeit und die Flexibilität in verschiedenen Umgebungen soll der Verzicht auf eine computer-basierte Werkzeugunterstützung und die Möglichkeit des Einsatzes einfacher Medien (Papier, Tafel etc.) gestatten. Dadurch wird die praktische Anwendbarkeit des Ansatzes aber auch zu einer neuen Herausforderung (im Vergleich zu den anderen Ansätzen), da er viel Platz zum Aufschreiben braucht und nachträglich notwendige Strukturierungen und Positionierungen

der Elemente auf den genannten Medien (ohne Werkzeugunterstützung) häufig sehr schwer sind. Auch wenn die Idee für ein reflektiertes Vorgehen zu begrüßen ist, muss die Umsetzung in einem größeren Rahmen und bei komplexeren Problemen somit skeptisch betrachtet werden. Weiterhin deckt ein Rationale Ansatz, der keine Konzepte für die Rationaleverteilung und -nutzung zur Verfügung stellt und dadurch die retrospektive Betrachtung des Rationale weitgehend ausschließt, einen wesentlichen Teil der Idee und des Nutzens von Rationale Management nicht ab.
Zusammenfassend zielen alle Ansätze auf eine bessere Entscheidungsfindung durch die Analyse der Entscheidungen in deren Vorfeld. Inwieweit das erfasste Rationale weiterverwendet werden kann (über eine Protokollfunktion hinaus) und wie die ständig wachsende Informationsmenge verwaltet werden und effizient Nutzung erfahren kann, ist jedoch unklar und muss in den konkreten Anwendungsfeldern evaluiert werden.

3.2 Wissensbereich Produktwissen – project and product rationale (PPR)

Das Wissen bezüglich eines Produktes unterteilt sich in Wissen zu dem System selbst und in Wissen zu dem Projekt, in dem das System entstanden ist (vgl. Kapitel 2.3). Für den Bereich Prozess-/Produktwissen (Projektwissen) wird in diesem Kapitel kein Rationale Ansatz erläutert Allerdings wird für diesen Bereich in Kapitel 4.4.1 ein Ansatz vorgeschlagen, der Rationale im Kontext von Projektlenkungsentscheidungen zugänglich machen soll.
Für den Wissensbereich Produktwissen werden im Folgenden also ausschließlich Rationale Ansätze aus der Schnittmenge Produkt-/Systemwissen vorgestellt. In dieser Schnittmenge wird entsprechend der zentralen Tätigkeitsbereiche Requirements Engineering, Architekturdefinition/ Software Design und Implementierung/Wartung jeweils ein Ansatz gewählt. Mit dem in Kapitel 3.1.1.4 vorgestellten Ansatz Wisdom wurde bereits ein Ansatz genannt, der auf die Anforderungserhebung (*requirements elicitation*) im Zusammenspiel mit zahlreichen unterschiedlichen Interessenvertretern zielt.

3.2.1 Rationale im Kontext von Requirements Engineering

Der im Folgenden vorgestellte Ansatz [DuPa01] zeigt eine interessante Möglichkeit auf, werkzeuggestützt das Rationale direkt mit den während des Softwareentwicklungsprozesses erstellten Anforderungselementen (Artefakte, Taxonomieelemente) in Verbindung zu setzten. Rationaleelemente werden hier als „*first class entities*“ integriert in einem CASE-Werkzeug (*Computer Aided Software Engineering*) mit Namen „Sysiphus“ [20] erfasst, in dem auch die Anforderungselemente und das weitere Modell der Software (z.B. Anforderungen in Form von User Tasks, Use Cases, Services etc.) beschrieben wird.
Im Grunde ist dieser Ansatz nicht nur auf das Requirements Engineering beschränkt, auch wenn er in diesem Fall auf einen Use Case getriebenen Softwareentwicklungsprozess (*Use Case Driven Software Development*) ausgerichtet ist (vgl. z.B. [PaKo03; DuPa00; Haef05]). Das Konzept, das Rationale mit den Artefakten der Software innerhalb eines CASE-Werkzeugs zu verknüpfen, kann aber auf das gesamte Software Engineering und die in diesem Kontext entstehenden Artefakte angewendet werden, z.B. auch auf Modellelemente aus der Architekturdefinition oder den Testfallspezifikationen. Das CASE-Werkzeug Sysiphus wird dahingehend weiterentwickelt.

[20] http://sysiphus.informatik.tu-muenchen.de/index.html

3.2.1.1 Ziel des Ansatzes

Das Ziel des Ansatzes ist es, zu einem reflektierten Vorgehen während der Gestaltungsentscheidungen der die Software beschreibenden (Anforderungs-) Elemente beizutragen und dadurch die Gesamtqualität des Systems zu verbessern. Weiterhin sollen die unvermeidbaren Änderungsanfragen durch das erfasste Rationale besser beurteilt und entsprechend eingearbeitet werden können.

3.2.1.2 Repräsentation des Rationale

Zur Repräsentation des Rationale wird das QOC Schema verwendet (vgl. Kapitel 2.1.4.2). Die Elemente sind entsprechend in dem CASE-Werkzeug Sysiphus repräsentiert und werden in einer tabellarischen Form zur Verfügung gestellt. Ein Beispiel für das Aussehen einer solchen Repräsentation wird in Tabelle 3 gegeben.

Question: Wie Rationale zur Verfügung stellen?						
Criteria ► / **▼Options**	Erfassungsaufwand	Aufdringlichkeit	Verständlichkeit	Vollständigkeit/ Konsistenz	Übersichtlichkeit	Rekonstruktionsaufwand
Vorhandene, unveränderte Dokumente	+	+	+	-	k.A.	-
Rationale Template mit Rationale Elementen, welche genutzt werden, um darin das Rationale aufzunehmen	-	-	k.A.	+	+	k.A.

Legende:
+ : bzgl. des entspr. Kriteriums verhält sich die Option **positiv**
- : bzgl. des entspr. Kriteriums verhält sich die Option **negativ**
k.A. : bzgl. des entspr. Kriteriums findet keine Bewertung statt/ist keine Bewertung möglich

Tabelle 3: QOC Bewertung in tabellarischer Darstellung (Beispiel beschrieben in Kapitel 4.4.1.2)

3.2.1.3 Prozessimplementierung des Ansatzes

a) Rationaleidentifikation

Zur Identifikation von relevantem Rationale macht der Ansatz insoweit Vorgaben, dass im Grunde zu jedem Anforderungselement das zugehörige Rationale erfasst werden soll. Werden mehrere (alternative) Überlegungen zu einem Anforderungselement angestellt und Entscheidungen zu dessen Gestaltung getroffen, sollen all diese Überlegungen und Entscheidungen dem Element zugeordnet werden.

b) Rationaleerfassung

Die Rationaleerfassung ist in diesem Ansatz nicht immer verknüpft mit der Rationaleentwicklung. Vielmehr soll das Rationale bei den verschiedenen Tätigkeiten zur Anforderungserhebung und den Spezifikationstätigkeiten selbst festgehalten werden. Dies sind z.B. (Kunden-) Gespräche und Diskussionen zu einem Artefakt mit Software Designern und anderen Interessenvertretern oder auch Reviews (hiervon wird in [DuPa01, S. 806] unter der Bezeichnung *issue based rationale capture* ausgegangen).
Da die Repräsentation des Rationale mittels des QOC Schemas geschieht, liegt bei der Erfassung der Fokus auf der Überlegung, welche Optionen man zur Problemlösung hat und welche Kriterien für die Entscheidung eine Rolle spielen. Wenn Optionen generiert wurden, schlägt der Ansatz hier vor, als Kriterien für deren Evaluation die nicht-funktionalen Anforderungen (aus der Domäne) heranzuziehen. Weiterhin können als Evaluationskriterien z.B. Qualitätsattribute der ISO 9126 [ISO9126] oder auch andere Qualitätsmodelle herangezogen werden, wenn sich diese nicht ohnehin bereits in den nicht-funktionalen Anforderungen (*non-functional requirements*, NFR) für das Produkt widerspiegeln.

c) Rationaleentwicklung

Das erfasste Rationale muss dann in das QOC Schema des CASE-Werkzeugs überführt werden. Diese Tätigkeit wird vom so genannten *Rationale Maintainer* übernommen (vgl. Kapitel 2.2.2.2), der idealerweise eine spezialisierte Person für diese Aufgabe ist. Oft wird das Rationale aber auch von derjenigen Rolle eingepflegt, die für die Spezifikation des entsprechenden Anforderungselementes zuständig ist.
Wurde das Rationale während der Erfassung schon etwas strukturiert (Optionen, Kriterien etc.), gestaltet sich dessen Entwicklung nicht mehr ganz so schwer. Von einer Strukturierung kann aber nicht unbedingt ausgegangen werden. Also muss das Rationale spätestens zu diesem Zeitpunkt geeignete Struktur erhalten. Das CASE-Werkzeug bietet hierzu die Möglichkeit, die Frage, Kriterien (als NFRs) und Optionen zu erfassen.
In Abbildung 9 sind die Beziehung des Rationale zu den Anforderungselementen (*Requirement elements*) dargestellt. Der rechte Teil der Abbildung (Rationale) entspricht mit kleineren Modifikationen dem ursprünglichen QOC Schema (Kapitel 2.1.4.2).

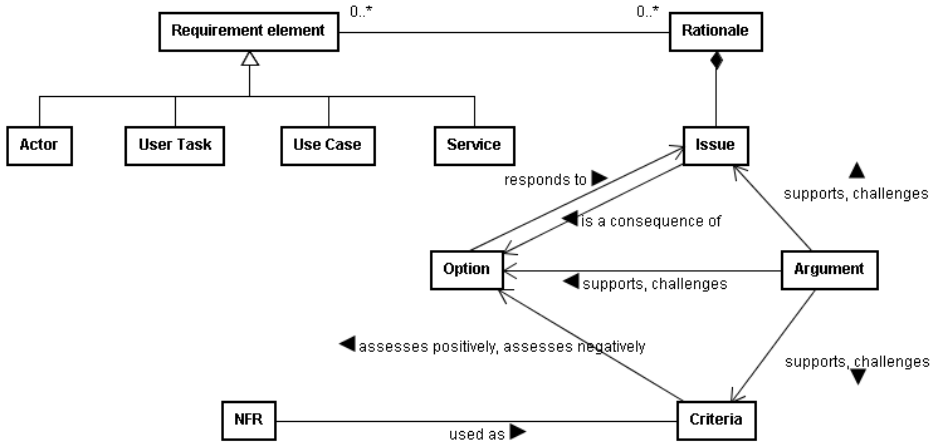

Abbildung 9: Schematische Darstellung der Beziehungen Anforderungs- und Rationaleelemente in dem Sysiphus Werkzeug (UML-Notation)

Nachdem das Rationale erfasst und entwickelt wurde, wird es wie in Tabelle 3 im Sysiphus Werkzeug dargestellt.[21] Die Verbindung und Darstellung von Anforderungselementen und Rationale im Sysiphus Werkzeug selbst wird in Abbildung 10 ersichtlich. Auf der linken Seite sieht man ein (stellvertretendes) Anforderungselement, das auch einen Verweis auf das zugehörige Rationale enthält (*Open Issues*). Von dort aus kann man z.B. zu dem verknüpften Rationale navigieren, welches dann auf der rechten Seite angezeigt wird. Eine weitere Möglichkeit ist die Selektion des Rationale, um von dort zu den betroffenen Anforderungen zu gelangen. Das in der Abbildung (rechts) dargestellte Rationale entspricht dem in Tabelle 3. Mit der Zweiteilung des Bildschirms mit Spezifikation (links) und zugehörigem Rationale (rechts) soll auch der Stellenwert des Rationale hervorgehoben werden.

d) Rationaleverteilung

Die Verteilung des Rationale findet mithilfe des Sysiphus Werkzeugs statt. Meist wird auf das Rationale über die Anforderungselemente zugegriffen, denen es zugeordnet ist. Mit diesen ist es über Verweise verbunden, über die innerhalb des Werkzeugs navigiert werden kann (vgl. auch Abbildung 10). Weiterhin bietet Sysiphus eine eigene Ansicht mit den Rationalefragestellungen mit einigen Filterfunktionen.

[21] Diese Abbildung ist eine tabellarische Art der Darstellung des Ausschnitts von Abbildung 18 zur Frage „Wie Rationale zur Verfügung stellen?“. Das Beispiel wird in Kapitel 4.4.1 ausführlich erläutert und soll an dieser Stelle nur einen Eindruck der Darstellungsform vermitteln.

Abbildung 10: Anforderungselement und zugehöriges Rationale in dem Werkzeug Sysiphus

e) Rationalenutzung

Wie zu Anfang erwähnt, soll der Nutzen des Ansatzes in einem reflektierten Vorgehen bei der Softwarespezifikation liegen, indem der Ansatz anhält, mehrere Gestaltungsoptionen in die Entscheidungen einzubeziehen und anhand von offen gelegten Kriterien die Optionen abzuwägen.
Weiterhin wird das Rationale nachträglich vor allem bei Bedarf zur Erklärung der Ausprägung von Anforderungselementen, zur Beurteilung von Änderungsanfragen und weiteren Verfeinerungen der Anforderungselemente verwendet. Die Beurteilung von Änderungsanfragen soll auch dadurch erleichtert werden, dass die verworfenen Optionen und die Gründe für deren Zurückweisung dokumentiert sind. Dadurch vermeidet man z.B. das (erneute) Einschlagen von Wegen, die sich als Sackgasse oder als ungeeignet erwiesen haben.

f) Rationaleerhaltung

In diesem Ansatz kommt der Erhaltung und Pflege des Rationale ein großer Stellenwert zu. Der *Rationale Maintainer* ist verantwortlich, das Rationale auf aktuellem und konsistentem Stand zu halten. Hierfür wird in [DuPa01, S. 807] eigens ein Pflegeprozess definiert (*rationale maintenance process*) und beschrieben.

3.2.2 Rationale im Kontext von Architekturentscheidungen

Der im Folgenden vorgestellte Ansatz dient der Erfassung von Architekturentscheidungen [TyAk05]. Er wurde entwickelt und angewendet im Kontext von IBM's e-Business Reference Architecture Framework [FlVi01], in dem Architekturentscheidungen eine entscheidende Rolle spielten. Es wird an dieser Stelle kein Beispiel für die Anwendung des Ansatzes gegeben; ein solches kann aber in [TyAk05] nachgeschlagen werden.

3.2.2.1 Ziel des Ansatzes

Die Dokumentation von Rationale im Zusammenhang mit Architekturentscheidungen soll diese transparenter und für alle Interessenvertreter verständlich machen. Weiterhin soll der Ansatz

- Änderungen leiten,
- Implikationen klar machen (z.B. Einflüsse auf die Organisation, Schulungsmaßnahmen etc.),
- das wiederholte Aufkommen von bereits gelösten Problemen und beantworteten Fragen reduzieren und
- eine Verfolgbarkeit herstellen zwischen den Zielen (z.B. Geschäftsanforderungen, Risiken, nicht-funktionale Anforderungen) und deren Umsetzung in den Architekturelementen, um sicherzustellen, dass die Architektur die Umsetzung der Ziele ermöglicht.

Architecture decision description template	
Issue	Describe the architectural design issue you're addressing, leaving no questions about why you're addressing this issue now. Following a minimalist approach, address and document only the issues that need addressing at various points in the life cycle.
Decision	Clearly state the architecture's direction – that is, the position you've selected.
Status	The decision's status, such as pending, decided, or approved.
Group	You can use a simple grouping – such as integration, presentation, data, and so on – to help organize the set of decisions. You could also use a more sophisticated architecture ontology, such as John Kyaruzi and Jan van Katwijk's, which includes more abstract categories such as event, calendar, and location [KyKa99]. For example, using this ontology, you'd group that deal with occurrences where the system requires Information under event.
Assumptions	Clearly describe the underlying assumptions in the environment in which you're making the decision – cost, schedule, technology, and so on. Note that environmental constraints (such as accepted technology standards, enterprise architecture, commonly employed patterns and so on) might limit the alternatives you consider.
Constraints	Capture any additional constraints to the environment that the chose alternative (the decision) might propose.
Positions	List the positions (viable options or alternatives) you considered. These often require ling explanations, sometimes even models and diagrams. This isn't an exhaustive list. However, you don't want to hear the question "Did you think about …?" during a final review; this leads to loss of credibility and questioning of other architectural decisions. This section also helps ensure that you heard others' options; explicitly stating other options helps enroll their advocates in your decision.
Argument	Outline why you selected a position, including items such as implementation cost, time to market, and required development resources' availability. This is probably as important as the decision itself.
Implications	A decision comes with many implications, as the REMAP metamodel denotes. For example a decision might introduce a need to make other decisions, create new requirements, or modify existing requirements; pose additional constraints to the environment; require renegotiating scope or schedule with customers; or require additional staff training. Clearly understanding and stating your decision's implications can be very effective in gaining buy-in and creating a roadmap for architecture execution.
Related decisions	It's obvious that many decisions are related; you can list them here. However, we've found that in practice, a traceability matrix, decision trees, or metamodels are more useful. Metamodels are useful for showing complex relationships diagrammatically (such as Rose models).
Related requirements	Decisions should be business driven. To show accountability, explicitly map your decisions to the objectives or requirements. You can enumerate these related requirements here, but we've found it more convenient to reference a traceability matrix. You can assess each architecture decision's contribution to meeting each requirement, and then assess how well the requirement is met across all decisions. If a decision doesn't contribute to meeting a requirement, don't make that decision.
Related artifacts	List the related architecture, deign, or scope documents that this decision impacts.
Related principles	If the enterprise has an agreed-upon set of principles, make sure the decision is consistent with one or more of them. This helps ensure alignment along domains or systems.
Notes	Because the decisions-making process can take weeks, we've found it useful to capture notes and issues that the team discusses during the socialization process.

Tabelle 4: Elemente des Rationale Ansatzes zur Dokumentation von Architekturentscheidungen (Quelle: [TyAk05], S. 21)

3.2.2.2 Repräsentation des Rationale

Um seine Ziele zu erreichen, greift der Ansatz auf Elemente der Rationale Ansätze REMAP (vgl. Kapitel 2.1.4.4) und DRL (Kapitel 2.1.4.3) zurück und fügt weitere zwei Elemente (*related principles, notes*) hinzu. Die Elemente und ihre Beschreibung sind in Tabelle 4 dargestellt. Diese Tabelle wird auch als Vorlage (*architecture decision description template*, im Folgenden T1) zur Dokumentation der Entscheidungen benutzt.
Als Ausgangspunkt für eine Entscheidung wird, wie in den meisten Ansätzen, das Problem identifiziert und als Frage formuliert (*issue*). Bevor es zu einer Entscheidung kommt, werden auch in diesem Ansatz alternative Lösungen betrachtet, die gegen Kriterien bewertet werden. Dies geschieht mit Hilfe einer weiteren Vorlage (*evaluation QOC template*, im Folgenden T2), welche die QOC Schemaelemente Frage, Optionen und Kriterien sowie die Bewertung der Optionen für jedes Kriterium enthält (zum Aussehen vgl. auch Tabelle 3 und Abbildung 11). Zur Bewertung, inwieweit/ob durch die Option das Kriterium unterstützt und realisiert wird, stehen die Möglichkeiten *Ja*, *Nein* und *Unbekannt* zur Verfügung. Auf der Evaluation basierend wählt man eine Option, die dann mittels T1 dokumentiert wird. T2, welches die Evaluation der Optionen enthält, wird dann aus T1 unter *Related requirements* heraus (logisch) verlinkt. Dadurch wird nachvollziehbar, wie die Bewertung stattgefunden hat und welche Ziele, Geschäftsanforderungen und nicht-funktionale Anforderungen (die als Kriterien verwendet werden) berücksichtigt wurden.
Als dritte Repräsentationsform des architektonischen Entscheidungsprozesses werden z.B. Entscheidungsbäume verwendet, um die Zusammenhänge zwischen den Entscheidungen darzustellen. Diese werden unter *Related decisions* in T1 verlinkt. Die Beziehungen der einzelnen Dokumentationselemente, wie sie im vorigen Absatz erläutert wurden, sind ausgehend von einer fiktiven Architekturentscheidung „D05“ in Abbildung 11 aufgezeigt.
Als Informationsträger für diesen Ansatz können einfache Office Dokumente (Microsoft Word oder PowerPoint etc.) verwendet werden, auch wenn die Autoren mit „Rational Rose“ arbeiten. Somit ist der Ansatz auch ohne weitere Werkzeugeinführung nutzbar.

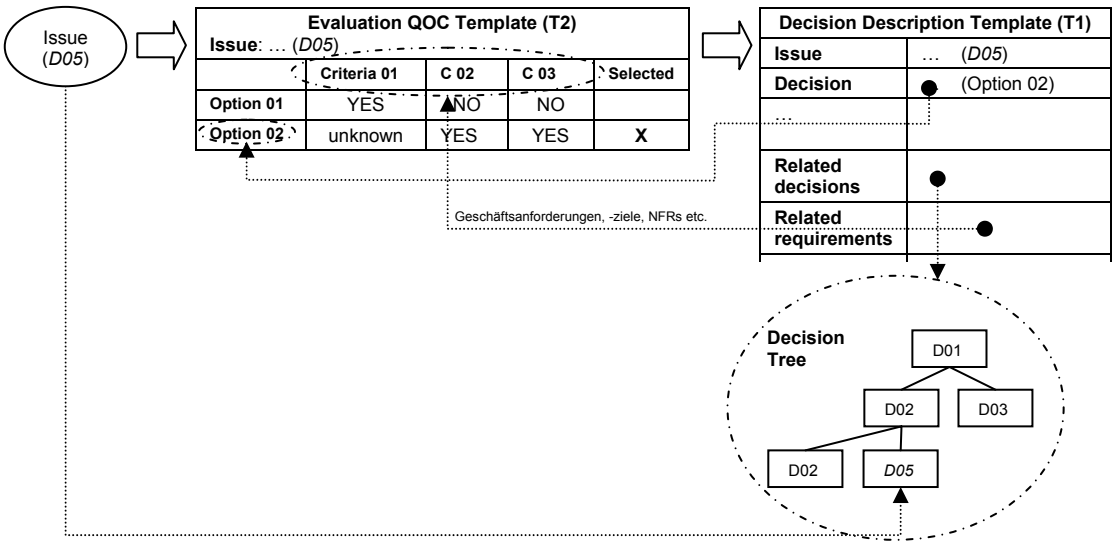

Abbildung 11: Zusammenhänge der Entscheidungsrepräsentation

3.2.2.3 Prozessimplementierung des Ansatzes

a) Rationaleidentifikation

Der Ansatz zielt darauf ab, die wichtigsten Architekturentscheidungen zu dokumentieren. Als Anhaltspunkt, was unter einer (wichtigen) „Architekturentscheidung“ verstanden wird, gibt er

folgendes vor: „To test a decision's architectural significance, an architect should ask the following question: does this decision affect one or more system qualities (performance, availability, modifiability, security, and so on)? If so, an architect should make this decision and document it completely" [TyAk05, S. 20]. Eine Entscheidung, ob bei einem System z.B. *Single Sign-On* (SSO) zur Authentifizierung Verwendung finden sollte, würde demnach als Architekturentscheidung gelten, da sie zumindest die Qualitätsattribute (NFRs) Benutzerfreundlichkeit und Sicherheit beträfe.
Weiterhin wird basierend auf [MaBr02] angemerkt, dass ein Architekt so wenige Entscheidungen wie möglich treffen und die detailreicheren Entscheidungen auf spätere Tätigkeiten verschieben sollte.

b) Rationaleerfassung

Der Ansatz macht keine direkten Angaben zur Erfassung. Es ist aber davon auszugehen, dass das Rationale in den Gesprächen mit den Interessenvertretern, der Analyse der Geschäftsanforderungen, -ziele und der NFR sowie den Diskussionen der Architekten untereinander erfasst wird. Um später die Entwicklung des Rationale in die formale Repräsentation zu ermöglichen, sollte in den Diskussionen der Architekten derart vorgegangen werden, dass man die Entscheidungen als Problem formuliert, Alternativen und Kriterien identifiziert, diese evaluiert (mittels T2) und die Entscheidungen grob in T1 dokumentiert. Zumindest sollten die Interessenvertreter über diese Elemente nachdenken, um dies später während der Rationaleentwicklung zu dokumentieren.

c) Rationaleentwicklung

Die endgültige Entwicklung des Rationale in der formalen Darstellung der Vorlagen (*templates*) und dem Entscheidungsbaum kann verquickt mit der Erfassung, aber auch erst im Nachhinein stattfinden. Dies kommt darauf an, wie viel Zeit man für das Rationale Management in den Treffen und während den Diskussionen aufwenden kann.

d) Rationaleverteilung/Rationalenutzung

Weil das mit diesem Rationale Ansatz erfasste Rationale wohl meist in Office Dokumenten dokumentiert werden wird, findet der Zugriff und die Verteilung meist über eine (zentrale) Ablage (Verzeichnis, Content Management System etc.) statt.
Da der Ansatz die Ziele verfolgt, die Architektur nachvollziehbar und transparent zu machen, die Interessenvertreter von den Entscheidungen zu überzeugen (*socialize the architecture*), die Entscheidungen kontrollierbar zu machen und zu überprüfen, ob alle wichtigen Anforderungen entsprechend berücksichtigt und gewichtet wurden, müssen die Entscheidungen möglichst vielen Leuten zugänglich gemacht werden. Die aus den Vorlagen erstellten Dokumente mit den Entscheidungen und ihrem Rationale werden also als Basis für die Kommunikation an die verschiedenen Interessenvertreter verwendet. Dabei existieren zwei Sichten:

- Die technische Sicht und die zugehörigen Detaildiskussionen spiegeln sich in den beschriebenen Repräsentationsformen (T1, T2, Entscheidungsbäumen etc.) wider. Diese werden genutzt, um die technischen Interessenvertreter zu adressieren.
- Eine zweite Sicht, die aber nicht näher von den Autoren beschrieben wird, wird verwendet, um die Entscheidungen den geschäfts- und managementnahen Interessenvertretern zu kommunizieren (meist in Form von PowerPoint Präsentationen).

Die Zugreifbarkeit und das Auffinden des gewünschten Rationale wird in diesem Ansatz dadurch unterstützt, dass zwischen den Entscheidungen (Entscheidungsbaum) und den Dokumenten Verbindungen hergestellt werden und dadurch eine Verfolgbarkeit entsteht (vgl. Abbildung 11). Wie in Kapitel 3.2.2.2 bereits beschrieben, wird dadurch auch eine Verfolgbarkeit bis auf die Ziele und die verschiedenen Anforderungen ermöglicht, die bei der Entscheidung berücksichtigt wurden und in die Architektur Eingang fanden. Das *Group* Element in T1 soll weiterhin eine Kategorisierung der Entscheidungen entlang einer

Taxonomie beziehungsweise Ontologie (vgl. Beschreibung zu Element *Group* in Tabelle 4) und somit eine Filterung z.B. für eine bestimmte Zielgruppe oder einen bestimmten Zweck ermöglichen. Wird eine Entscheidung also einer Kategorie zugeordnet (z.B. der Kategorie „Benutzungsoberfläche“), kann hiernach gefiltert werden, falls man nach einer Entscheidung aus diesem Bereich sucht.

3.2.3 Rationale im Kontext von Implementierung und Wartung

Im Folgenden wird ein Rationale Ansatz präsentiert, der während der Tätigkeit der Implementierung und der Softwarewartung (*maintenance*) und -evolution angesiedelt ist [BuBr06; Burg05; BuBr03; BuBroJ]. Er ist eingebettet in die integrierte Entwicklungsumgebung (*integrated development environment*, IDE) Eclipse [22] – genauer in die Java Entwicklungsumgebung von Eclipse – mittels eines Plug-in namens SEURAT [23] (*Software Engineering Using RATionale*). Dieses hat momentan aber noch prototypischen Charakter. Durch die Einbettung in die IDE kann der Ansatz erst einmal nur in diesem technischen Kontext verwendet werden. Eine entsprechende Werkzeugunterstützung vorausgesetzt könnte er aber auch in anderen Umgebungen zum Einsatz kommen.
Auch für diesen Ansatz wird kein Beispiel vorgestellt, es sei aber für ein solches auf [BuBr06] verwiesen.

3.2.3.1 Ziel des Rationale Ansatzes

Das Hauptziel des SEURAT Ansatzes ist die Untersuchung und anschließende Verwendung von Rationale zur Unterstützung der Softwarewartung und -evolution. Dabei werden dem Nutzer vielfältige Hilfsmöglichkeiten zur Verfügung gestellt, die im Folgenden noch ausführlicher erläutert werden.
Ein weiteres Ziel ist, die Hemmschwelle für die Erfassung und Nutzung von Rationale zu reduzieren, indem das Werkzeug in eine IDE integriert wird. Hiermit sollen die mit der Implementierung und Wartung betrauten Rollen zur Verwendung von Rationale motiviert werden.

3.2.3.2 Repräsentation des Rationale

SEURAT bietet verschiedene Prüf- und Inferenzmöglichkeiten für das erfasste Rationale. Hierzu muss das Rationale in eine strukturierte und formalisierte Darstellung gebracht werden. Die Autoren haben dazu ein Schema entwickelt, das sie RATSpeak nennen und welches auf dem DRL Schema basiert (vgl. Kapitel 2.1.4.3). Allerdings wurden an diesem Schema vereinzelt Änderungen vorgenommen. Das RATSpeak Schema ist in Abbildung 12 dargestellt und enthält folgende Elemente (Entitäten) [BuBr06, S. 280f.]:

Requirements. Diese beinhalten sowohl funktionale und nichtfunktionale Anforderungen. Sie können entweder explizit im Rationale angegeben werden oder Verweise auf Anforderungen in einem Anforderungsdokument oder einer Datenbank sein. *Requirements* spiegeln eine Art der Ziele wider, die mit Entscheidungen erreicht werden sollen und sind daher eine Basis für die Argumentation (für oder gegen Alternativen).
Decision Problems. Dies sind die Entscheidungen, die während des Entwicklungsprozesses getroffen werden müssen.
Questions. Dies sind Fragen, die beantwortet werden müssen, bevor das *Decision Problem* bestimmt werden kann. Eine Frage kann z.B. inhaltlich auf Prozeduren oder Programme bezogen sein, die ausgeführt werden müssen. Sie können aber auch nach Personen fragen, die eine Antwort auf ein Problem geben können. Fragen erweitern die Problemdiskussion, indem sie die Quelle einer Information angeben, die verwendet wird, um eine Entscheidung zu treffen.

[22] http://www.eclipse.org/
[23] http://www.users.muohio.edu/burgeje/SEURAT/index.html

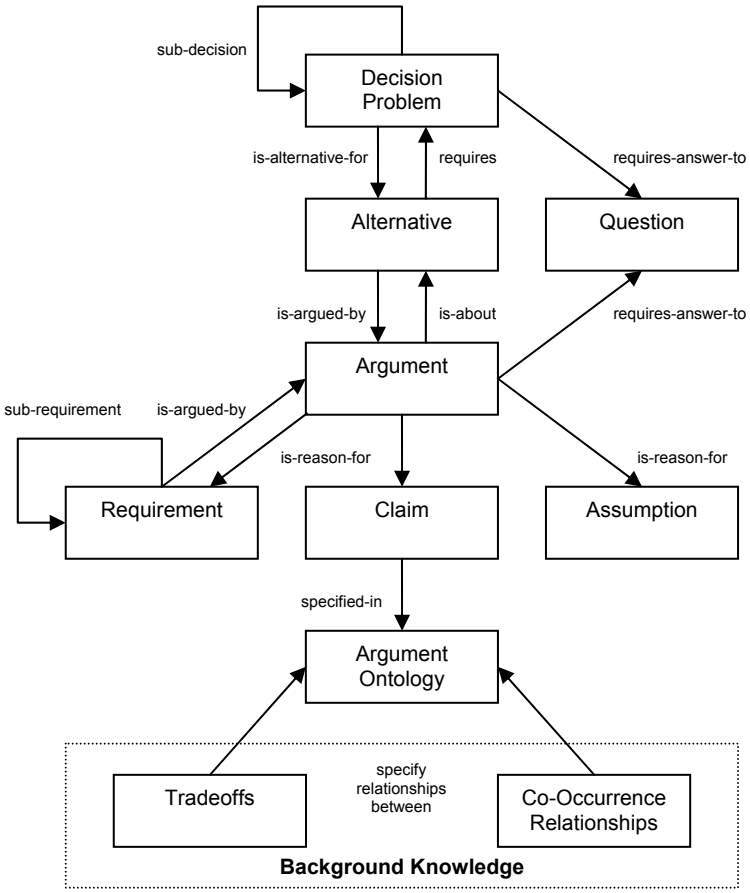

Abbildung 12: Beziehungen zwischen den RATSpeak Rationale Entitäten (Quelle: in Anlehnung an [Burg05, S. A-8])

Alternatives. Dies sind alternative Lösungsmöglichkeiten für ein *Decision Problem*. Alternativen haben einen Status der besagt, ob sie angenommen (*accepted*), zurückgewiesen (*rejected*) oder keines von beidem sind (*pending*).
Arguments. Dies sind Argumente für oder gegen vorgeschlagene Alternativen. Sie können entweder

- auf Anforderungen (*requirements*) verweisen (z.B. eine Alternative ist gut oder schlecht bezüglich einer Anforderung),
- auf Behauptungen (*claims*) über eine Alternative oder
- auf Annahmen (*assumptions*), welche Gründe für oder gegen die Wahl einer Alternative sprechen.

Diese Elemente sind dann der „Inhalt" eines Arguments. Jedes Argument bekommt eine Gewichtung (inwieweit ein Argument auf eine Alternative zutrifft, z.B. wie flexibel oder teuer eine Alternative ist) und eine Wichtigkeit (wie wichtig das Argument bezüglich der Entscheidung oder bezüglich des Gesamtsystems ist). Damit haben sie einen bestimmbaren Einfluss auf das Projekt.
Claims. Dies sind Behauptungen, warum eine Alternative gut oder schlecht ist. Jede Behauptung bezieht sich auf einen Eintrag in einer Argumentontologie (*Argument Ontology*), die allgemeine Argumente für Softwareentscheidungen enthält. Ein *Claim* enthält auch Informationen darüber inwiefern ein Argument eine Alternative unterstützt oder zurückweist: z.B. ob eine Alternative NICHT sicher ist (negative Beziehung) oder flexibel IST (positive Beziehung). *Claims* können ebenfalls mit einer „Wichtigkeit" attributiert werden, die von den

auf den *Claim* verweisenden Argumenten (*arguments*) übernommen oder überschrieben werden kann.
Assumptions. Annahmen sind ähnlich zu *Claims*, allerdings ist unklar, ob sie immer wahr sind oder in der Zukunft gültig bleiben. Annahmen beziehen sich auch nicht auf Einträge in der *Argument Ontology*.
Argument Ontology. Dies ist eine Hierarchie von allgemeinen argumentativen Begründungen, die für Entscheidungen bezüglich eines Softwaresystems herangezogen werden können (z.B. Entwicklungskosten, Qualitätsattribute der ISO 9126 [ISO9126] etc.). Diese werden verwendet, um ein gemeinsames Vokabular für die Argumentation und Schlussfolgerungen (*inferencing*) vorzugeben. Jeder Ontologieeintrag enthält dafür einen Standardwert (*default*) für die Wichtigkeit, der aber von darauf verweisenden Behauptungen (*claims*) überschrieben werden kann. Eine komplette Liste der Ontologieeinträge kann in [Burg05] gefunden werden.
Background Knowledge. Dieses enthält Wechselwirkungen (*tradeoffs*) und Abhängigkeiten (*Co-Occurrence Relationships*) in den Beziehungen der Ontologieelemente untereinander. Es wird nicht für die Schlussfolgerungen genutzt sondern zur Überprüfung von Verletzungen dieser Zusammenhänge im angegebenen Rationale.

Einige der Elemente und ein Teil der *Argument Ontology* sind auch in Abbildung 13 dargestellt.

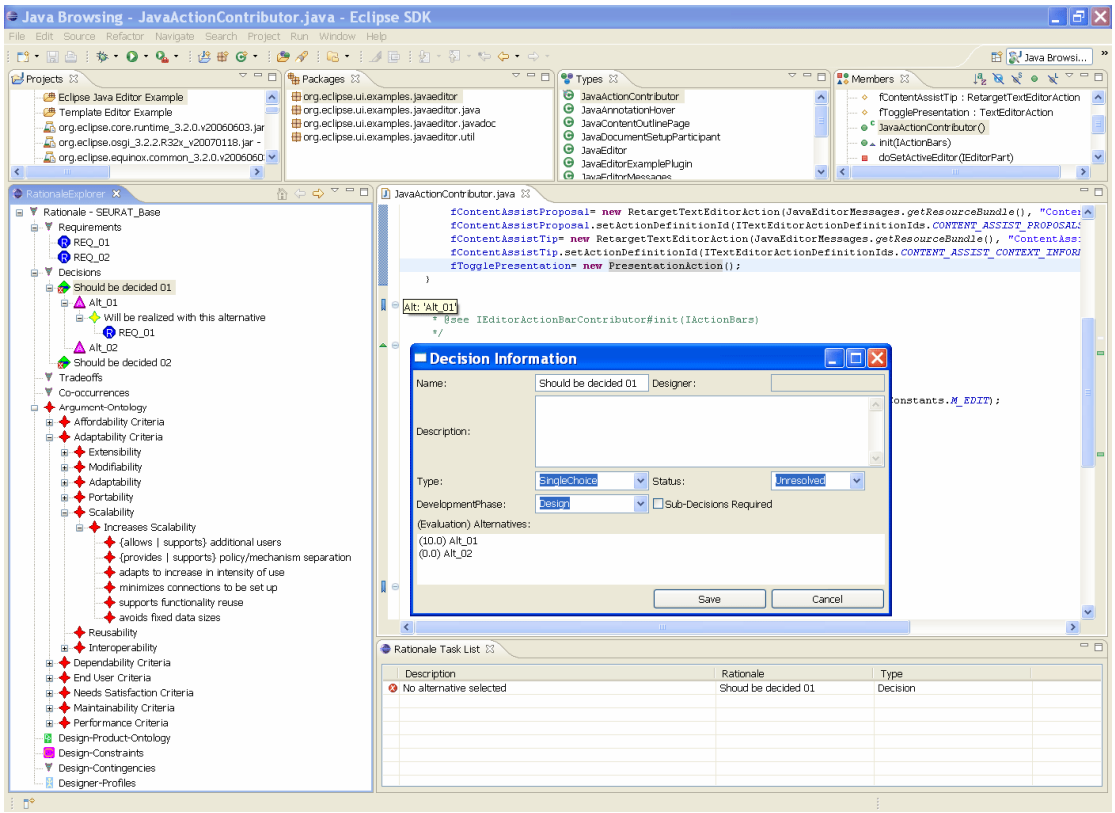

Abbildung 13: SEURAT Werkzeug in der Eclipse IDE

3.2.3.3 Prozessimplementierung des Ansatzes

a) Rationaleidentifikation

Zur Rationaleidentifikation macht dieser Ansatz keine präzisen Vorgaben und leistet damit auch keine Hilfe bei der Identifikation des Rationale, das dokumentiert werden soll. Allerdings ist dies den Urhebern des Ansatzes durchaus bewusst und auch zum jetzigen Stand der Entwicklung intendiert, da sie die späteren Aufgaben des Rationale Management

„Rationaleverteilung“ beziehungsweise insbesondere „Rationalenutzung“ fokussieren [BuBr06, S. 284].

b) Rationaleerfassung/Rationaleentwicklung

Die Erfassung und die Entwicklung finden verzahnt durch den Entwickler statt. Dieser soll während des Entwicklungsprozesses das Rationale als integralen Bestandteil seiner Arbeit betrachten, was durch die Integration von SEURAT in die IDE unterstützt wird. Obgleich auch der Fokus nicht auf diesen Aufgaben des Rationale Management liegt, werden Mittel zur Unterstützung dieser Aufgaben zur Verfügung gestellt. In der IDE existieren z.B. Dialoge zur Erfassung der einzelnen Rationaleelemente, zum Editieren und zur Verknüpfung mit den Codeelementen (vgl. auch Abbildung 13). Bei der Erfassung und Entwicklung des Rationale durch SEURAT wird auch gleich dessen Vollständigkeit geprüft. Wird z.B. eine Lösungsalternative für eine Problemstellung gewählt, so wird davon ausgegangen, dass es hierfür eine Begründung gibt oder die Wahl auf Basis einer Evaluation der Alternativen bezüglich bestimmter Kriterien stattfand. Fehlt beides – eine Begründung und eine Evaluation – meldet SEURAT, dass das Rationale nicht vollständig erfasst wurde.
Wie bei der Beschreibung von RATSpeak erwähnt, ist auch eine Argumentationsontologie (*Argument Ontology*) zentraler Bestandteil von SEURAT [Burg05, S. 96ff.]. Diese Ontologie wird, abgestimmt auf das Projekt, dem Entwickler zur Verfügung gestellt und soll ein gemeinsames Verständnis schaffen, welche Aspekte bei der Entwicklung und damit verbundenen Entscheidungen beachtet werden müssen. Weiterhin bildet sie die Grundlage für die Inferenzfunktionalität. Denn die einzelnen Ontologieelemente können nach ihrer Bedeutung für den Design-Prozess gewichtet und dann für die (automatisierte) Evaluation und Wahl der optimalen Lösungsalternativen herangezogen werden [BuBr06, S. 282].

c) Rationaleverteilung

Der Zugriff auf das Rationale findet über das SEURAT Werkzeug statt, wofür es mehrere Sichten zur Verfügung stellt (vgl. auch Abbildung 13). Eine enthält z.B. die Markierung der Codeelemente im Codeeditor, zu denen Rationale existiert. Von diesen Markierungen kann zum Rationale navigiert werden. Weiterhin gibt es auch hierarchische Ansichten des gesamten Rationale, welches auf verschiedene Art gefiltert werden kann (vgl. [BuBr06, S. 279]). Unterstützt werden auch mehrerlei Abfragemöglichkeiten für die Rationaleelemente (vgl. [BuBr06, S. 291]), welche zusätzlich (über relationale Abfragesprachen) flexibel anpassbar sind, da das Rationale in einer relationalen Datenbank hinterlegt wird.

d) Rationalenutzung

Das Rationale kann immer dann verwendet werden, wenn ein Codeelement, zu welchem Rationale erfasst wurde, eine Veränderung erfahren soll. SEURAT ist aber nicht darauf beschränkt und ermöglicht über seinen Inferenzmechanismus weitaus mehr Möglichkeiten, um den Entwickler durch Warnungen darin zu unterstützen, Änderungen sorgfältig und konsistent durchführen und verfolgen zu können. Im Folgenden werden zwei Beispiele hierfür genannt, weitere findet man in [BuBr06, S. 284ff.].

> *Beispiel 1*. Während der Entwicklung eines Software Systems werden einmal getroffene Annahmen (*Assumptions*) gegebenenfalls mit der Zeit ungültig. SEURAT erlaubt es, solche Annahmen zu erfassen und als Argument für die Wahl einer Alternative anzugeben (vgl. Kapitel 3.2.3.2). Werden die Annahmen später ungültig, kann der Entwickler dies angeben. SEURAT prüft dann alle darauf basierenden Begründungen und evaluiert sie neu. Werden somit gewählte Alternativen suboptimal, wird der Nutzer gewarnt. Von der Warnung aus kann zu den betroffenen Codebereichen navigiert werden. Entscheidungen können dann revidiert und es können aktuell bessere Lösungsalternativen gewählt und implementiert werden.
>
> *Beispiel 2*. Es kann vorkommen, dass sich die Schwerpunkte beziehungsweise Prioritäten für die einzelnen Begründungen (*Arguments*) einer Entscheidung im Verlaufe der

Entwicklung ändern. Z.B. könnte zu Beginn einer Produktentwicklung der Schwerpunkt auf schnelle Ergebnisse (z.B. Prototypen) gelegt werden und später die Flexibilität und Wiederverwendbarkeit des Codes stärkere Gewichtung erfahren. Es wäre nun von Vorteil, wenn man die Auswirkungen dieser Bewertungsänderung „simulieren" könnte. SEURAT unterstützt dies über so genannte „*what-if*" Inferenz. Hier können die Gewichtungen an zentraler Stelle geändert werden. Damit berechnet SEURAT erneut die optimale Alternative. Ändert sich diese zur alten optimalen Alternative, wird der Entwickler über alle Entscheidungen informiert, in denen das der Fall ist und kann hierüber wiederum zu den betroffenen Codebereichen navigieren und entsprechend reagieren.

3.2.4 Vergleichende Zusammenstellung der Ansätze für den Bereich PPR

Wie im Kapitel zur Anwendung von Rationale im Kontext von Diskussion und Treffen, zeigen sich auch im Umfeld des PPR verschiedenen Herangehensweisen an das Rationale Management.

Der vorgestellte Ansatz aus dem Bereich Requirements Engineering (Kapitel 3.2.1) ist ein Beispiel für die Integration von Ideen des Rationale Management in einen Softwareentwicklungsprozess und das zugehörige CASE-Werkzeug als „*first class entities*" (vgl. auch [WoDuoJ]). Hierdurch wird den Rationaleelementen ein zentraler Stellenwert eingeräumt, der nochmals unterstrichen wird durch die Rolle des *Rationale Maintainer* und speziell beschriebenen Prozessen für das Rationale Management innerhalb des Entwicklungsprozesses [DuPa01, S. 806ff.]. Dadurch soll auch die Bewusstseinsänderung der Interessenvertreter hin zu einer selbstverständlichen Durchführung von Rationale Management und Nutzung von Rationale während des Software Engineering unterstützt werden.

Mit der Integration des Rationale in das CASE-Werkzeug soll eine Senkung der Schwelle für die Rationaleerfassung stattfinden, indem den für die Erfassung und Nutzung zuständigen Rollen ihre gewohnte Arbeitsumgebung zur Verfügung gestellt wird. Ein wichtiger Gesichtspunkt ist in diesem Zusammenhang auch die Verfolgbarkeit (*traceability*) von Anforderungselementen, deren Verfeinerungen und dem zugehörigen Rationale, zwischen denen durch physische Verweise[24] einfach navigiert werden kann.

Eine Herausforderung für die Anwendung des Ansatzes in der Industrie ist die Rechtfertigung der initialen Zusatzkosten für die Erfassung des Rationale. Dies resultiert vor allem daraus, dass nicht nur Aufgaben von existierenden Rollen um rationale-relevante Tätigkeitsbereiche ergänzt werden, sondern auch zusätzliche Rollen wie der *Rationale Maintainer* eingeführt werden. Und deren Aufgaben können nicht einfach nebenher oder zusätzlich ausgeführt werden. Die Rechtfertigung hierfür wird umso schwerer, je weniger der Wert und Nutzen des Rationale anerkannt ist.

Außerdem besteht die Schwierigkeit, das Rationale in den Elementen des recht starren QOC Schemas zu repräsentieren (siehe dazu auch [LoLo00, S. 222]).

Auch der Rationale Ansatz im Kontext von Architekturentscheidungen (Kapitel 3.2.2) postuliert die Erhebung der Entscheidungen und deren Rationale zu einem „*first-class*" Status [TyAk05, S. 20]. Das Hauptziel ist, die Architekturentscheidung transparent und verständlich für alle Interessenvertreter zu machen, wozu die Vorlage T1 (siehe Tabelle 4) als „gemeinsame Sprache" zur Diskussion von Entscheidungen dient. Durch die Verwendung der verschiedenen Vorlagen erhält der Ansatz mehr Ausdruckskraft, da in diesen die Elemente von verschiedenen argumentations-basierten Rationale Schemata (REMAP, DRL, QOC)

[24] Als physische Verweise werden hier Verweise bezeichnet, die in den Softwaresystemen repräsentiert sind und eine unmittelbare Navigation über das System ermöglichen (z.B. Hyperlinks). Logische Verweise sind dagegen Verweise, anhand derer man nicht unmittelbar über das System zur Referenz gelangt. Ein herkömmlicher Eintrag in einem Literaturverzeichnis wäre z.B. ein logischer Verweis.

enthalten sind. Dies bietet auch eine größere Flexibilität der Beschreibung des Rationale als z.B. nur das QOC Schema des Ansatzes für das Requirements Engineering.
Der Ansatz ermöglicht weiterhin eine von einem speziellen Werkzeug unabhängige Durchführung, auch wenn die Autoren ihn innerhalb von „Rational Rose“ verwenden. Allerdings impliziert das gleichzeitig, dass die Verwaltung des Rationale nicht integriert stattfindet und deshalb verschiedene Einschränkungen bestehen: z.B. sind die logischen Verweise für die Verfolgbarkeit zwischen den verschiedenen aus den Vorlagen hervorgegangenen Dokumenten sowie dem Entscheidungsbaum aufwendiger aktuell und konsistent zu halten. Eine werkzeuginterne Navigation über physische Verweise, wie in dem Sysiphus Werkzeug oder auch dem SEURAT Werkzeug, vereinfacht hier die Verwaltung und Nutzung des Rationale.
Der SEURAT Ansatz hebt sich von den anderen Ansätzen dadurch ab, dass er der nutzenden Rolle des SEURAT Werkzeugs eine Reihe von Prüfungen des Rationale durch Inferenzmechanismen sowie Abfrage- und Filtermöglichkeiten zur Verfügung stellt. Über diese vielfältigen Prüfungen und Inferenzmöglichkeiten und die resultierenden Warnungen bei Regel- beziehungsweise Konsistenzverletzungen unterstützt er den Entwickler bei seiner Tätigkeit. Diese Unterstützung besteht sowohl bei der Erfassung und Entscheidungsfindung (präskriptiv), als auch bei der Nutzung des Rationale (deskriptiv). Durch die Speicherung des Rationale in einer Datenbank basierenden auf dem zugrunde liegenden Schema RATSpeak und die explizite und „physische“ Verknüpfung dieser Elemente mit den Codeelementen in der IDE, wird auch eine Verfolgbarkeit z.B. von den Bewertungskriterien und Argumenten zu den Entscheidungen und den betroffenen Codeelementen möglich.
Die Herausforderung dieses Ansatzes ist im Zusammenhang mit der Menge des Rationale zu sehen, die auf Codeebene auftreten kann. Denn bei mehreren Millionen Zeilen Code kann selbst die Rationaleerfassung und Verwaltung für einen Bruchteil der Codeelemente einen enormen Aufwand darstellen.

3.3 Wissensbereich Organisationswissen – rationale for organizing bodies of knowledge (OBR)

Auch der Wissensbereich des Organisationswissens umfasst die Teilbereiche System- und Prozesswissen. Allerdings unterscheiden sich die Aufgaben des Rationale Management und die Repräsentation des Rationale für diesen Wissensbereich sehr von dem des Produktwissens [PAECH06]. Vor allem das Erfassungsproblem hat weniger Bedeutung, da das Rationale über verschiedene Projekte gesammelt wird und die Dringlichkeit der Rationaleerfassung und -entwicklung parallel zu den Softwareentwicklungstätigkeiten nicht so hoch ist. So steht mehr Zeit zur Aufbereitung zur Verfügung. Weiterhin sind die Nutzer des Rationale nicht nur Entwickler, sondern auch (Hochschul-) Lehrer, Trainer, Berater und Studierende. Dies hat Auswirkungen auf die Repräsentation des Rationale sowie die Rationaleverteilung und -nutzung. Um das Rationale zugänglich zu machen, wird diese Art von Rationale häufig in (Lehr-) Büchern beziehungsweise offenen oder firmeninternen Archiven (*repositories*) erfasst, repräsentiert und zur Verfügung gestellt.
Das wohl bekannteste Beispiel für generalisiertes ORB [25] im Umfeld des Software Engineering sind die Entwurfsmuster (*design pattern*) für Architektur, Design und Implementierung [GHJV95]. [HHLP06] beschreibt einen entsprechenden Musteransatz für den Wissenstransfer im Tätigkeitsbereich des Requirements Engineering (so genannte RE Patterns), welche im Gegensatz zu ersteren (Wissensbereich Organisations-/Systemwissen)

[25] Als *fallbezogenes* (*case-based*) ORB wird ORB bezeichnet, welchem individuelle beziehungsweise einmalige Erfahrungen zugrunde liegen. *Generalisiertes* (*generalized*) ORB zielt auf die Konsolidierung und Generalisierung von verschiedenen einzelnen Erfahrungen [Paec06, S. 349].

dem Wissensbereich Organisations-/Prozesswissen zugeordnet sind. Im Internet existiert hierzu ein *Requirements Engineering Pattern Repository* (REPARE), welches die RE Muster zugänglich macht [REPARE].
Die genannten zwei Beispiele für ORB repräsentieren jedes einen Wissensbereich des Organisationswissens. Im Folgenden wird ein weiterer Rationale Ansatz für den Wissensbereich Organisations-/Prozesswissen ausführlicher dargestellt.

3.3.1 Rationale im Kontext von Requirements Engineering Process Improvement (REPI)

Der vorgestellte Ansatz findet Anwendung im Kontext von Prozessverbesserungen im Requirements Engineering (*Requirements Engineering Process Improvement*, REPI) [PaRi06]. Das hier erfasste Rationale wird in [PaRi06, S. 393] „REPI Rationale“ genannt. REPI Rationale Management soll von den Prozessverantwortlichen (*Process Owner*) beziehungsweise der Prozessbegleitung [26] (*Process Governance*) während der Prozessänderung beziehungsweise -einführung erfasst werden.

3.3.1.1 Ziele des Ansatzes

Ziel des REPI Rationale Ansatzes ist es, die Gründe für die Prozessverbesserungen zu erfassen. Dies soll genutzt werden, um z.B. die Vorgängervarianten der Prozesse mit dem neuen Stand vergleichen zu können. Damit kann man wirklich erzielte Verbesserung identifizieren, Entscheidungen begründet zurücknehmen, zu vormaligen Prozessvarianten zurückkehren oder andere alternative Wege einschlagen. Weiterhin soll die Akzeptanz und Erfüllung der Prozessvorgaben gestärkt und über die Präsentation und Kommunikation des Rationale an die beteiligten Interessenvertreter und deren Rückmeldungen weitere Verbesserungen erzielt werden.

3.3.1.2 Repräsentation des Rationale

Das Rationale in diesem Ansatz wird in tabellarisch strukturiertem Freitext erfasst. Der Weg zu einer Lösung und die Gründe einer Wahl werden vor allem über Verknüpfungen von verschiedenen Tabellen (-einträgen) ersichtlich. Um dies verständlich zu machen, werden erst einmal das zugrunde liegende REPI-Rahmenwerk (*framework*) und dessen Vorgehen sowie dessen Elemente skizziert.
Das Rahmenwerk legt das grundlegende Vorgehen bei der Prozessverbesserung fest und beschreibt, welche Themen adressiert werden müssen. Es basiert auf der Festlegung von gewünschten strategischen Ergebniszielen (*strategic outcomes*) für einen Prozess, wie z.B. Zeitersparnis bei bestimmten Tätigkeiten oder die Reduzierung von Anforderungsdefekten beziehungsweise deren frühzeitige Erkennung. Der Erreichungsgrad dieser Ziele wird auf Basis von Kenngrößen (*process dimensions/parameter*) gemessen (*process measurement*), und falls er nicht den Vorgaben entspricht, werden darauf basierend die Defizite (*process gaps*) in den Prozessen identifiziert (*process gap identification*). Sind die Defizite identifiziert, so werden sie priorisiert z.B. anhand von Dringlichkeit, Wichtigkeit und verbundenen Beseitigungskosten.
REPI Rationale soll nun für die zentralen Elemente des Rahmenwerks erfasst werden. Es sollen also z.B. die Gründe erfasst werden, warum bestimmte strategische Ergebnisziele (*strategic outcomes*) aufgestellt werden (*arguments to establish the strategy*, Tabelle 5) und

[26] Die Prozessbegleitung (*Process Governance*) unterstützt die Institutionalisierung der unternehmensweiten Geschäftsprozesse und deren Kommunikation. Sie liefert die Expertise, dass Prozessdefinitionen im Rahmen wohldefinierter Regeln entstehen und in der Organisation verankert werden, damit ein konformes Gesamtbild aller Prozesse entsteht und erhalten bleibt. Dazu gehört z.B. auch die Unterstützung bei der Modellierung, Ausgestaltung, Wartung und kontinuierlichen Verbesserung der Prozesse, Beratungstätigkeiten in Hinblick auf Geschäftsprozesse und die Durchführung von Assessments.

was man sich davon erhofft (*benefits*, Tabelle 5). Ein Beispiel für ein solches ‚Strategieelement' und das zugehörige REPI-Rationale ist in Tabelle 5 dargestellt.

process	strategy	arguments to establish the strategy	benefits
…			
RE	increased number of good quality requirements	Cost and schedule overruns are noticed by Z1's customers; therefore, it is important and urgent to reduce costs and schedule overruns	reduced cost, optimized schedules and enhanced customer satisfaction
…			

Tabelle 5: Rationale für Strategieelemente (Quelle: in Anlehnung an [PaRi06, S. 402])

Werden nun basierend auf den Kenngrößen Prozessdefizite identifiziert, werden deren Beseitigungsmaßnahmen ebenfalls erklärt und begründet. Dies geschieht über eine Adaption des QOC Schemas, welches etwas informaler verwendet wird. Die Frage (*Issue*) beschreibt hierbei die Prozessdefizite, die Kriterien (*Criteria*) die damit verbundenen Probleme und die Optionen (*Options*) die einbezogenen Lösungs- und Beseitigungsmöglichkeiten für die Prozessdefizite. Ein Beispiel wird im Folgenden gegeben:

Issue (*process gap*): Die Qualitätssicherung der Anforderungsspezifikationen wird nicht konsequent durchgeführt.
Criteria: Die Weiterarbeit mit den spezifizierten Anforderungen stellt sich für nicht an der Spezifikation beteiligte Rollen als schwierig heraus, und die Verständlichkeit der Anforderungen (die Intention der Autoren) kann nicht sichergestellt werden.
Options: Eine Checkliste zur Überprüfung der Anforderungsqualität wird den Autoren zur Verfügung gestellt. Der Autor der Anforderungsspezifikation soll seine Anforderungen daraufhin überprüfen oder
Es wird ein formaler Inspektionsprozess etabliert, der ebenfalls – basierend auf den Qualitätskriterien in der Checkliste – die Anforderungen in einem Abstimmungsprozess zwischen den beteiligten Interessenvertretern überprüft.

Über die Anwendung dieses Rationale Ansatzes werden also für das in Ausschnitten in Tabelle 5 gegebene und in dem adaptierten QOC Schema weitergeführte Beispiel folgende Überlegungen mit zugehörigem Rationale erfasst:
Es wird für den Requirements Engineering Prozess (Erstellen der Anforderungsspezifikation) das strategische Ergebnisziel aufgestellt, die erfassten Anforderungen qualitativ zu verbessern. Dies wird festgesetzt, da Projekte mit Kosten- und Zeitüberschreitungen zu kämpfen haben (*arguments to establish the strategy*, Tabelle 5). Sollte dieses Ergebnisziel erreicht werden, erhofft man sich Kosten- und Zeiteinhaltung sowie verbesserte Kundenzufriedenheit (*benefits*, Tabelle 5). Auf Basis von Kenngrößen (z.B. Anzahl der missverständlichen Systemanforderungen) in einer Spezifikation, die in diesem Beispiel nicht näher beschrieben werden, wird das Prozessdefizit (*process gap*) identifiziert. Dieses besteht darin, dass die Qualitätssicherung der Systemanforderungen nicht konsequent realisiert wurde (*Issue*). Die damit verbundenen konkreten Probleme werden dann als Kriterien für die Bewertung der Lösungsvorschläge herangezogen: mangelnde Verständlichkeit der Anforderungen und daraus resultierend eine schwierige Weiterarbeit mit diesen Anforderungen (*Criteria*). Als Lösungsalternativen werden deshalb entweder eine Qualitätssicherung der Anforderungen anhand einer Checkliste mit Qualitätskriterien durch den Autor selbst oder in einem Abstimmungsprozess mit mehreren Interessenvertretern vorgeschlagen (*Options*).

3.3.1.3 Prozessimplementierung des Rationale Ansatzes

a) Rationaleidentifikation

Bei der Identifikation des Rationale kann man zwischen zwei Dingen unterscheiden:

- der Rationaleidentifikation für die Elemente des REPI-Rahmenwerks (*strategic outcomes, process dimensions/parameter*), d.h. für die Evaluation der Prozesse und

- der Rationaleidentifikation für die Defizite und Prozessverbesserungsvorschläge, d.h. die eingeleiteten Maßnahmen.

Für beides soll entsprechendes Rationale erfasst werden. Ersteres wird in entsprechenden Tabellen erfasst, letzteres in der QOC verwandten Darstellung (vgl. Kapitel 3.3.1.2).

b) Rationaleerfassung/Rationaleentwicklung

Die Rationaleerfassung und -entwicklung kann sowohl gemeinsam, als auch entkoppelt stattfinden. Es können z.B. zuerst die Probleme sowie Prozessdefizite identifiziert und deren Lösung diskutiert werden. Das Rationale wird anschließend in einem weiteren Arbeitsschritt in die beschriebene Form gebracht und abgelegt.

c) Rationaleverteilung

Das Rationale ist dokumentiert in (Office) Dokumenten. Werden diese zur Verfügung gestellt, kann man auf das Rationale zugreifen. Zur Ablage muss eine sinnvolle Struktur gefunden werden, da die Menge an Rationale sonst schwer verwaltbar wird.

d) Rationalenutzung

Die Nutzung des Rationale ist für zwei Zielgruppen gedacht:

1. Die Prozessverantwortlichen (*Process Owner*) und Prozessbegleitung (*Process Governance*), die hierdurch die Überlegungen der Prozessänderungen beziehungsweise -einführung externalisieren. Damit können sie bei weiteren Änderungen dieser Prozesse auf dieses Wissen zurückgreifen, die Auswirkungen von Änderungen besser abschätzen beziehungsweise beurteilen und gegebenenfalls die Revision von Änderungen einleiten.
2. Die Ausführenden der Prozesse sollen ebenfalls mit dem Rationale konfrontiert werden, um die Intentionen der Prozesse zu verstehen. Hierdurch soll einerseits die Akzeptanz der Prozesse unterstützt werden, andererseits erhofft man sich durch die Rückmeldung der Ausführenden wertvolle Anregungen für weitere Prozessverbesserungen.

3.3.2 Vergleichende Zusammenstellung der Ansätze für den Bereich ORB

Der REPI Rationale Ansatz hat gezeigt, auf welche Weise man Rationale im Bereich der Prozessplanung einsetzen kann. Das Rationale wird hierbei in tabellarischer beziehungsweise QOC verwandter Form ohne spezialisiertes Werkzeug erfasst. Ein explizites und integriertes Schema zur Rationaleerfassung existiert ebenfalls nicht. Ähnlich dem Rationale Ansatz für Architekturentscheidungen (vgl. Kapitel 3.2.2), der sich von der Kommunikation der Entscheidungen und daraus resultierenden konstruktiven Rückmeldungen einen Beitrag zur Identifikation von Defiziten und zur Entscheidungsverbesserung erhofft, soll die Bereitstellung des REPI Rationale die Ausführenden der Prozesse über deren Intention informieren. Wenn diese dann als ‚Experten' ihrer Tätigkeiten bessere Umsetzungsmöglichkeiten zum Erreichen der Intention sehen, kann dies einen sehr wichtigen Beitrag für weitere Prozessverbesserungen liefern und dem Prozessentwickler (Prozessverantwortlicher, Prozessbegleitung) weitere Perspektiven aufzeigen.
Eine weitere Möglichkeit, Rationale im Kontext von Organisationswissen zu erfassen, stellen die in der Einleitung dieses Abschnittes genannten, aber nicht weiter beschriebenen Musteransätze (*design pattern*, RE Pattern) dar. Der Hauptunterschied der Musteransätze zu den meist schema-basierten Ansätzen (aus dem Bereich des PPR) besteht darin, dass sie weniger mit der Betrachtung und Evaluation von Optionen (Lösungsmöglichkeiten) zur Lösung eines Problems befasst sind [Paec06, S. 349]. Vielmehr umfassen sie eine Problembeschreibung (*problem description*) und einen Lösungsvorschlag (*recommended solution*) für

das Problem und die Diskussion des Vorschlags. Sie fokussieren damit weniger die Unterstützung der Entscheidungsfindung über eine Problemanalyse (*design space analysis*), sondern machen Vorschläge (aus der Erfahrung heraus), wie man ein gegebenes Problem lösen kann [Paec06, S. 349].

3.4 Zusammenfassung

Das Kapitel hat einen Überblick darüber gegeben, welche Anwendungsmöglichkeiten von Rationale Management und konkrete Umsetzungen in Prozessen, Werkzeugen und der Praxis existieren. Hierfür wurden die verschiedenen Wissensbereiche aus Kapitel 2.3 herangezogen und für jeden Wissensbereich ein Ansatz vorgestellt beziehungsweise darauf verwiesen. Da Rationale verstärkt in verschiedenen Tätigkeitsbereichen innerhalb des Wissensbereiches Produkt-/Systemwissen angewendet wird, wurden für die zentralen Tätigkeitsbereiche (Requirements Engineering, Architekturdefinition, Implementierung/Wartung) jeweils einzelne Ansätze gewählt und vorgestellt. Für den Wissensbereich Produkt-/Prozesswissen wurde kein Ansatz aus der Literatur aufgezeigt. Allerdings wird in Kapitel 4.4.1 hierzu ein Ansatz erarbeitet. Weiterhin wurden wissensbereichsunabhängige Ansätze dargestellt, die zur Strukturierung von Treffen und Diskussion und deren Abbildung in eine graphische Repräsentation geeignet sind.
In Tabelle 6 sind die genannten Ansätze und deren Zugehörigkeit zu den Wissensbereichen analog zu Tabelle 2 (Wissensbereiche im Software Engineering) zusammengefasst.

<table>
<tr><th></th><th>Systemwissen</th><th>Prozesswissen</th><th></th></tr>
<tr><td>Produktwissen</td><td>- Rationale im Kontext von Requirements Engineering (Kapitel 3.2.1)
- Rationale im Kontext von Architekturentscheidungen (Kapitel 3.2.2)
- Rationale im Kontext von Implementierung und Wartung (Kapitel 3.2.3)</td><td>- Kommunikation von Projektlenkungsentscheidungen (Kapitel 4.4.1)</td><td rowspan="3">Reasoning Loop Ansatz (Kapitel 3.1.2)</td></tr>
<tr><td>Organisations-wissen</td><td>- Design Patterns (siehe [GHJV95])</td><td>- Rationale im Kontext von Requirements Engineering Process Improvement (REPI) (Kapitel 3.3.1)
- RE Patterns (siehe [HHLP06, REPARE])</td></tr>
<tr><td></td><td colspan="2">Compendium Ansatz (Kapitel 3.1.1), Wisdom Ansatz (Kapitel 3.1.1.4)</td></tr>
</table>

Tabelle 6: Vorgestellte Rationale Ansätze und ihre Zugehörigkeit zu den Wissensbereichen

In diesem Kapitel wurde gezeigt, dass die einzelnen Ansätze sich in der Wahl ihrer Repräsentationsform unterscheiden. Tendenziell wählen die Ansätze zur Unterstützung von Diskussionen und deren parallelen Abbildung in Dokumentation (Dialogue Mapping) ein einfaches formales Schema. Die Ansätze im Bereich des Produkt-/Systemwissen verwenden dagegen eher umfangreichere und formalere Schemata, die Ansätze des Organisationswissens eher informale Schemata. Außerdem arbeiten sie verstärkt mit der Musteridee.
Weiterhin kann festgestellt werden, dass meist ein Spannungsverhältnis zwischen der Einfachheit der Abbildung von Diskussionen in ein Schema und der Verarbeitungs-möglichkeit des Schemas durch den Computer besteht. Je formaler das Schema, desto restriktiver sind die Freiheitsgrade für die Abbildung der Diskussion und Argumente in dieses Schema durch den Erfasser. Allerdings erhöht sich mit der Formalität die computergestützte Verarbeitungsmöglichkeit (vgl. SEURAT in Kapitel 3.2.3).

Wie bereits in Kapitel 2.1.4.4 angemerkt, basieren die vorgestellten Rationale Ansätze häufig auf den ursprünglichen Rationale Ansätzen wie IBIS, QOC, REMAP oder DRL. Dabei gehen sie zwar von diesen aus, einige nutzen aber eine „Vereinigung" der unterschiedlichen Elemente (z.B. der Ansatz zur Anwendung von Rationale für Architekturentscheidungen mit REMAP, DRL und QOC, Kapitel 3.2.2), wohingegen andere versuchen, die Gemeinsamkeiten der Ansätze zu identifizieren und sozusagen deren „Schnittmenge" als ein „generisches Modell" zu etablieren (z.B. Reasoning Loop, Kapitel 3.1.2) und einen „schlanken" Ansatz zur Verfügung zu stellen. Der Ansatz, eine Vereinigung von Schemata als neues Schema zu etablieren erhöht dessen Ausdruckskraft (aufgreifbare Aspekte). Dies kann das Problem von starren, formalen Schemata (voriger Absatz) etwas mildern, da die Beschreibungsmöglichkeit einer Entscheidung durch die Anzahl der Elemente erhöht wird. Gleichzeitig steigt zu einem gewissen Grad aber auch die Komplexität eines Ansatzes mit der Anzahl der einbezogenen Rationaleelemente.
Es wurde offensichtlich, dass sich die vorgestellten Ansätze zumeist auf einen begrenzten Ausschnitt des gesamten Software Lebenszyklus beziehen. Dutoit et al. stellen hierzu fest: „This led to a wealth of approaches for RM in specific SE activities. However, none of these approaches supports even half of the overall project activities. In other words, we are far away from having integrated RM support for SE decision making" [DMMP06a, S. IXf.]. Und weiter „While general-purpose methods have not been widely adopted, specialized approaches addressing narrow problems have emerged" [DMMP06b, S. 43]. Also muss die Integration der isolierten Ansätze noch stärker in den Fokus rücken, um die Nutzung des gesammelten Rationale an verschiedenen Stellen des Software Lebenszyklus zu unterstützen.
Erste Ansätze, eine integrierte Betrachtung von Rationale Management und die Einpassung des Rationale in ein „Big Picture" zu erhalten, werden zwar diskutiert, sind aber noch sehr allgemein gehalten. Man findet sie bei Dutoit et al. [DMMP06b, S. 25ff.], die einen Überblick geben, wie Rationale Management die im SPICE [SPICE] Prozessstandard definierten Aktivitäten unterstützen kann. Noch allgemeinere Betrachtungen des Nutzens von Rationale Management bei verschiedensten Aktivitäten während des Software Engineering findet man ebenfalls bei Dutoit et al. [DMMP06b, S. 16ff.] und weiterhin im *Handbook of Software Engineering and Knowledge Engineering* [DuPa01, S. 797ff.].
Neben diesen Herausforderungen gibt es noch zwei weitere, die in den meisten (vorgestellten) Ansätzen noch angegangen werden müssen:

1. Die Unterstützung bei der Identifikation, welches Rationale dokumentiert werden soll (aus der riesigen Mengen von Entscheidungen)
2. Das Aufzeigen einer konkreten Nutzung und damit Nutzens des Rationale in den weiteren Tätigkeiten nach dessen Erfassung

Zur zweiten Herausforderung bemerkt Burge: „[...] the usefulness of rationale has not been studied in as much detail as the capture and representation [...]" [BuBr06, S. 275]. Doch ist der konkrete Nutzen des Rationale der zentrale Anreiz, dieses auch nachhaltig zu erfassen („[...] the use of rationale is what is ultimately needed to motivate its capture" [BuBr06, S. 274]). Denn wenn dem Erfasser des Rationale nicht ganz eindeutig und greifbar klar ist, was der Nutzen für seine Disziplin und seinen Aufwand während der Erfassung ist, dann wird er dies nicht nachhaltig tun, erst recht nicht, wenn er nicht persönlich profitiert (vgl. Kapitel 2.2.2.4). Dutoit und Paech [DuPa01, S. 813] sehen die Möglichkeit für eine Nutzenanalyse allerdings erst, wenn gewisse Voraussetzungen gegeben sind: „Once adequate tool and organization support exists for capturing and disseminating rationale, it then becomes possible to conduct systematic quantitative studies to identify which subset of rationale knowledge are useful and used by developers and in which applications."
Weiterhin kann die erste Herausforderung (Identifikationsunterstützung) erst vollständig adressiert werden, wenn klar ist, welches Rationale von Nutzen ist und setzt deshalb Antworten auf die zweite Herausforderung voraus.

Die vorangegangenen Ausführungen zeigen einen Teil der Schwierigkeiten auf, mit denen die Forschung im Bereich des Rationale Management noch zu kämpfen hat und die mit ein Grund dafür sind, weshalb Rationale Management zurzeit kaum im industriellen Kontext verbreitet ist.
Zusammenfassend besteht also der Forschungsstand darin, dass vor allem die Rationale-erfassung und -entwicklung als auch die Repräsentation von Rationale bereits über längere Zeit untersucht werden, was sich an den vorgestellten Ansätzen in diesem Kapitel widerspiegelt. Deshalb werden zu anderen Aufgaben als Rationaleerfassung, -entwicklung und teils noch -verteilung sowie zur Repräsentation des Rationale weniger ausführliche Aussagen getätigt. Der Fokus müsste sich nun verschieben auf die Identifikationsunterstützung, die Nutzung und den Nutzen von Rationale. Erste Anstrengungen in diese Richtung macht in diesem Bereich z.B. der SEURAT Ansatz, dessen Autoren ausdrücklich feststellen: „We have chosen to address the use of and usefulness of rationale because the use of rationale is what is ultimately needed to motivate its capture“ [BuBr06, S. 274]. Auch gibt es erste Feldversuche, inwieweit dokumentiertes Rationale Fragen von Entwicklern und Designern beantworten [Kars96] beziehungsweise wie Rationale Änderungsprozesse unterstützen könnte [BrJR00, Burg05; BuBr06].

Kapitel 4

Anwendungsmöglichkeiten von Rationale Management im Rahmen des Softwareentwicklungsprozess-modells PIL@AP bei der SAP AG

In diesem Kapitel werden die Konzepte des Rationale Management auf das Umfeld der SAP AG angewendet, indem Vorschläge zur Durchführung von Rationale Management in verschiedenen Bereichen unterbreitet werden. Den Rahmen, in dem die Überlegungen zur Integration von Rationale Management stattfinden, gibt das in einem Teilbereich der SAP praktizierte Softwareprozessmodell „PIL@AP“ vor.
Dieses Prozessmodell wird im ersten Abschnitt des Kapitels (Kapitel 4.1) eingeführt. Es folgt eine allgemeine Definition der Ziele (entsprechend der Aufgabe „Identifikation von Rationale Zielen“ in Kapitel 2.2.2.1), die mit den Vorschlägen zur Integration von Rationale Management adressiert werden sollen (Kapitel 4.2). Daraufhin werden in Kapitel 4.3 Kriterien aufgestellt, um die Bewertung der Lösungsalternativen innerhalb der Vorschläge durchführen zu können. Diese Bewertung findet später zum Teil mittels Rationale Ansätzen statt, um deren Anwendung aufzuzeigen. In Kapitel 4.4 werden dann drei Vorschläge zur Anwendung von Rationale Management im PIL@AP Umfeld unterbreitet: Ein Vorschlag im Kontext der Kommunikation von Projektlenkungsentscheidungen (Kapitel 4.4.1), einer bei der Kommunikation und Verwaltung von Rahmenvorgaben zu Produktcharakteristiken (Kapitel 4.4.2) und einer zur Unterstützung von Priorisierungsentscheidungen von Anforderungen an Produkteigenschaften (Kapitel 4.4.3). Kapitel 4.5 fasst die Ausführungen abschließend zusammen.

4.1 Das Softwareprozessmodell PIL@AP

Softwareprozessmodelle geben einen Rahmen vor, in dem ein (Software-) Produkt innerhalb einer Organisation entwickelt wird [Balz00, S. 54]. Bei der SAP AG findet für die Produktentwicklung ein Prozessmodell namens „*Product Innovation Lifecycle*“ (PIL) Anwendung. Es besteht im Kern aus bewährten Verfahren (*best practices*), die für die globalen Entwicklungsaktivitäten der SAP AG standardisiert wurden. Diese beziehen z.B.

auch rechtliche Anforderungen und Regularien, Industriestandards, Kundenerwartungen und SAP-spezifische Qualitätsanforderungen ein.
Der PIL@AP ist eine Variante des PIL, angepasst für die Entwicklung der SAP Anwendungsplattform (*Application Platform*, AP) [GDS+07]. Diese Anwendungsplattform baut auf einer von SAP entwickelten Technologieplattform (SAP NetWeaver) auf und erweitert diese in Hinblick auf betriebswirtschaftliche Funktionalität (Geschäftslogik). Ergebnis ist eine Geschäftsprozessplattform (*Business Process Platform*, BPP).
Die Anpassung des PIL fand aufgrund des Vorhabens der SAP AG statt, für die Geschäftsprozessplattform (BPP) und die darauf basierenden Produkte einen verstärkt modellgetriebenen und ingenieurmäßigen Entwicklungsansatz zu etablieren. Damit sollten u.a. folgende drei Ziele erreicht werden:

1. **Produktivitätssteigerung** und klare Trennung der Softwareschichten durch eine so genannte Verantwortlichkeitstrennung („*separation of concern*“). Dahinter verbirgt sich die Bereitstellung einer Geschäftsprozessplattform mit der Geschäftslogik, die für verschiedene Produkte wieder verwendet und flexibel konfiguriert werden kann. Die Verantwortlichkeitstrennung eröffnet also die Möglichkeit des entkoppelten Konstruierens („*engineered decoupling*“) der Plattform und des Produktes (Konfiguration und Anpassung der Plattform und Bereitstellung einer Benutzungsoberfläche) und die Nutzung der Plattform für verschiedene Produkte.
2. **Qualitätssteigerung** der Software durch „Qualität durch Entwurf“ („*quality by design*“). Basis hierfür bilden die Softwaremodelle, aus denen Codeskelette (*proxies*) generiert werden.
3. **Qualitätssteigerung** durch „frühest möglichen Test“ („*test early*“). Grundlage hierfür ist das praktizierte Requirements Engineering, aus dem die dezidierten Anforderungen an die Produkteigenschaften hervorgehen, die in den Software Requirements Spezifikationen (SRS) festgehalten werden. Diese bilden mit den (statischen) Modellen gemeinsam die Grundlage für die Tests, die bereits parallel zur Produktentwicklung aufgestellt und baldmöglichst ausgeführt werden.

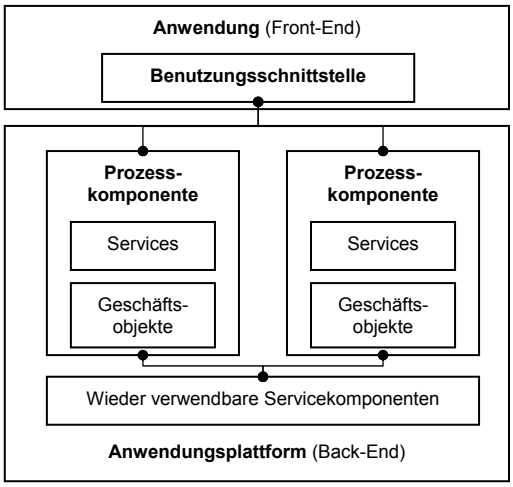

Abbildung 14: Zusammenhang der Enterprise Service-orientierten Architektur und ihrer Entitätstypen wie sie auch in der PIL@AP Taxonomie widerspiegeln (Quelle: in Anlehnung an [Gebh07a, S. 3]

Ausgerichtet wird der Entwicklungsprozess an wohldefinierten **Entitätstypen**, die in einer **Taxonomie** festgehalten sind und darüber kommuniziert werden [GDS+07, S. 6 und S. 30; Gebh07a]. Die Taxonomie beschreibt über die Entitätstypen u.a. die Elemente, aus denen sich das Produkt modular zusammensetzt (z.B. Szenarien, Prozesskomponenten, Geschäftsobjekte,

Services) und stellt daher (in Ausschnitten) das zugrunde liegende Architekturmodell des Produktes dar (vgl. Abbildung 14). Die einzelnen Entitätstypen werden hier nicht im Detail erläutert. Allerdings stellen z.B. Services Dienste zur Nutzung von Softwarefunktionalität zur Verfügung (vergleichbar mit dem allgemeinen Konzept der Web Services) und Geschäftsobjekte kann man sich ähnlich zu Klassen in UML [UML] vorstellen. Im Grunde stellen die Entitätstypen Analogien zu den UML-Elementen wie Klassen und Paketen etc. dar, sind aber nicht auf einen allgemeinen Standard ausgerichtet, sondern auf die Modellierung innerhalb der Softwareentwicklung bei der SAP AG im Kontext der Enterprise Service-orientierten Architektur (*enterprise SOA*) [WoMa06].
Mittels der Entitätstypen werden dann die Objekte der „realen" Welt als Entitäten modelliert. Daraus werden dann die Codeskelette (*proxies*) generiert und auf Basis der dynamischen Produktanforderungen (die in den SRS repräsentiert sind und zu den verschiedenen Entitäten erfasst werden) ausimplementiert. Diesem Vorgehen folgt auch die Entwicklung der BPP und die darauf aufbauenden Produkte im Rahmen des PIL@AP.

Zur Umsetzung der Strategien und Produktideen in ein Softwareprodukt beschreibt der PIL@AP verschiedene Phasen, welche ähnliche Tätigkeiten und Aktivitäten in Prozessen zusammenfassen [DISZ04]. Resultat einer Phase sind bestimmte Ergebnisse (*Deliverables, Outputs*, Artefakte), die in die nächste Phase Eingang finden. Angelehnt an Cooper (siehe auch [Hohm03]) werden die Phasen von so genannten Qualitätstoren (*quality gates,* bei Cooper *stage gates*) eingerahmt, die auf Basis der bisherigen Arbeitsergebnisse eine Risikoabschätzung vornehmen und eine Fortsetzung oder eine Revision der Produktentwicklung beschließen. Die Definition von Phasen resultiert nicht in einem sequentiellen Vorgehen. Vielmehr wird nach den strategischen, marktorientierten Rahmenvorgaben für den Produktumfang in der so genannten *Invent*-Phase in den folgenden Phasen meist stark iterativ vorgegangen und sich somit iterativ dem „Gesamt-produkt" angenähert [GDS+07, S. 8ff.; Gebh07a, S. 4].

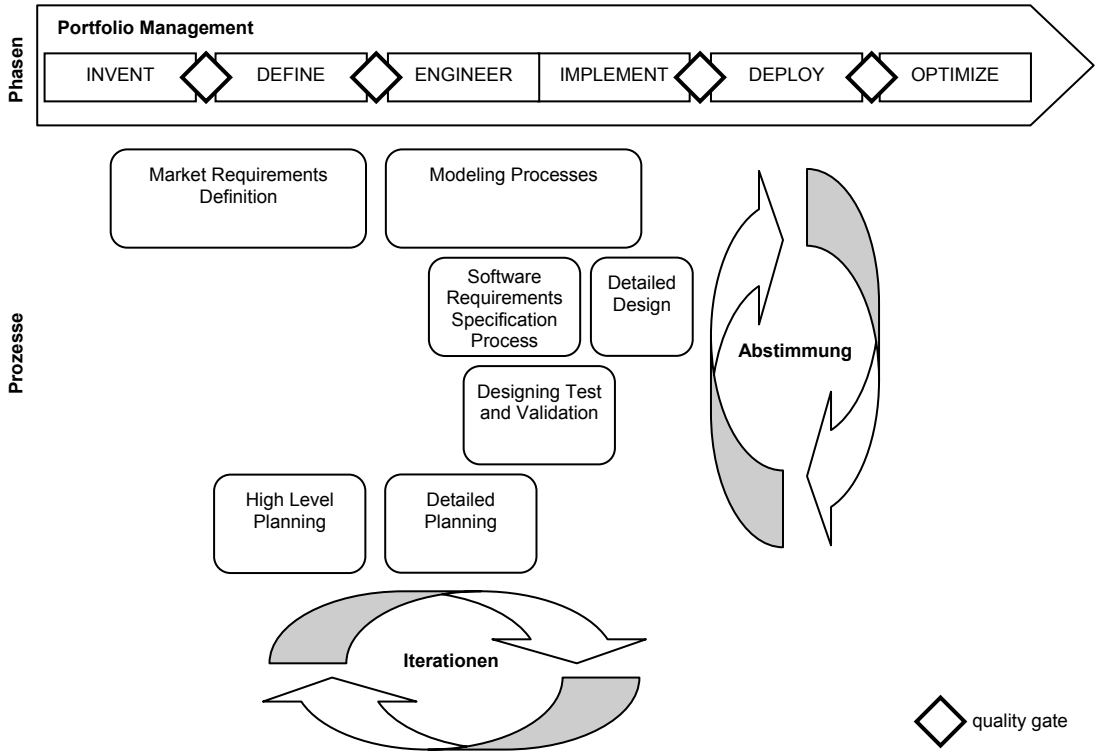

Abbildung 15: Phasen und Prozesse des PIL@AP (Quelle: in Anlehnung an [GDS+07, S. 18]

Die Abfolge der Phasen des PIL@AP und die im weiteren Kontext relevanten Aspekte sind in Abbildung 15[27] illustriert, deren Inhalt im Folgenden beschrieben wird [DISZ04; GDS+07]:

Invent (Ausrichtung): In dieser Phase werden erste Ideen zur Neu- oder Weiterentwicklung von SAP Produkten ausgearbeitet und das Produktportfolio der SAP abgegrenzt und verwaltet.[28] Dies findet auf sehr hoher Planungsebene statt. Es werden also verschiedene Analysen des Marktes und des Verhaltens von Wettbewerbern, Kundenanforderungen und -erwartungen, der SAP-eigenen Strategie etc. berücksichtigt und daraus Produktvorschläge und die Ausrichtung der Produkte festgelegt. Dies geschieht unter Beteiligung der höchsten Managementebenen des Unternehmens.

Define (Abgrenzung/Festlegung): Das Ziel dieser Phase ist die Ableitung und strukturierte Erfassung der Marktanforderungen (*Market Requirements Definition*) und deren Einpassung in die vorhandenen Ressourcen (*High Level Planning*) sowie die Definition der grundlegenden Softwarearchitektur. Auch die technische Machbarkeit der Anforderungen wird einer ersten Abschätzung unterzogen, und erste Festlegungen zur Benutzungsschnittstelle und der Repräsentation der Marktanforderungen in der Softwarestruktur (Architektur) werden gemacht.

Engineer (Konstruktion): Diese Phase ist ein zentraler Bestandteil des ingenieurmäßigen Vorgehens im Rahmen des PIL@AP. Es wird hier die Planung der Software in Form von detaillierten Modellen vorgenommen, entlang derer die Software spezifiziert wird (*Requirements Specification, Modellierungsdokumente*). Die Softwarespezifikationen teilen sich in

- statische Aspekte (Struktur), die während der Modellierungsprozesse (*Modeling Processes*) festgelegt werden und sich in den Modellierungsdokumenten wieder finden und
- dynamische Aspekte (Verhalten), die in einem Spezifikationsprozess für die Anforderungen an die Produkteigenschaften (*Software Requirements Specification Process,* vgl. auch Kapitel 4.4.3.1) festgelegt werden und sich in den Software Requirements Specification Dokumenten (SRS) wieder finden.

Die Spezifikation der statischen Aspekte bildet die Basis für einen „modell-basierten Vertrag“ (*model-based contract*) und die spätere Generierung von Softwareelementen (*proxies*). Die Spezifikation der dynamischen Aspekte spiegelt das Verhalten des Produktes wider, welches aus den Marktanforderungen hergeleitet und weiter detailliert wurde. Basierend auf diesen Spezifikationen, die parallel erarbeitet und eng untereinander abgestimmt werden, wird auch begonnen, die Verifikation (*Designing Test and Validation*) vorzubereiten, um einen Ansatz zu realisieren, der frühest möglichen Test („*test early*“) zulässt.

Implement (Implementierung): In dieser Phase werden auf Grundlage der erstellten (statischen) Modelle möglichst viele Softwarecodeobjekte (*proxies*) generiert. Das entsprechende Verhalten, welches die Geschäftslogik repräsentiert, wird aus den SRS „übernommen“ und in den generierten Objekten ausimplementiert. Da nicht alle Konzepte und Algorithmen für das Verhalten der Software bei den vorangegangenen Tätigkeiten in allen Details festgelegt wurden, wird dies noch vor der Implementierung durchgeführt (*Detailed Design*).

[27] Die Graphik zeigt die in den weiteren Ausführungen genannten Aspekte und enthält nicht alle Prozesse und Einzelheiten.
[28] Wobei eine kontinuierliche Verwaltung des Produktportfolios (*Portfolio Management*) über den gesamten Lebenszyklus stattfindet.

Deploy (Aufstellen): In dieser Phase wird das bei SAP getestete und validierte[29] Produkt erstmals bei ausgewählten Kunden eingesetzt (*ramp-up*). Deren Rückmeldungen und die dabei gemachten Erfahrungen werden ausgewertet und in die Entscheidung zur massenhaften Auslieferung einbezogen. Die Phase dient also auch einer letzten Prüfung, inwieweit die verschiedenen Organisationseinheiten und Partner der SAP (Verkauf, Support, Service, Beratung) für die groß angelegte Produkteinführung aufgestellt sind. Die Rückmeldungen werden weiterhin dazu genutzt, letzte Änderungen und Optimierungen an dem Produkt vorzunehmen.

Optimize (Optimierung): Diese Phase fasst die Tätigkeiten zusammen, die während des Betriebs des Produktes durch den Kunden notwendig sind. Dazu gehören u.a. die Handhabung von Fehlermeldungen und Änderungsanfragen, die Wartung des Produktes und die Verteilung von Korrekturen an die Kunden.

Für die weiteren Ausführungen in diesem Kapitel sind drei Gruppen von Beteiligten des PIL@AP besonders von Bedeutung:

Solution Management (SM): Das Solution Management repräsentiert die Kundensicht. Es ist an der Sammlung der Marktanforderungen beteiligt und überführt diese in die Produktanforderungen in den SRS. Es tritt damit als Stellvertreter des Kunden innerhalb der SAP AG und vor allem gegenüber der Entwicklung (-sorganisation) auf.

Entwicklung (Dev): Die Entwicklung ist für die Umsetzung der betriebswirtschaftlichen Anforderungen in dem Softwareprodukt verantwortlich. Sie gliedert sich in zwei Teile: die Entwicklung des Front-Ends und die Entwicklung der Objekte mit ihrer Logik (*Business Process Platform*).

Operations: Die Operations-Abteilung kontrolliert den Fortschritt der Projekte mittels Überwachung (*monitoring*) und Berichtswesen (*reporting*). Auch die Platzierung und die Überwachung der Qualitätssicherungsmaßnahmen sowie die Bereitstellung der Entwicklungsinfrastruktur (Prozesse, Methoden, Verfahren, Werkzeuge, Best Practices etc.) und geeignete Maßnahmen zur erfolgreichen Projektausführung gehören zu ihrem Verantwortungsbereich.

4.2 Zieldefinition für die Anwendung von Rationale Management im Rahmen des PIL@AP

Grundlage für die Ausarbeitung von Vorschlägen zur Integration von Rationale Management in den Softwareentwicklungsprozess ist eine Zieldefinition, wie sie auch in Kapitel 2.2.2.1 unter den Aufgaben für das Rationale Management beschrieben wurde („Identifikation von Rationale Zielen“). Die Ziele für das Rationale Management im Kontext des PIL@AP wurden zu Beginn der Arbeit aufgestellt. Dies geschah in Abstimmung mit zwei langjährigen Mitarbeitern der SAP AG aus der Operations-Abteilung, die zurzeit vor allem mit der Prozess- und Methodenentwicklung betraut sind.
Die Ziele sind sehr allgemein gehalten und rücken die Bereiche in den Vordergrund, in denen der größte Nutzen aus Sicht der Organisation zu erwarten ist. Sie dienen als Grundlage für die Ausrichtung der vorgeschlagenen Ansätze für die Positionierung von Rationale in einigen (Teil-) Bereichen des PIL@AP. Die Ziele sind:

[29] In diesem Zusammenhang bedeutet ein validiertes Produkt, dass intern bei der SAP AG ein Kunde „simuliert“ wurde, der dieses Produkt in Empfang genommen und dieses eingesetzt hat, wie es die Kunden außerhalb der SAP AG später ebenfalls tun.

1. **Reflektierte Entscheidungsfindung**. Rationale Management soll dazu beitragen, dass Entscheidungen reflektiert und rational gefällt werden und Entscheider darin unterstützt werden. Dieses Ziel resultiert aus der Tatsache, dass bei Entscheidungen häufig nicht alle Sichtweisen, Optionen etc. berücksichtigt werden beziehungsweise diese nicht für alle Entscheider klar sind und offen liegen.
2. **Bessere Nachvollziehbarkeit und höhere Transparenz**. Die Nachvollziehbarkeit von Entscheidungen soll durch den Einsatz von Rationale Management verbessert werden. Die Entscheidungen sollen also für die Ausführenden und Betroffenen transparenter werden, um ein tieferes Verständnis der Entscheidungen herbeizuführen und aufkommende sowie wiederkehrende Spekulationen und Diskussionen über Entscheidungen zu reduzieren.
3. **Unterstützung von Entscheidungsrevision**. Da Entscheidungen häufig von aktuellen Gegebenheiten abhängen und man diese bei Änderung des Kontextes gegebenenfalls revidieren muss, sollte Rationale solche Revisionen unterstützen.
4. **Bewusstsein schaffen**. Die Ideen des Rationale Management sollen sich im Bewusstsein der Mitarbeiter in der Organisation verankern. Dies wird als notwendig erachtet, da der Begriff Rationale (Management) bislang noch nicht in der Organisation verbreitet ist.
5. **Zusätzliche Ressourcen gering halten**. Für eine erste Einführung und Durchführung von Rationale Management sollen möglichst wenig zusätzliche Ressourcen bereitgestellt werden müssen. Dieses Ziel ist mit den vorgenannten auszubalancieren. Werden dann erste positive Auswirkungen mit Rationale Management erzielt, kann dieses Ziel in den Hintergrund treten.

Bei allen Überlegungen zum Einsatz von Rationale Management soll eine möglichst große Zielgruppe innerhalb der Organisation erreicht und die Entscheidungen der mittleren bis höheren Hierarchieebenen einbezogen werden. Hintergrund dafür ist das Ziel, ein Bewusstsein für Rationale Management in der Organisation zu schaffen und die Unterstützung der höheren Hierarchieebenen für die Integration von Rationale Management zu erhalten.

4.3 Kriterien für die Bewertung der Vorschläge zum Rationale Management

Mit der Einführung von Rationale Ansätzen zum Rationale Management wird selbstverständlich immer die Erreichung vorher definierter Ziele bzw. der allgemeinen Ziele von Rationale Management (Kapitel 2.2.1) angestrebt. Ziele für die Anwendung von Rationale Management im Umfeld des PIL@AP – aus denen einzelne für die jeweiligen Vorschläge im Vordergrund stehen – wurden im vorigen Kapitel entwickelt. Diese Ziele kann man als eine Art „funktionaler Anforderungen“ betrachten, die realisiert werden sollen. Zu deren Realisierung können allerdings mehrere alternative Lösungsoptionen existieren, die gegebenenfalls diese Ziele unterschiedlich gut umsetzen. Um zu bewerten, wie gut die Lösungsoptionen sind, werden nun (Qualitäts-) Kriterien aufgestellt (vergleichbar mit nichtfunktionalen Anforderungen). Diese werden u.a. ausgehend von den mit Rationale Management verbundenen Herausforderungen (Kapitel 2.2.2.4) identifiziert, da ein Rationale Ansatz, der sowohl die Ziele adressiert, als auch die Problembereiche reduziert, als ‚gut’ gelten kann. Die Qualität eines Rationale Ansatzes hängt also auch davon ab, inwieweit er eine Lösung für die Problembereiche des Rationale Management erzielt. Die Kriterien kann man nach den beiden Bereichen unterscheiden, zu denen Rationale Ansätze Aussagen tätigen: der Prozessimplementierung und der Repräsentationsform des Rationale.

Wie in Kapitel 2.2.2.4 beschrieben, bilden der Aufwand und die notwendige Disziplin, die zur Durchführung der Aufgaben notwendig sind, eine hohe Schwelle für den Einsatz von Rationale Management. Insbesondere wird hier das Erfassungsproblem genannt, welches mit durch den Aufwand bei der Rationaleerfassung und -entwicklung verursacht wird. Verstärkt wird es zudem dadurch, dass nicht immer klar ist, wozu Rationale erfasst werden soll. Daraus werden folgende zwei Kriterien aufgestellt:

Selektionsunterstützung. Dieses Kriterium bewertet den Grad, in dem der Ansatz die Rolle zur Erfassung von Rationale dabei unterstützt, welches Rationale erfasst werden soll (Unterstützung der Aufgabe „Rationaleidentifikation" und Adressierung des Nutzungsproblems).

Erfassungsaufwand. Dieses Kriterium macht eine Aussage zum Aufwand, den der Rationale Ansatz während der Erfassung und Entwicklung von Rationale (Aufgabe „Rationaleerfassung" und „-entwicklung") erfordert. Im Grunde ist hier natürlich ein geringer Zusatzaufwand für die Erfassung anzustreben. Dies ist jedoch meist mit einem höheren Aufwand bei anderen Aufgaben (z.B. „Rationalenutzung") des Rationale Management verbunden (vgl. Kapitel 4.4.1).

Um die Unterstützung und den Aufwand bei den Aufgaben Rationaleverteilung und -nutzung zu bewerten, dienen die Kriterien:

Abfragemöglichkeit. Über dieses Kriterium ist eine Aussage möglich, wie einfach das benötigte Rationale wieder gefunden werden kann (z.B. über Navigations-, Filter- oder Suchfunktionalität).

Abfrageaufwand/Suchaufwand. Dieses Kriterium bewertet, welcher Aufwand für die Abfrage beziehungsweise die Suche nach relevantem Rationale notwendig ist.

Rekonstruktionsaufwand. Dieses Kriterium bewertet, welcher Aufwand notwendig ist, um Rationale aus seiner Darstellungsform so zu rekonstruieren, dass der Nutzer es auf seine Problemstellung anwenden kann. Rationale z.B., welches mit einem Ansatz erfasst wird, der die Rationaleelemente und ihre Beziehungen nicht ganz explizit, sondern mehr informal darstellt, benötigt mehr Aufwand zur Rekonstruktion als ein Ansatz, der das Rationale mehr formal in einem Schema, durch welches Elemente und Beziehungen klar festgehalten werden, erfasst.

Ein weiteres, häufig in der Literatur genanntes Problem in Verbindung mit Rationale Management ist die Aufdringlichkeit eines Ansatzes (vgl. Tabelle 1, S. 9). Eine Aussage hierzu macht das Kriterium:

Aufdringlichkeit (intrusiveness). Dieses Kriterium bewertet den Grad, inwiefern der Ansatz in den Arbeitsablauf (im Vergleich zum Arbeitsablauf ohne Rationale Management) eingreift und ihn ändert (vgl. Tabelle 1).

Neben diesen Kriterien zur Prozessimplementierung eines Rationale Ansatzes werden nun noch Kriterien aufgestellt, die sich auf die Qualität der Repräsentationsform des Rationale beziehen.
Eine geeignete Darstellungsform kann in diesem Zusammenhang die Vollständigkeit und Konsistenz des Rationale unterstützen, indem sie z.B. fehlendes Rationale leicht identifizierbar und Abhängigkeiten leicht überprüfbar macht. Dies führt zu dem Kriterium:

Vollständigkeit/Konsistenz. Dieses Kriterium bewertet, inwieweit der Rationale Ansatz den Nutzer darin unterstützt, Rationale vollständig beziehungsweise konsistent zu erfassen. So hilft z.B. ein (semi-) formaler Ansatz dem Nutzer stärker, die relevanten Aspekte (z.B. über die Schemaelemente) zu erfassen, als dies Ansätze tun, die Vorgaben nicht so detailliert machen.

Unter den Problemen, mit denen Rationale Management konfrontiert ist, wurden auch die kognitiven Grenzen des Menschen bei der Informationsverarbeitung genannt. Um diesem

etwas entgegen zu wirken, kann versucht werden, die Informationen möglichst strukturiert und übersichtlich darzustellen:

> *Übersichtlichkeit*. Dieses Kriterium macht eine Aussage zur Übersichtlichkeit des Rationale, wenn es dem Benutzer präsentiert wird, beziehungsweise wenn er auf dieses zugreifen muss.

Je nachdem welche und wie ausführlich die Rationaleelemente erfasst werden und der Kontext einer Entscheidung (dadurch) beschrieben wird, kann sich deren Verständlichkeit unterscheiden (z.B. nach einer länger verstrichenen Zeitdauer oder für Interessenvertreter, die nicht an der Entscheidung beteiligt waren). Die Verständlichkeit wird deshalb auch als Kriterium für eine Bewertung herangezogen:

> *Verständlichkeit*. Dieses Kriterium bewertet, wie „einfach" das Rationale für den Nutzer verständlich ist, der dieses z.B. nicht selbst erfasst oder länger nicht mit diesem konfrontiert wurde. Teils ist (semi-) formales Rationale nicht einfach verständlich, wenn man mit dem Ansatz z.B. nicht vertraut ist oder nicht Domänenexperte ist, da nur gewisse Elemente der Entscheidung erfasst werden. Bei informaler Erfassung dagegen könnte man flexibler und gegebenenfalls verständlicher beschreiben.

Wichtig bei der Einführung von Rationale Management ist auch der Aufwand, der benötigt wird, um diesen Ansatz in Prozesse oder Werkzeuge zu integrieren. Deshalb sollte auch dieser Implementierungsaufwand berücksichtigt werden:

> *Implementierungsaufwand.* Mit diesem Kriterium wird der Aufwand bewertet, der notwendig ist, um Rationale Management in einem bestimmten Kontext umzusetzen (z.B. in Prozessen oder Werkzeugen).

Mit Sicherheit können noch weitere Kriterien identifiziert werden, die bei der Bewertung von Rationale Ansätzen sinnvoll sind. Allerdings werden in der vorliegenden Arbeit die genannten Kriterien als eine Grundmenge zur Evaluation der vorgeschlagenen Rationale Ansätze herangezogen.

4.4 Anwendungsmöglichkeiten von Rationale Management im Rahmen des PIL@AP

Im Folgenden werden drei Möglichkeiten diskutiert, wie Rationale Management in einem ersten Schritt bei der SAP AG im Rahmen des PIL@AP eingeführt werden kann. Die Auswahl dieser drei Vorschläge wurde von den Überlegungen geleitet, dass in der Organisation die Ideen des Rationale Management erst noch Verbreitung finden müssen. Vor diesem Hintergrund sollen bestimmte Gruppen in der Organisation für Rationale Management sensibilisiert werden und dadurch ihre Unterstützung anbieten. Deshalb wurde sich für die unterbreiteten Vorschläge einmal auf Entscheidungsbereiche konzentriert, die höhere Hierarchieebenen betreffen (Kapitel 4.4.1) und strategische Bedeutung haben (Kapitel 4.4.2). Gleichzeitig sollten die Entscheidungen eine große Zielgruppe adressieren, um die Verbreitung der Ideen voranzutreiben (Kapitel 4.4.1 und Kapitel 4.4.2). Ein weiteres Auswahlkriterium war die Demonstration des Nutzens für bestimmte Verantwortlichkeitsbereiche, was besonders bei dem Vorschlag in Kapitel 4.4.3 deutlich wird.
Da die Arbeit in der Operations-Abteilung verfasst wurde, bestand bei den Mitarbeitern ein Überblick über gewisse Problembereiche (*pain points*), aus denen dann die konkreten Möglichkeiten für die Vorschläge nach obigen Kriterien selektiert wurden.

4.4.1 Kommunikation von Projektlenkungsentscheidungen

Bei der SAP AG ist die Produktentwicklung nach einem „Release Master Plan“ ausgerichtet, welcher die Orientierung der Organisationseinheiten in Hinblick auf die Produktfertigstellung vorgibt. Die einzelnen Beiträge für das Endprodukt kommen im Allgemeinen aus einer **Projektorganisation** mit verschiedenen Aggregationsleveln. Auf oberster Ebene finden sich **Programme**, die gegebenenfalls Subprogramme und diese gegebenenfalls wiederum **Projekte** beinhalten können. Die Programme liefern den **Kontext**, in dem sich die Subprogramme und Projekte mit ihren Funktionseinheiten und Themenbereichen beschäftigen (meist **bezogen auf ein Produkt**). Häufig sind die Programme quer zu einer **Linienorganisation** angeordnet, welche die Funktionsbereiche der Organisation – wie z.B. Solution Management, Entwicklung und Vertrieb – abdeckt. Somit ergibt sich eine Art **Matrixorganisation**.
Die Kommunikationswege in diesen Programmen sind recht komplex, und die Anzahl der Mitarbeiter, die an solchen Programmen beteiligt sind, geht von ca. 50 Personen bis an die 1000. Deshalb sind effiziente **Kommunikationsinstrumente** in diesem Zusammenhang von großer Bedeutung. Diese Kommunikationsinstrumente müssen u.a. auch **Projektlenkungs-entscheidungen** von Verantwortlichen kommunizieren und dafür sorgen, dass diese in der Organisation umgesetzt werden. Eine Möglichkeit dies zu tun, wird in diesem Kapitel betrachtet, und es wird untersucht, wie sich hier Ideen des Rationale Management umsetzten lassen.
In Kapitel 2.2.1 wurde festgestellt, dass Rationale die Zusammenarbeit in Teams unterstützt, indem es die Arbeit anderer Teammitglieder **versteh- und nachvollziehbar** macht. Dies geschieht durch die Begründung von Entscheidungen über das Herstellen von Beziehungen zu wichtigen Elementen einer Entscheidung. Hiermit wird der **Weg transparent**, der zu dieser Entscheidung geführt hat. Rationale kann damit eine Art **Verfolgbarkeit** (*traceability*) zwischen verschieden Entscheidungen und Entscheidungen und deren Elementen herstellen, welche in einem ähnlichen Netzwerk resultiert wie z.B. die Verfolgbarkeit von Anforderungen zu Testfällen.
Diese Aspekte adressieren auch die in Kapitel 4.2 aufgestellten Ziele, welche mit den in dieser Arbeit vorgeschlagenen Maßnahmen erreicht werden sollen. In diesem Sinne ist ein durchgängiges Erreichen einer Entscheidungsnachverfolgbarkeit mit zugehöriger Entscheidungsbegründung bereits ein wesentlicher Schritt hin zu einem verbesserten und transparenteren Kommunikationsinstrument. Dies sollte in einem Verbesserungsvorschlag für ein Kommunikationsinstrument berücksichtigt werden.
Es muss allerdings festgestellt werden, dass ein Kommunikationselement von Projektlen-kungsentscheidungen durchaus **politischen Einflüssen** ausgesetzt ist, sowohl aus Unter-nehmenssicht als auch aus individueller Sicht. Es kann also gewünscht sein, nicht alle Information über Entscheidungen zu kommunizieren. Die Gründe hierfür können vielfältig sein (vgl. auch [DMMP06b, S. 21]), denn die ausführliche und nachvollziehbare Dokumentation von Entscheidungen kann z.B. nicht fundiert getroffene Entscheidungen beziehungsweise Fehlentscheidungen enthüllen und den Einzelnen angreifbar machen (dies gilt auch für rückblickend ungünstige Entscheidungen). Außerdem gibt es gewisse Entscheidungen, die innerhalb einer bestimmten Gruppe vertraulich behandelt werden müssen.
Vor diesem Hintergrund soll die Anwendung von Rationale auf die Kommunikation von Projektlenkungsentscheidungen in einem Subprogramm bei der SAP AG untersucht werden, welches in den PIL@AP eingebettet ist. Programmplanung, -monitoring, -kontrolle und -steuerung wird von einer zentralen Operations-Abteilung koordiniert. Um die Steuerung des Programms auf Basis des Programmmonitoring und der Programmkontrolle vorzunehmen, gibt es festgelegte Treffen der Verantwortlichen. In diesen werden die wichtigen Entscheidungen zur Programmsteuerung getroffen, welche dann in die Organisation kommuniziert werden müssen.

Den Anlass für diese Untersuchung gab die Beobachtung zweier langjähriger SAP Mitarbeiter aus dem Umfeld der Operations-Abteilung, die zurzeit schwerpunktmäßig mit Prozess- und Methodenentwicklung (*Process Governance*) betraut sind. Nach Angaben dieser besteht in der Entscheidungskommunikation weiterhin ein Verbesserungsbedarf, und die Anwendung von Elementen des Rationale Management könnte hier gegebenenfalls erfolgreich eingesetzt werden. Um an Informationen über diesen Bereich zu kommen, wurde in (Prozess-) Unterlagen recherchiert und Gespräche mit den beiden Mitarbeitern und der verantwortlichen Rolle für die Entscheidungskommunikation (im Folgenden als „Entscheidungs-kommunikator" bezeichnet) geführt.

4.4.1.1 IST-Situation des Decision Log

a) Decision Log – Lösung ohne Rationale

Ursprünglich wurden die Entscheidungen der Treffen zur Projektlenkung über persönliche Kanäle kommuniziert. Diese Vorgehensweise hat die Erreichung der Adressaten nicht immer gewährleistet und oft zu zeitlichen Verzögerungen geführt. Um diesen Unzulänglichkeiten zu begegnen, wurde damit begonnen, einen so genannten „Push-Mechanismus" zu implementieren, der die Umsetzenden mit den Entscheidungen versorgen soll. Hierzu wurde als Maßnahme eine Informations-E-Mail eingeführt, die über die bedeutenden Entscheidungen informiert. Diese E-Mail enthält – sehr kurz dargestellt – die getroffenen Entscheidungen ohne Begründungen und wird über einen Verteiler an die Entwicklungsorganisation versand.
Um die reine E-Mail Lösung zu ergänzen, wurde eine Informationsseite im Intranet aufgesetzt, die einen so genannten **Decision Log** [DeLo] für die Entscheidungen beinhaltet. Dieser soll die Entscheidungskommunikation zusätzlich unterstützen, indem er kurz und prägnant die getroffenen Entscheidungen auflistet. Enthaltene Informationen zu einer Entscheidung in diesem Decision Log sind:

Date: Das Datum, an dem die Entscheidung getroffen wurde.
Topic: Das Thema, das von der Entscheidung betroffen war (z.B. Testaktivitäten, Zeitpläne, Meilensteine, Prioritäten von Aktivitäten und Maßnahmen etc.).
Decisions: Die Beschreibung der Entscheidung, die getroffen wurde.
More Information: Verweise auf verschiedene Informationen (nicht weiter spezifiziert).

Damit liefert der Decision Log eine integrierte Darstellung der Entscheidungshistorie in chronologischer Ordnung über einen längeren Zeitraum hinweg und dies für die wichtigsten Entscheidungen aus den einbezogenen Treffen. Ergänzt wird die Intranetseite des Decision Log mit einer Liste der Informations-E-Mails (Historie), die Verweise auf die E-Mails enthält. In Abbildung 16 ist das Konzept des Decision Log und der darin vorhandenen Informationen schematisch dargestellt.

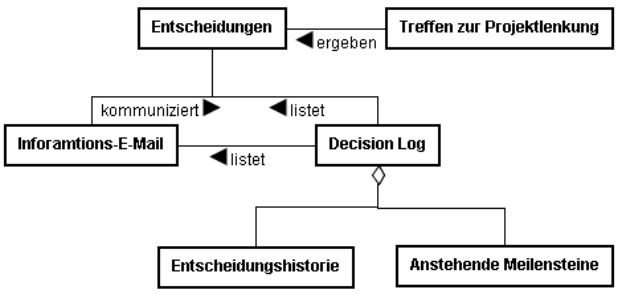

Abbildung 16: Konzept des Decision Log (UML-Notation)

Der Decision Log wird von einem Mitarbeiter der Operations-Abteilung gepflegt (Entscheidungskommunikator), der auch als Ansprechpartner bei Rückfragen zur Verfügung

steht. Die Aufgabe und die Arbeitschritte des Entscheidungskommunikators sind in einem User Task und Use Cases in Tabelle 12, Tabelle 13 und Tabelle 14 dargestellt (siehe Anhang, S. 84f.). Diese Darstellungsform wurde an dieser Stelle gewählt, da sie sich zur Aufgabenbeschreibung und Erfassung des Kontextes einer Arbeitsumgebung in einer Domäne bewährt hat. Das Use Case Diagramm in Abbildung 17 zeigt den gesamten Kontext des Decision Log. Neben den User Tasks und den Use Cases in den genannten Tabellen, die zur Verteilung der Informationen beitragen, wird in diesem Diagramm auch die Erzeugung der Informationen („Treffen von Projektlenkungsentscheidungen") und der „Konsum" der Informationen („Umsetzung von Projektlenkungsentscheidungen") gezeigt.

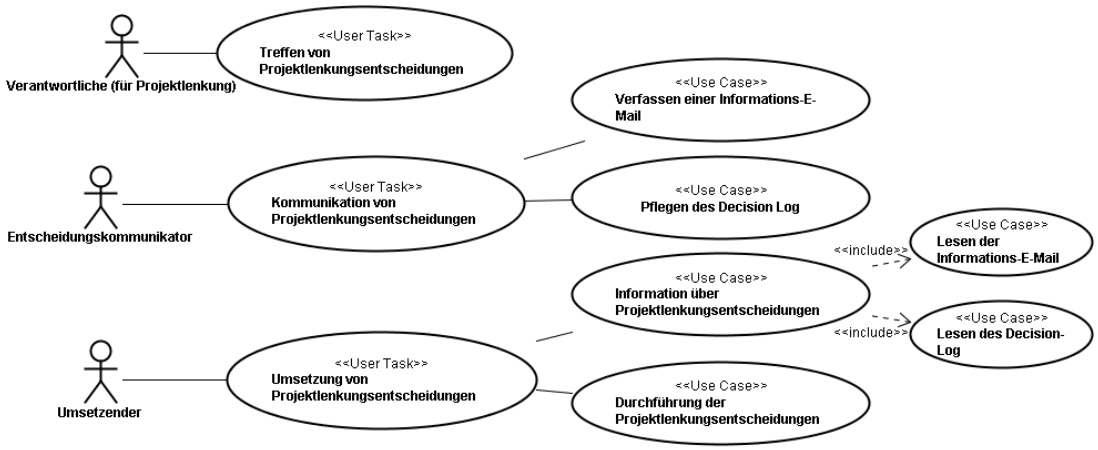

Abbildung 17: Kontext der Projektlenkungsentscheidungen (Use Case Diagramm, UML-Notation)

b) Eintrag im Decision Log – Beispiel ohne Rationale

Um die Art der Entscheidungskommunikation darzustellen, wird im Folgenden ein Beispiel gegeben, welches die Kommunikation einer Entscheidung mit dem Ziel der Verbesserung der Anforderungsspezifikation (SRS) mittels Anforderungsinspektionen [30] (*Requirements Inspections*) beinhaltet. Die Kommunikation dieser Entscheidung über den Decision Log ist in Tabelle 7 dargestellt.

General Decisions (Program Management (Mgmt) Meetings)			
Date	**Topic**	**Decision**	**More Information**
2007-02-08	SRS Quality Improvement	The Mgmt Team agreed, that Requirements Inspections have to be conducted in all Sub-Programs with PRIO 1. 50% of the „Review version" SRS (status) should be covered until end of March. This has to be communicated to all the authors of the SRS to schedule the inspection meetings with all required participants (SM, Front-end Dev, Back-End Dev)	n/a

Tabelle 7: Eintrag im Decision Log – ohne Rationale

[30] Eine Inspektion untersucht dabei eine SRS, welche die gewünschten Charakteristiken der späteren Software als Anforderungen (Anforderungen an Produkteigenschaften) erfasst, in einem formal aufgesetzten Prozess. Dieser beinhaltet bei der SAP AG [GeDi06]

- dezidierte Rollen (Moderator, SRS-Autor, Vertreter der Bereiche, die mit der SRS arbeiten müssen),
- eine kontrollierte Inspektionsrate innerhalb der angesetzten Zeit (normalerweise sollen eine bestimmte Anzahl von Anforderungen in einer gewissen Zeit inspiziert werden),
- eine Checkliste mit formalen Kriterien, auf die die Anforderungen überprüft werden (*Design-independence, Unambiguity, Consistency, Testability, Logical correctness, Comprehensibility, Formal conformity*) und
- eine Liste mit den gefundenen Fehlern.

In der Informations-E-Mail stehen zu einigen Entscheidungen noch ausführlichere Informationen als im Decision Log (Tabelle 7), allerdings nicht explizit mit dem Eintrag der Entscheidung im Decision Log verlinkt. In o.g. Beispiel könnte sich darin z.B. ein Verweis auf die Einführungsunterlagen (Prozessbeschreibungen) zu den Inspektionstreffen befinden. Dort wären u.a. Informationen über die Ziele, Charakteristiken, Rollen, Verantwortlichkeiten, das Vorgehen und die formalen Kriterien beziehungsweise Checklisten der Inspektion enthalten.
Im obigen Beispiel (Tabelle 7) könnte ein Adressat der Entscheidung der Meinung sein, dass die Methode der Inspektion nicht hilft, Anforderungsdokumente zu verbessern und auf einen abschließenden Stand für die Übergabe an die Entwicklung zu bringen. Hier bietet die Darstellung des Weges zur Entscheidung (überlegte Alternativen, Ergebnisse und Feedback evtl. durchgeführter Piloten beziehungsweise erhobener Daten) die Möglichkeit, dennoch zu einer Einsicht zu gelangen. Diese Offenlegungen würden auch eine Einordnung der Entscheidung nach dem Vergehen einer längern Zeitspanne einfacher machen.

c) Abgeleitete Probleme und anvisierte Ziele

Untersucht man das beschriebene Kommunikationsinstrument unter Rationale Gesichtspunkten, fehlen hier die wesentlichen Aspekte einer transparenten Entscheidungskommunikation. Denn die bisherige Lösung bildet zwar eine geeignete Maßnahme, um die Adressaten mit der getroffenen Entscheidung zu erreichen, allerdings liefert sie keine Nachvollziehbarkeit und Begründung der Entscheidungen. Hiermit sind zwei **Probleme** verbunden:

- einerseits ein Einsichts- beziehungsweise Akzeptanzproblem und
- andererseits das Problem, dass man diese Entscheidung nach einiger Zeit nicht mehr in ihren Kontext einordnen und damit nachvollziehen und verstehen kann.

Beide Probleme betreffen die Adressaten der Entscheidung in Form der Umsetzenden. Das zweite Problem betrifft auch die Entscheidungsträger, die aus verschiedenen Gründen wieder auf den Entscheidungskontext zugreifen müssen.

Aufgrund der Probleme stehen für die Überlegungen zur Integration von Rationale folgende zu erreichende **Ziele** (aus der Zieldefinition in Kapitel 4.2) im Vordergrund:

- Transparenz und Nachvollziehbarkeit der Entscheidungen (dies kann zur Behebung beider Probleme beitragen)
- Schaffung eines Bewusstseins für Rationale (als Zusatzziel)
- Geringe zusätzliche Ressourcen

Ein weiteres Ziel für diesen Vorschlag resultiert aus den zu Anfang des Kapitels diskutierten Entscheidungsgegenständen, deren Begründungen beziehungsweise Kontexte nicht unbedingt für jeden zugänglich sein sollten:

- Es soll möglich sein, den Zugriff auf das Rationale hinter den Entscheidungen auf bestimmte Gruppen einzuschränken

Inwieweit diese Ziele erreicht werden, wird rückblickend in Kapitel 4.5 betrachtet.

4.4.1.2 SOLL-Situation des Decision Log

a) Decision Log – Vorschlag für Aufnahme von Rationale (Diskussion)

Im Folgenden wird ein Vorschlag diskutiert, der die genannten Probleme beheben und die aufgestellten Ziele umsetzen soll. Zusätzlich zur Diskussion werden die bei der Erarbeitung des Vorschlags angestellten Überlegungen und die getroffenen Entscheidungen, d.h. das Rationale, in verschiedenen Darstellungsformen für Rationale repräsentiert (vgl. Abbildung 18 auf S. 65, Tabelle 3 auf S. 31, Abbildung 21 auf S. 86), um dem Leser einen Eindruck von den verschiedenen Dokumentationsmöglichkeiten zu geben.

Um weitere Überlegungen zur Anreicherung des Decision Log mit Rationale anstellen zu können, muss untersucht werden, woher man die Entscheidungsbegründungen, die ja zumindest implizit vorhanden sein müssen, erhalten kann. Die Entscheidungen in den Treffen werden auf Basis von konkreten Problemstellungen getroffen, die in den Treffen präsentiert und diskutiert werden. In diesen Präsentationen sind die Gründe beziehungsweise Ursachen für die Entscheidungen in einer gewissen Weise dokumentiert (meist Microsoft PowerPoint Dokumente beziehungsweise in den Protokollen der Treffen; meist Microsoft Word Dokumente).
In den Protokollen findet man allerdings häufig nur die Agenda sowie die resultierenden Action Items der Treffen. Im ungünstigsten Falle enthalten diese Dokumente also nur eine kurze Darstellung des Problems. Bestenfalls umfassen die Dokumente aber bereits folgende Informationen (in Teilen), da diese in einigen Vorlagen (*templates*) für Präsentationen vorgegeben sind:

- Problem-/Fragestellung *(Problem/Issue to be resolved)*
- Alternativen beziehungsweise nur *eine* vorgeschlagene Option *(Recommended Approach)*
- Unterstützende und verworfene Argumente *(Benefits, Impact of the approach, Trade-offs, Implications, Risks)*
- vorgeschlagenes Vorgehen bei Wahl einer Alternative *(Proposed Action Plan)*
- getroffene Entscheidung *(Decision taken)*

Bei Entscheidungen zu bereits durchgeführten Steuerungs- und Lenkungsmaßnahmen (weiterer Einsatz, Korrektur, Absetzen) sind zumeist folgende Punkte enthalten:

- Durchgeführte Lenkungsmaßnahme mit Zielen
- Feststellungen *(Key Findings)* mit positiven und negativen Aspekten *(Pain Points)*
- Schlussfolgerungen
- (Verbesserungs-/Änderungs-) Vorschläge

Werden diese Informationen (die auf einer zentralen Ablage liegen) über den Decision Log zugänglich gemacht, wäre bereits ein großer Schritt hin zu einem transparenteren Entscheidungsprozess gewährleistet.
Auf der Grundlage, dass die oben beschriebenen Informationen (aus den Vorlagen) vorhanden sind und offen gelegt werden sollen, ist erst einmal zu entscheiden, wie das Rationale repräsentiert werden soll. Hierfür stehen zwei alternative Möglichkeiten zur Verfügung:

i. Darstellung durch die bereits vorhandenen Dokumente (ohne Änderung)
ii. Darstellung in einer Vorlage (*template*), die eigens zur Darstellung der Rationaleelemente entwickelt wird (formaler als die erste Alternative)

Die zu treffende Entscheidung könnte man als Frage/Problem folgendermaßen formulieren:

Wie soll das Rationale zur Verfügung gestellt werden?

Wie in Kapitel 4.3 angemerkt, werden die Ansätze und ihre alternativen Umsetzungsmöglichkeiten an ihrer **Zielerreichung** und den aufgestellten Kriterien gemessen und abgewogen.
Alternative (i) setzt hierbei verstärkt das Ziel um, mit geringen zusätzlichen Ressourcen auszukommen. Aber auch die weiteren in Abschnitt 4.4.1.1c) genannten Ziele werden erreicht. Allerdings ist Alternative (ii) bei der Nachvollziehbarkeit und Transparenz deshalb im Vorteil, da die Rationaleelemente expliziter dargestellt und je nach Vorlage (*template*) auch miteinander in eine explizite, bedeutungstragende Beziehung gesetzt werden könnten. Dadurch würde auch die Idee von Rationale umfassender in das Bewusstsein der involvierten Personen aufgenommen.
Kriterien für die Bewertung von Rationale Ansätzen wurden in Kapitel 4.3 aufgestellt und werden nun ebenfalls zur Auswahl der Alternativen herangezogen. Im Folgenden werden die Alternativen in Bezug auf die Kriterien diskutiert, und in Abbildung 18 sind die Bewertungen

der Alternativen bezüglich der aufgestellten Kriterien zusätzlich in der graphischen QOC-Notation dargestellt, das Teilproblem („Wie soll das Rationale zur Verfügung gestellt werden“) in tabellarischer Form ebenfalls in Tabelle 3 (S. 31). In Abbildung 21 (S. 86) wird die Diskussionserfassung mittels des Compendium Werkzeugs in dessen IBIS Notation gezeigt.

Selektionsunterstützung. Keine der Alternativen unterstützt den Entscheidungskommunikator darin, welches Rationale er erfassen soll. Allerdings ist dieses Kriterium auch nicht zur Bewertung der Darstellungsform eines Rationale Ansatzes geeignet, sondern eher für die Prozessimplementierung des Rationale Ansatzes. Diesbezüglich bietet der vorgeschlagene Ansatz aber auch keine Selektionsunterstützung.

Erfassungsaufwand. Der Aufwand für die Erfassung des Rationale ist für beide Alternativen ungefähr gleich. Alternative (ii) wird nur höheren Aufwand bedeuten, wenn mehr (Rationale-) Elemente zur Verfügung gestellt werden sollen als in den Dokumenten ohnehin vorhanden sind.
Allerdings soll hier auch der Aufwand für die Rationaleentwicklung mit einbezogen werden, die eng mit der Rationaleerfassung (vgl. Kapitel 2.2.2.1) zusammenhängt. Hier ist der Aufwand für Alternative (ii) höher, da das Rationale aus den ursprünglichen Dokumenten extrahiert und in die Vorlage eingepflegt werden muss. In Hinblick auf das Streben nach einem Ansatz mit geringem Aufwand ist bezüglich dieses Kriteriums Alternative (ii) deshalb vorzuziehen.

Aufdringlichkeit. Durch die Wiederverwendung bestehender, unveränderter Dokumente als Rationalerepräsentation (Alternative (i)) würde der vorgeschlagene Ansatz nicht gravierend in den Arbeitsablauf eingreifen, der ohnehin zur Pflege des Decision Log beziehungsweise zur Verfassung der Dokumente notwendig ist (nicht aufdringlich). Lediglich eine Verlinkung auf die abgelegten Dokumente und gegebenenfalls zwischen verwandten Entscheidungen und auf die Informations-E-Mail ist zusätzlich notwendig, aber nicht mit großem Aufwand verbunden. Daher wäre Alternative (i) bezüglich dieses Kriteriums vorzuziehen.

Abfragemöglichkeit. Die Alternativen können in der bisher betrachteten Allgemeinheit nicht gegen dieses Kriterium bewertet werden. Denn die reine Darstellung sagt erst einmal nichts über die Abfragemöglichkeit aus. Eine Bewertung kann hier erst stattfinden, wenn die technische Umsetzung der Repräsentation klar ist. Dies wird später noch betrachtet.

Abfrageaufwand/Suchaufwand. Bezüglich dieses Kriteriums ist Alternative (ii) als günstiger einzustufen, da hier das Rationale strukturierter und kompakter zusammengestellt ist. Bei Alternative (i) kann es nötig sein, in größeren Dokumenten suchen zu müssen, in denen das Rationale nicht derart explizit dargestellt ist, wie in Alternative (ii).
Hier lässt sich allgemein feststellen, dass mit einer informalen Darstellung des Rationale der Aufwand für das Auffinden gewünschter Information steigt (der Aufwand zur Rationaleerfassung und -entwicklung ist dafür aber geringer).

Verständlichkeit. Die Verständlichkeit des Rationale kann leiden, falls es nur kurz in wenigen Rationaleelementen in einer Vorlage repräsentiert werden muss. In den vorhandenen Dokumenten kann der Kontext, in dem die Entscheidung gefällt wurde, gegebenenfalls besser dargestellt und verständlich gemacht werden, da hier z.B. auch die graphische Repräsentationen von Zahlenmaterial möglich ist. Dies hat im Umfeld des Projektmonitoring und -controlling meist eine große Bedeutung, und auf dieser Basis werden die meisten Projektlenkungsentscheidungen getroffen.

Vollständigkeit/Konsistenz. Alternative (i) wählt eine informale Darstellungsform. Die Konsequenz dieser Eigenschaften ist allerdings, dass keine Vollständigkeit der

Entscheidungsdokumentation gewährleistet sein kann. Es kann vorkommen, dass für Entscheidungen keine weiteren Informationen zur Verfügung gestellt werden beziehungsweise, dass die zur Verfügung gestellte Dokumentation nicht die gewünschten Informationen zur Begründung enthält. Hier könnte man versuchen, die Protokolle der Treffen aussagekräftiger zu gestalten, worauf später nochmals kurz eingegangen wird.
Um eine gewisse Vollständigkeit des Rationale zu garantieren, ist die Vorgabe von Schemata mit vordefinierten Elementen meist besser geeignet (Alternative (ii)) als das Vertrauen, dass die verwendeten, unveränderten Dokumente die relevanten Gesichtspunkte auch wirklich enthalten (Alternative (i)). Über die Erfassung des Rationale in einem Schema kann schneller eine Kontrolle der Konsistenz unter den einzelnen Entscheidungen erreicht werden (Alternative (ii)).

Übersichtlichkeit. Die Übersichtlichkeit in einer strukturierten Vorlage ist höher als in einem gegebenenfalls recht unstrukturierten Dokument. Allerdings wurde weiter oben bereits erwähnt, dass auch Vorlagen für Dokumente (Präsentationen) existieren, die eine gewisse Strukturiertheit aufweisen. Dies würde auch bei Alternative (i) in gewissem Maße zur Realisierung des Kriteriums führen.

Rekonstruktionsaufwand. Der Rekonstruktionsaufwand kann aufgrund des Umfangs der Dokumente in Alternative (i) höher sein. Allerdings kann in Alternative (ii) der Kontext verloren gehen, da man sich in den Vorlagen in der Regel kurz fassen muss und ein starres Schema zur Erfassung des Rationale vorgegeben wird (vgl. Verständlichkeit).

Erfassungs-/Entwicklungs-/Abfrage-/Such-/Rekonstruktionsaufwand. In Bezug auf diese Kriterien muss bei Einführung eines Rationale Ansatzes meist eine grundsätzliche Entscheidung getroffen werden. Denn erfahrungsgemäß verhält sich der Aufwand Erfassung/Entwicklung gegen Rekonstruktion/Abfrage von Rationale gegenläufig. Eine Entscheidung hängt mit Sicherheit von der Rekonstruktionshäufigkeit ab. Denn müssen viele Projektbeteiligte darauf zugreifen und dies rekonstruieren, steigt hier in der Summe der **Aufwand** im Gegensatz zu einem **einmaligen** Mehraufwand bei der Erfassung. Diesen hat normalerweise nur eine Rolle zu tragen. Im Zusammenhang mit dem Decision Log wird die Zahl der Nutzer recht hoch sein. Auf der anderen Seite ist noch unklar, inwieweit diese das Rationale wirklich verwenden und auch die Möglichkeit, in die Erfassung/Entwicklung des Rationale mehr Zeit zu investieren, ist momentan nicht gegeben (vgl. auch Ziel 5 in Kapitel 4.2). Diese Tatsachen legen also die Verwendung der vorhandenen, unveränderten Dokumente als Rationalerepräsentation nahe.

Mit der Frage nach der Repräsentationsform (Wie?) verknüpft ist die Frage, wo man das Rationale dann zur Verfügung stellt. Das Problem soll wie folgt formuliert werden:

Wo soll das Rationale zur Verfügung gestellt werden?

Geht man davon aus, dass für das „Wie“ Alternative (i) (Verwendung der vorhandenen, unveränderten Dokumente als Rationalerepräsentation) als Lösung gewählt wird, kann man für das „Wo“ zwei Alternativen betrachten:

1. Tabelle des Decision Log unverändert lassen und in der Spalte „*More Information*“ Verweise auf Dokumente zur Verfügung stellen, die das Rationale enthalten.[31]
2. Zur Tabelle des Decision Log eine weitere Spalte „*Rationale*“ hinzufügen, die Verweise auf Dokumente zur Verfügung stellt, die das Rationale enthalten.

[31] Bislang wurde die Spalte „More Information“ so gut wie nicht verwendet. Falls sie verwendet wurde, wurde nicht auf Dokumente verwiesen, aus denen man Rationale hätte rekonstruieren können. Beispiele für in der Vergangenheit verlinkte Dokumente waren geänderte Zeitpläne oder Schulungsunterlagen.

Wird Alternative (ii) für das „Wie“ gewählt, gibt es ebenfalls zwei alternative Möglichkeiten für das „Wo“ einer Umsetzung in Zusammenhang mit dem Decision Log:

3. Tabelle, die den Decision Log beinhaltet, mit weiteren Spalten ergänzen. In diese würde dann das Rationale aufgenommen werden.
4. Darstellung des Rationale in einer Vorlage, auf die aus der Tabelle heraus verwiesen wird.

Alternative (1) und (2) unterscheiden sich kaum. Falls die Wahl zwischen diesen beiden Alternativen allerdings notwendig ist, da sich für die vorhandenen Dokumente als Rationalerepräsentation entschieden werden sollte (Alternative (i) bei „Wie“), könnte man zwei Argumente für Alternative (2) anbringen:

> In Alternative (2) wird explizit der Begriff „Rationale“ verwendet. Dadurch drängt er stärker in das Bewusstsein der Interessenvertreter, was von Vorteil sein kann, wenn man Rationale auch in anderen Bereichen verstärkt nutzen und ein Bewusstsein dafür schaffen möchte (vgl. aufgestellte Ziele).
> Weiterhin ist eine klarere Trennung zwischen Zusatzinformation und Rationale zu einer Entscheidung vorhanden.

Im Folgenden sind nur die Kriterien (Kapitel 4.3) angegeben, die sich nicht neutral gegenüber den Alternativen verhalten und relevant für die Fragestellung sind:

> *Übersichtlichkeit*. Die Übersichtlichkeit des Decision Log ist für Alternativen (1), (2) und (4) höher, da mit weniger Spalten ausgekommen wird.
>
> Gegen Alternative (3) kann man außerdem folgende zwei Argumente anführen:
> Es kann sein, dass einige Mitarbeiter sich nicht für das Rationale hinter der Entscheidung interessieren, sei es aus Zeitmangel, Desinteresse oder dass sie die Entscheidung ohnehin unterstützen. Für diese Zielgruppe ist die Seite ohne zusätzliche Tabellenspalten angenehmer und schneller zu überschauen.
> Betrachtet man Alternative (3), müsste man auch sicherstellen, dass die Rationaleeinträge pro Spalte der Tabelle die bisherigen Einträge des Decision Log nicht in ihrem Umfang überschreiten (Gleichartigkeit der Einträge). Das Rationale müsste somit sinnvoll verdichtet werden, was nicht immer einfach möglich ist und mit zusätzlichem Aufwand verbunden wäre, sollte es nicht bereits verdichtet vorliegen.
>
> *Abfragemöglichkeit*. Das Kriterium *Abfragemöglichkeit* kann über alle Alternativen hinweg betrachtet werden und wird hier noch evaluiert. Die Navigation zwischen den Entscheidungen ist für alle Alternativen gegeben. Ausgehend von den Entscheidungen in der Tabelle kommt man zum Rationale, sei es über die Verweise oder über den direkten Eintrag in der Tabelle. Weiterhin ist in allen Alternativen gewährleistet, dass eine Suchfunktion über das Intranet besteht. Im Fall der Verweise auf die Dokumente (Alternativen (1), (2), (3)) werden diese über die Links indexiert, im Fall der Tabellenspalten (Alternative (3)) über die Intranetseite selbst.

In den Zielen wurde auch festgelegt, dass bestimmte Entscheidungen nicht für alle Decision Log Nutzer zugänglich sein sollten (**zielgruppenspezifische Berechtigung**). Dies bedeutet zwar, dass für einzelne Nutzer das Rationale nicht zu allen Entscheidungen zur Verfügung steht, im Gesamten das Rationale allerdings erfasst und vernetzt wird und auch berechtigungsspezifisch zugänglich ist. Bis auf Alternative (3) (zusätzliche Rationale Spalten in der Tabelle) würden alle weiteren Alternativen über die Verlinkung auf externe Dokumente dies unterstützen. Denn allen Dokumenten bei der SAP AG werden über deren Ablageort Zugriffsrechte zugeordnet. Bei Zugriff auf die Dokumente über das Intranet (Link) werden diese Zugriffsrechte entsprechend der anfragenden Rolle überprüft. Bei Nutzung der Alternativen (1), (2) und (4) erhält man also über die Ablage der Dokumente die gewünschte Zugriffskontrolle. Bei Alternative (3) wäre dies nicht ohne weiteres möglich, da auf die Intranetseite erst einmal alle Mitarbeiter der SAP AG Zugriff haben.

b) Entscheidung – Wahl einer Alternative

Nachdem nun die Bewertungen der Alternativen, die Argumente und die Diskussion betrachtet wurden, fällt die Wahl der Frage „Wie“ auf Alternative (i). Damit bleiben für die Frage „Wo“ noch Alternativen (1) und (2), und die Wahl fällt aufgrund der angeführten unterstützenden Argumente (vgl. auch Abbildung 18 und Abbildung 21, S. 86) auf Alternative (2).

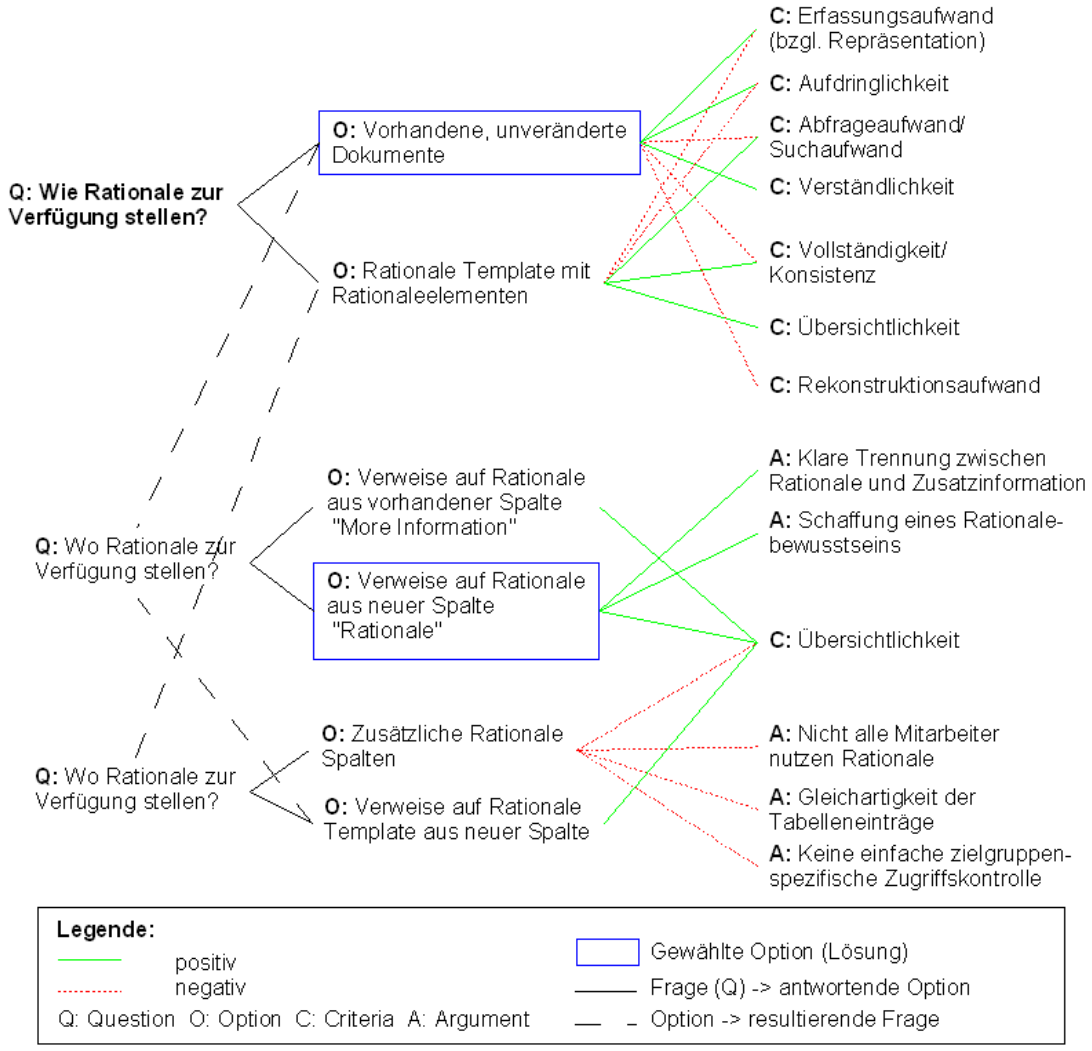

Abbildung 18: Rationale für Rationale Integration in Decision Log (graphische QOC Notation)

c) Weitere Überlegungen

Durch weitere einfache Modifikationen des Decision Log könnte das Gesamtbild einer Entscheidung noch erweitert und das Rationale noch vervollständigt werden. So kann es z.B. durchaus vorkommen, dass es zu einer getroffenen Entscheidung mehrere verwandte Entscheidungen gibt, die eben bei der Gewinnung eines einheitlichen Gesamtbildes helfen. Durch eine weitere Spalte in der Tabelle, die auf verwandte Entscheidungen verlinkt, wäre dies einfach zu erreichen. Man kann nun argumentieren, dass verwandte Entscheidungen bereits über die Topic-Spalte identifizierbar sind. Dies ist für einen groben Zusammenhang auch gegeben (z.B. *SRS Quality Improvement* in Tabelle 8), allerdings sind hierüber nicht aufeinander aufbauende beziehungsweise verwandte Entscheidungen in diesen groben Gruppen zu verknüpfen. Mit einer weiteren Spalte wäre also eine zweistufige Hierarchie und Gliederung zu erreichen.

Wie oben angemerkt, befinden sich in den Informations-E-Mails häufig noch weitere Informationen zu den Entscheidungen, und eine Historie dieser E-Mails findet sich auf derselben Intranetseite. Es besteht aber meist keine eins-zu-eins Beziehung zwischen einer Entscheidung im Decision Log und einer Informations-E-Mail, da diese normalerweise mehrere Entscheidungen gebündelt kommunizieren. Ein zusätzlicher Link von der Entscheidung auf diese E-Mails macht die darin enthaltenen Zusatzinformationen ebenfalls ohne weiteres Suchen zugänglich. Die neuen Beziehungen innerhalb des Decision Log nach den vorgeschlagenen Änderungen gegenüber der ursprünglichen Version (Abbildung 16) sind in Abbildung 19 dargestellt.

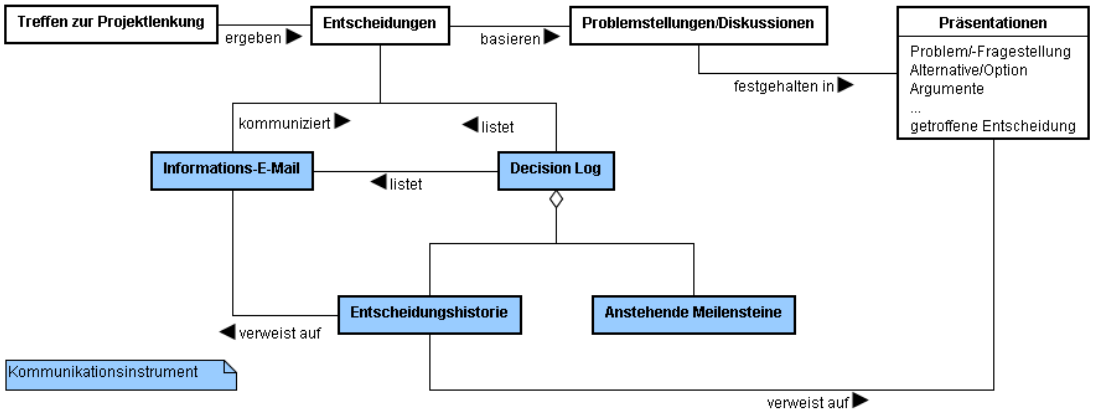

Abbildung 19: Decision Log mit expliziten Verweisen auf weitere Informationen (UML-Notation)

Die oben gemachten Vorschläge gehen davon aus, dass die zur Verfügung gestellten Dokumente ausreichende Informationen beinhalten, um das Rationale hinter einer Entscheidung alleine durch die Verfügbarkeit der Dokumente rekonstruieren zu können. Dies ist aber nicht immer gegeben. Die Spiegelung der Diskussionen in die Protokolle wäre hier eine Lösungsmöglichkeit, um dieser Herausforderung zu begegnen. Um dies zu erreichen, kann man den Ansatz des **Dialogue Mapping** (wie er in Kapitel 3.1.1 beschrieben wurde) anwenden. Dieser versucht eine Möglichkeit zu bieten, eine unstrukturierte bis semi-strukturierte Diskussion in ein einfaches semi-formales Schema zu überführen. Dieses würde dann das Protokoll ergänzen und könnte zusätzlich aus dem Decision Log heraus zugänglich gemacht werden.
Weiterhin würde sich der Vorschlag über die Integration des Dialogue Mapping auch etwas in Richtung präskriptivem Ansatz ausrichten. Denn die Strukturierung der Diskussion und das offen legen von Sichtweisen der verschiedenen Interessenvertreter unterstützt die Entscheidungsfindung (Kapitel 3.1.1.3). Dies wurde bislang in der deskriptiven Ausrichtung der Vorschläge noch nicht berücksichtigt.
Diese Art der Rationaleerfassung könnte gegebenenfalls sogar zu einem gewissen Grad auch innerhalb sowieso ausgeführter Aktivitäten durch Disziplin und eine Bewusstseinsänderung erzielt werden, so z.B. durch die „Verschiebung“ der Protokollantenrolle eines Treffens hin zu einer *Dialogue Mapper* Rolle (vgl. Kapitel 2.2.2.2).
Die gesamte Diskussion der in diesem Abschnitt diskutierten Lösung ist auch nochmals mittels eines weiteren Rationale Ansatzes (IBIS, vgl. Kapitel 2.1.4.1) und des Compendium Werkzeugs (vgl. Kapitel 3.1.1) in Abbildung 21 (siehe Anhang, S. 86) dargestellt. Durch die vorangegangenen Erklärungen der Diskussion wird Abbildung 21 verständlich.

d) Eintrag im Decision Log – Beispiel mit Rationale

Nach Aufnahme der Vorschläge in den Decision Log ist die Entscheidung aus Kapitel 4.4.1.1b) nun erneut in Tabelle 8 dargestellt.

General Decisions (Program Mgmt Meetings)					
Date	**Topic**	**Decision**	**Related Decisions**	**More Information**	**Rationale**
…					
2007-02-08	SRS Quality Improvement	The Mgmt Team agreed, that Requirements Inspections have to be conducted in all sub-programs with PRIO 1. 50% of the „Review version" SRS (status) should be covered until end of March. This has to be communicated to all the authors of the SRS to schedule the inspection meetings with all required participants (SM, Front-end Dev, BP Dev)	2006-08-12	Guidance to conduct an inspection	Information-E-Mail Presentation that led to the decision
…					
2006-08-12	SRS Quality Improvement	Supplementary to the SRS review process there will be an SRS inspection process. Some pilots will be conducted in every sub-program.			
…					

Tabelle 8: Eintrag im Decision Log – mit Rationale

Durch die Verlinkung auf die weiteren Dokumente aus dem Decision Log, kann die Entscheidung nun nachvollzogen, besser verstanden und umgesetzt werden. Die einzelnen Dokumente enthalten in diesem Beispiel folgende Informationen und sind von der Entscheidung aus erreichbar, was ursprünglich nicht der Fall war:

Informations-E-Mail:

- Verweis auf die Einführungsunterlagen zu den Inspektionstreffen (*Guidance to conduct an inspection*)
- Weitere Entscheidungen, die in den Managementtreffen gefällt wurden und in dieser Informations-E-Mail mit kommuniziert wurden

Präsentation:

- Problemstellung (Qualitätsprobleme mit den SRS)
- Vorgeschlagenen Lösungen (Inspektion)
- Erfahrungen mit Piloten (positives, negatives)
- Schlussfolgerungen

Einführungsunterlagen zu den Inspektionstreffen, welche allerdings nicht als Rationale zu betrachten sind, aber weitere Kontextinformationen zur Verfügung stellen:

- Ziele einer Inspektion
- Charakteristiken einer Inspektion
- Rollen, Aufgaben und Verantwortlichkeiten
- Vorgehen bei einer Inspektion
- formale Kriterien beziehungsweise Checklisten zur Durchführung einer Inspektion

4.4.2 Festsetzung von Produktcharakteristika und deren Kommunikation

Innerhalb des Softwarelebenszyklus PIL – und damit auch bei der Variante PIL@AP – beginnt die Entwicklung eines Produktes im Allgemeinen mit der Phase *Invent* (vgl. Kapitel 4.1). Eingang (*input*) in diese Phase finden erste Produktideen, Marktanforderungen und strategische Überlegungen etc. Auf Basis dieser Informationen wird dann vom *Product Portfolio Council* (PPC) das Produktportfolio der SAP AG angepasst beziehungsweise erweitert und an die *Define*-Phase „übergeben". In dieser wird dann der Umfang des Produkts konkretisiert und die Projektstruktur aufgesetzt.
Im Falle des neuen Mittelstandprodukts der SAP AG, welches auf der BPP aufbaut, und dem dafür angepassten Entwicklungsprozess PIL@AP wurde diese Phase abweichend gestaltet. Der Entscheidung für das neue Produkt lag ganz klar eine strategische Überlegungen der SAP AG zugrunde: der Eintritt in den Mittelstandsmarkt (*Small and Midsize Enterprises*, SME). Hierbei sollte auf die lange Erfahrung im Großkundensegment zurückgegriffen werden, in dem meist mehr, aber nicht zwangsläufig komplexere Geschäftsprozesse angesiedelt sind. Strategisch motiviert war auch die Untermauerung der technologischen Vorreiterrolle der SAP AG, indem das Produkt konsequent eine *Enterprise Service-oriented Architecture* (*enterprise SOA*) umsetzen sollte. Somit wurde die *Invent*-Phase als ein Forschungsprojekt mit der Erstellung eines Prototypen durchgeführt, welches der Definition der Rahmenarchitektur und der Produktausrichtung diente.
Aus den Strategien und Erfahrungen ergaben sich die so genannten *Product Assumptions* [PAss]. Sie sind grundlegende Festlegungen (**Prämissen**), die als Basis für die Entwicklung des neuen Produktes dienen und die Richtung beschreiben, auf die sich geeinigt wurde. Damit werden sie an den Anfang der *Define*-Phase des Mittelstandprodukts gestellt.
Die Product Assumptions enthalten auf hoher Abstraktionsebene

- generelle Prämissen zu technischen Themen, Internationalität, Installation, Wartung und Betrieb,
- Architekturkonzepte, wie Model Driven Development (MDD), Softwaredekomposition (*enterprise SOA* Entitäten), Konzepte zum UI, zur Erweiterung, zur Konfiguration, zur Sicherheit, zur Identitäts- und Zugangskontrolle etc. und
- nicht-funktionale Anforderungen zu Themen wie Total Cost of Ownership (TCO), Softwarestruktur (Wiederverwendung, Namenskonventionen), Test (-abdeckung), SAP Produktstandards[32] und Internationalität (*globalization*).

Die Product Assumptions sind also die zentrale Orientierung für die Produktentwicklung, mit der die Beteiligten des Programms/Projektes (vgl. Kapitel 4.4.1) arbeiten müssen. Um die Prämissen aber geeignet umsetzen zu können, müssen diese ausreichend verstanden werden. Außerdem ist aus Sicht der Ersteller der Product Assumptions (und letztendlich der gesamten Entwicklungsorganisation) wichtig, deren **Aktualität** und **Gültigkeit** effizient überprüfbar zu machen. Denn durch Änderung des Kontextes (Erfahrungen, Markterwartungen, Wettbewerber, Kunde, Unternehmensstrategie), in dem die Prämissen entstanden sind, können diese ihre Gültigkeit verlieren. Sie müssen dann gelöscht oder modifiziert werden beziehungsweise neue Prämissen müssen hinzukommen. Weiterhin können die Prämissen auch einfach falsch sein und sollten einer Validierung unterzogen werden können, denn die „Fortpflanzung" einer falschen Annahme wird sich später in dem Produkt widerspiegeln.
Als Qualitätsziele für eine aufgestellte Prämisse können also angenommen werden:

[32] Die SAP Produktstandards sind Eigenschaften, die von allen Produkten der SAP eingehalten werden müssen, unabhängig von ihrer betriebswirtschaftlichen Herkunft. Sie dienen der Herstellung einer einheitlichen Produktqualität und umfassen übergreifende funktionale (z.B. Data Archiving, Business Solution Configuration) und nicht-funktionale (z.B. Security, Performance, Usability, Accessibility) Anforderungen, die vom Markt implizit erwartet werden. Die Produktstandards können somit mit einer Art „SAP internen" ISO 9126 [ISO9126] verglichen werden.

- Verständlich-/Nachvollziehbarkeit
- Aktualität
- Zeitliche Gültigkeit
- Validität

Aufgrund dieser Tatsachen erscheint ein ausgeprägtes „*Assumption Management*“ sinnvoll, wie es auch Lehman und Fernández-Ramil [LeFe06] grundsätzlich für die Softwareentwicklung vorschlagen. Wie die gerade identifizierten Qualitätsziele für das Assumption Management und damit für die in den Product Assumptions aufgestellten Prämissen durch den Einsatz von Rationale umgesetzt werden können, wird im Folgenden kurz diskutiert.
Die Anregung für die Untersuchung der Product Assumptions kam von einem langjährigen Mitarbeiter aus dem Umfeld der für das Produkt zuständigen Operations-Abteilung. Dieser sah Handlungsbedarf in Bezug auf obige Ziele, da die Product Assumptions eine zentrale Orientierung für das Entwicklungsprogramm/-projekt darstellen und von vielen Projektbeteiligten genutzt werden müssen. Mit dem Mitarbeiter der Operations-Abteilung wurden mehrere Gespräche darüber geführt, wie die Integration von Rationaleelementen in die Product Assumptions aussehen könnte.

4.4.2.1 IST-Situation der Product Assumptions

a) Product Assumptions – Lösung ohne Rationale

Die Product Assumptions existieren als ca. 470 einzelne Prämissen (*assumptions*) zu oben genannten Themen in einer Microsoft PowerPoint Präsentation [PAss] und in einem detaillierten Microsoft Excel Sheet [WSHE], das jede Prämisse identifizierbar enthält. Das Microsoft Excel Sheet enthält auch eine (logische) Referenz auf die Folie der PowerPoint Präsentation, auf der sich die Prämisse befindet. Der potentiell beschränkten Gültigkeitsdauer beziehungsweise dem Hinzukommen von Prämissen wird Rechnung getragen, indem diese angepasst und in verschiedenen Versionen veröffentlicht werden. Die Änderungen von einer Version zur nächsten werden in dem Microsoft Excel Sheet ebenfalls für jede einzeln identifizierbare Prämisse festgehalten. Es wurden allerdings weder die Begründungen für die initiale Version der Prämissen festgehalten noch werden die Änderungen von Version zu Version begründet. Die Struktur des Excel Sheets ist in Tabelle 9 dargestellt.

	Version 1.0		**…**	**Version 0.1**	
ID	**Slide**	**Assumption**		**Slide**	**Assumption**
…					
…					

Tabelle 9: Struktur der Excel Tabelle zur Verwaltung der Product Assumptions

b) Abgeleitete Probleme und anvisierte Ziele

Es sollen nun zwei Beispiele für Prämissen gegeben werden, um die vorhandenen Defizite bezüglich den Qualitätskriterien für das Assumption Management aufzuzeigen und davon ausgehend im folgenden Abschnitt eine Verbesserung vorzuschlagen:

1. Das Produkt soll unabhängig sein von der verwendeten Datenbank.
2. Für jede *enterprise SOA* Entität vom Typ X muss ein Modell entwickelt werden.[33]

Das erste Beispiel für eine Prämisse wird verwendet, um einmal die **Probleme** bei der Anwendung der Product Assumptions und zum anderen die Probleme bei deren Verwaltung darzustellen. Für beide Prämissen wird dann bei der Beschreibung der SOLL-Situation die jeweilige Verbesserung durch die Erfassung von Rationale erarbeitet.

[33] Diese Prämisse steht im Zusammenhang mit der Anwendung des Model Driven Development (MDD), in dessen Umfeld Softwaresysteme (durchgängig) mit Hilfe von Modellen beschrieben werden.

Als Entwickler, der nun mit diesen Prämissen arbeiten muss (Anwendung), kann man diese nun einfach hinnehmen, ohne sie weiter zu hinterfragen. Man könnte aber auch Mutmaßungen anstellen, dass SAP die Unabhängigkeit ihrer Kunden so viel bedeutet, dass diese bezüglich einer Datenbank nicht eingeschränkt werden sollen (bezogen auf Prämisse 1). Ob dies aber der Realität entspricht, bliebe offen.
Auch die Überprüfung der Prämissen auf Validität, Gültigkeit und Aktualität ist eingeschränkt (Verwaltung). Denn man kann z.B. nicht ohne den Grund für die Prämisse zu kennen, beurteilen, ob diese noch gültig und aktuell ist, beziehungsweise überhaupt valide war. Es kann ja sein, dass sich die Strategie der SAP AG gewandelt hat und man nun den Kunden an eine Datenbank aus dem eigenen Portfolio binden und deshalb die Verwendung anderer Datenbanken einschränken möchte.

Daher sind die vordergründigen **Ziele** für diesen Vorschlag zur Lösung der Probleme die Folgenden:

- Bessere Kommunikation und Erklärung der Entscheidungen in den Product Assumptions (Transparenz und Nachvollziehbarkeit)
- Unterstützung bei der Erkennung von Kontextänderungen und resultierenden Änderungen der Prämissen (Entscheidungsrevision)

Zusätzlich sollte auch ein weiteres Ziel beachtet werden:

- Möglichst geringe zusätzliche Ressourcen und wenig Änderungen an Bestehendem

4.4.2.2 SOLL-Situation der Product Assumptions

a) Product Assumptions – Vorschlag für Aufnahme von Rationale (Diskussion)

Für die Aufnahme von Rationale im Kontext der Product Assumptions können dieselben Fragen gestellt werden wie in Kapitel 4.4.1.2 (**wie** und **wo** Rationale erfasst werden soll). Die erste Frage „Wie soll Rationale zur Verfügung gestellt werden?“ läuft auch hier darauf hinaus, ob das Rationale stärker formalisiert oder stärker informal repräsentiert wird. Hier ergeben sich identische Argumentationslinien wie in Abschnitt 4.4.1.2a). Somit wird diese Diskussion hier nicht erneut geführt.
Für die Fragestellung „Wo soll das Rationale zur Verfügung gestellt werden?“ werden zwei Alternativen betrachtet:

i. Aufnahme des Rationale in dem Excel Sheet, das auch die Änderungen der Prämissen dokumentiert
ii. Aufnahme des Rationale in einem eigenen Dokument zusätzlich zu dem genannten Excel Sheet

Diese Alternativen werden im Folgenden nach den anwendbaren, sich nicht neutral verhaltenden Kriterien aus Kapitel 4.3 bewertet:

Abfrageaufwand. Die Abfragemöglichkeit wird in beiden Alternativen über das Programm zur Verfügung gestellt werden, in welchem das Dokument erstellt wird. Deshalb ist dieses Kriterium nicht extra aufgeführt. Der *Abfrageaufwand* kann allerdings für beide Alternativen unterschiedlich sein, wenn man Informationen zu den Prämissen zusammen mit dem Rationale erhalten möchte. Denn wenn beides in einem Dokument ist (Alternative (i)), dann sind diese Informationen integriert, nebeneinander und auf einen Blick vorhanden.

Verständlichkeit. Die Verständlichkeit ist für beide Alternativen nahezu gleich gut. Der Vorteil von Alternative (i) ist darin zu sehen, dass man nur ein Dokument benötigt, um diese Verständlichkeit zu gewährleisten, bei Alternative (ii) benötigt man mehrere Dokumente, um die Informationen entsprechend zu verknüpfen und sie verständlich zu machen.

Übersichtlichkeit. Hier ist Alternative (i) etwas im Nachteil, da das Dokument sehr groß wird (ca. 470 Prämissen in verschiedenen Versionen mit Rationale).

Erfassungsaufwand. Der Erfassungsaufwand für Alternative (i) ist etwas niedriger, da nur mit einem Dokument umgegangen und nur dieses verwaltet werden muss.

Implementierungsaufwand. Dieser ist ebenfalls als sehr gering einzustufen. Es muss lediglich ein Dokument bei Alternative (i) geändert beziehungsweise ein neues bei Alternative (ii) eingeführt werden.

b) Entscheidung – Wahl einer Alternative

Da die Prämissen strategische Produktentscheidungen darstellen beziehungsweise sich daraus ableiten und es vordergründig um die Ziele Verständlichkeit und Überprüfbarkeit geht, ist eine weniger formale Repräsentation des Rationale ausreichend, die das Ziel und/oder eine Begründung für die Prämisse enthält. Dies kann geschehen, ohne unbedingt explizit die Alternativen und Kriterien etc. für die Entscheidung über eine Prämisse berücksichtigen zu müssen.
Basierend auf den Abwägungen zur Frage „Wo?“ wird sich weiterhin für die Integration des Rationale in das vorhandene Excel Sheet entschieden.
Um die gewünschten Informationen in das Dokument aufzunehmen, wird dieses durch Spalten erweitert. Dabei sollten auch die Gründe für fortschreitende Änderungen (Versionen) der Prämissen mit erfasst werden. Die vorgeschlagene veränderte Struktur des Microsoft Excel Sheet ist in Tabelle 10 dargestellt.

	Version 1.0			**…**	**Version 0.1**			
ID	**Slide**	**Assumption**	**Reason for Change/ Assumption**		**Slide**	**Assumption**	**Reason for Assumption**	**Objective of Assumption**
…								
…								

Tabelle 10: Product Assumption Management mit Rationale

c) Product Assumptions – Beispiel mit Rationale

Um die erzielten Verbesserungen aufzuzeigen, wird nun noch einmal auf die beiden beispielhaften Prämissen zurückgegriffen.

Der Grund für Prämisse 1 (Datenbank) könnte wie folgt angegeben werden:

Datenbanken sind nicht das Kerngeschäft von SAP. Deswegen soll der Kunde nicht auf eine bestimmte Datenbank festgelegt werden.

Damit kann die Prämisse nun einmal vom Entwickler nachvollzogen werden und bei Änderung der Strategie von SAP, welche nun auch eine eigene Datenbank entwickeln könnte, kann die nicht mehr aktuelle/gültige Prämisse schnell identifiziert und entsprechend angepasst werden.
Dies gilt auch für eine Begründung von Prämisse 2:

*Das gesamte Produkt soll sich modular aus Instanzen der in einer Taxonomie (*vgl. Kapitel 4.1*) definierten Entitätstypen aufbauen und die Codeskelette der modellierten Instanzen (Entitäten) sollen daraus generiert werden (entsprechend der modellgetriebenen Softwareentwicklung, MDD).*

Für diese Prämisse könnte man auch das Ziel formulieren:

SAP verfolgt eine konsequente Umsetzung des ingenieurmäßigen und modellgetriebenen Ansatzes.

Auch die Gründe hierfür – wie sie in Kapitel 4.1 aufgeführt sind – könnten noch genannt werden, um den Kontext stärker auszuführen und verständlich zu machen.

Eine einmalige Aufnahme der Gründe/Ziele der einzelnen Product Assumptions ist aufgrund des damit verbundenen Aufwands vermutlich nicht praktikabel. Allerdings ermöglicht der vorgeschlagene Ansatz eine schrittweise Aufnahme beziehungsweise Rekonstruktion der Gründe/Ziele. Dies könnte immer dann geschehen, wenn eine Änderung an einer Prämisse notwendig ist.

4.4.3 Dokumentation von Priorisierungsentscheidungen

Große Softwaresysteme werden heute nicht mehr in einem Schritt geplant und dann auch nicht in einem Schritt implementiert, wie es etwa durch die strikte Anwendung des Wasserfallmodells gegeben wäre [Royc70]. Vielmehr wird heute in mehreren Iterationen vorgegangen, um zeitlich flexibler agieren zu können und die Risiken von Fehlplanungen und Fehlentwicklungen zu verringern, die man sich sonst mit langfristiger Planung einkauft. Diese Ideen finden sich schon im Spiralmodell [Boeh88] wieder und werden aufgrund der Erfolge durch Anwendung von agilen Vorgehensmodellen [Fowl05] auch in die „schwergewichtigeren Prozesse“ aufgenommen. In diesen werden dann bestimmte Aktivitäten „agil“ adaptiert, um das Vorgehen effizienter zu gestalten.
Wie bereits in Kapitel 4.1 erwähnt, enthält auch der PIL und der PIL@AP verstärkt iterative Elemente. So wird im PIL@AP die Entwicklung in Iterationen vorangetrieben, in denen geschlossen verwendbare Anteile des Produktes entstehen. Diese stellen nicht notwendigerweise ein eigenes Release dar, tragen aber letztlich zu einem solchen bei. Der Umfang von Releases muss vor deren Planung und Implementierung in den Phasen „*Engineer*“ und „*Implement*“ geplant werden. Diese Planungsaktivitäten beinhalten Priorisierungsentscheidungen, die festlegen, welche Anforderungen auf einer „Wunschliste“ in die kommenden Releases aufgenommen und welche nach hinten verschoben werden. Innerhalb der Releases wird dann nochmals der „Iterationsarbeitsvorrat“ für die nächste Iteration und der „Rest“ (*backlog*) unterschieden. Die Wunschliste kann also aufgrund ihres Umfangs nicht komplett abgearbeitet werden, wodurch in einem Abstimmungsprozess zwischen Solution Management und Entwicklung die Priorisierungen vorgenommen werden müssen. Die Priorisierungsentscheidungen haben zum Ziel, den Gesamtnutzen (unternehmensintern und -extern) zu maximieren.
Diese Entscheidungen sind schwierig, weil sie mit einem hohen Maße an Unsicherheit (z.B. bezüglich Marktwert oder Entwicklungsaufwand der Anforderung) verbunden sind. Gleichzeitig haben sie starke Auswirkungen auf die Kundenzufriedenheit und wirken sich damit auch indirekt auf den Produkterfolg aus. Deshalb sind wohl überlegte und begründete Priorisierungsentscheidungen gewünscht. Eine Sensibilität für Probleme beziehungsweise vergangene Fehlentscheidungen kann weiterhin die Basis bilden für eine kontinuierliche Verbesserung der Entscheidungen.
Einen Ansatz, die Prozesse zur Anforderungsauswahl für die Releases zu verbessern, zeigen Karlsson et al. [KRKO03] auf. Die Autoren schlagen eine retrospektive Analyse aller Requirements vor (in Form eines Workshops), die potentiell in die Releases hätten einfließen können. Dies soll geschehen, nachdem sich die jeweiligen Softwarereleases bereits 18 Monate auf dem Markt befanden. Hierbei soll den Beteiligten Rückmeldung gegeben werden, ob sich die Entscheidungen ausgezahlt haben, wo suboptimale Entscheidungen getroffen wurden und noch Verbesserungspotential besteht.
In diesem Kapitel soll nun untersucht werden, wie Rationale im Rahmen von Releaseentscheidungen im Umfeld des PIL@AP angewendet werden kann. Bevor in die Diskussion eingestiegen wird, wird noch der Prozess vorgestellt, in dessen Kontext die Entscheidungen gefällt werden.
Die Idee für die folgenden Überlegungen in diesem Bereich kam aus eigenen Beobachtungen während der Anwesenheit bei mehreren Software Requirements Inspektionen. Hierin hat sich gezeigt, dass es während des Abstimmungsprozesses verschiedene Problembereiche gibt, die

gegebenenfalls durch Integration von Rationaleelementen verbessert werden könnten (vgl. Kapitel 4.4.3.2b)).

4.4.3.1 Der Prozess zur Spezifikation von Anforderungen an Produkteigenschaften innerhalb des PIL@AP

Der Spezifikationsprozess für Anforderungen an Produkteigenschaften (*Software Requirements Specification Process*) innerhalb des PIL@AP (Kapitel 4.1) besteht aus sechs Schritten, die weiter unten noch ausführlicher beschrieben werden. Der Input für diesen Prozess und die entstehenden Anforderungsdokumente (*Software Requirements Specification*, SRS) kommt aus den Marktanforderungen, die in der *Define*-Phase identifiziert und meist sehr grobgranular als Geschäftsszenarien (*Business Scenario*) zur Verfügung gestellt werden. Diese Marktanforderungen sind in der „Sprache" des Marktes ohne Systembezug verfasst und stellen einen Wunsch nach Unterstützung des Geschäftsszenarios dar. Durch die Erstellung der SRS wird dieser Wunsch nun als Anforderungen an das Produkt formuliert (Produktanforderungen), d.h. es wird spezifiziert, was das Produkt zur Unterstützung des Geschäftsszenarios leisten soll.
Die SRS für die Geschäftsszenarien bilden allerdings nur den Ausgangspunkt für weitere SRS verfeinerter Geschäftsentitäten. Die Spezifikation der Anforderungen an die Produkteigenschaften findet also in einer Hierarchie statt. Diese dient der Strukturierung der Anforderungen und verfeinert sie schrittweise entlang der (*enterprise SOA*) Entitätstypen auf verschiedenen Detailebenen, ausgehend von einer festgelegten Taxonomie (vgl. Kapitel 4.1).
Die Schritte zur Erstellung einer SRS sollen kurz angerissen werden. Dabei unterscheiden sich diese für die verschiedenen Entitätstypen nicht, allerdings unterscheiden sich die ausführenden Rollen (*Specification Author* aus SM oder Dev) und der verwendete Input und Output (z.B. Vorlagen, Richtlinien etc.). Abbildung 20 zeigt den Prozess im Überblick mit dessen Aktivitäten, Rollen, Dokumenten und Werkzeugen. Welche Entität mit welchem Entitätstyp spezifiziert wird, ergibt sich als Output eines vorgelagerten Prozesses.

Anlegen der SRS (Create Software Requirements Specification Instance). In diesem Schritt wird eine SRS angelegt, d.h. sie wird dem späteren Autor als Vorlage im Projektmanagement- (cProject) und Spezifikationswerkzeug (SAPSpec) zur Verfügung gestellt.

Schreiben der SRS (Describe Scope and Business Context). In diesem Schritt wird der Umfang und der Geschäftskontext der SRS in verschiedenen Kapiteln und dezidierten Anforderungen beschrieben. Je nach Granularität wird die SRS von einem Autor (*Specification Author*) aus dem Solution Management spezifiziert, welcher die Anforderungen aus Kunden- und betriebswirtschaftlicher Sicht in Geschäftsszenarien und Szenarienvarianten (*Scenario Variants*) beschreibt. Die Spezifikation der technischen Elemente (*enterprise SOA Entities*) auf den feingranularen Ebenen dagegen übernehmen Autoren der Entwicklungsabteilungen (Dev).

Pflegen der SRS (Maintain Software Requirements). Dient der Aufrechterhaltung von Aktualität, Konsistenz etc. der SRS.

Durchsehen der SRS (Review SRS). Um Inkonsistenzen, Probleme und Fehler in den SRS zu identifizieren und eine Abstimmung zwischen den verschiedenen Parteien zu erreichen, die mit einer SRS arbeiten (SM, Dev, InfoDev), werden in diesem Schritt Reviews beziehungsweise Inspektionen mit Experten aus den verschiedenen Bereichen praktiziert.

SRS abschließen (Clarify Review Topics). Nach den Reviews und Inspektionen müssen – falls vorhanden – offene Punkte diskutiert, ausgearbeitet und in die SRS übernommen werden. Gegebenenfalls wird danach nochmals ein Review oder eine Inspektion angesetzt.

SRS übergeben: In diesem Schritt wird die fertig gestellte SRS zum weiteren Softwaredesign freigegeben. Gegebenenfalls findet das auch gemeinsam mit einer formalen Übergabe in Form einer Inspektion statt, die das Dokument auf formale Kriterien (z.B. *Design independence, Unambiguity, Testability, Comprehensibility*) hin überprüft.

Die Festlegung der Priorisierungen findet vor allem im Schritt „*Schreiben der SRS*“ statt. Gegebenenfalls notwendige Änderungen werden später in Abstimmung mit den anderen von der SRS betroffenen Parteien in den Reviews und Inspektionen vorgenommen. Die genauen Bedingungen werden weiter unten beschrieben.

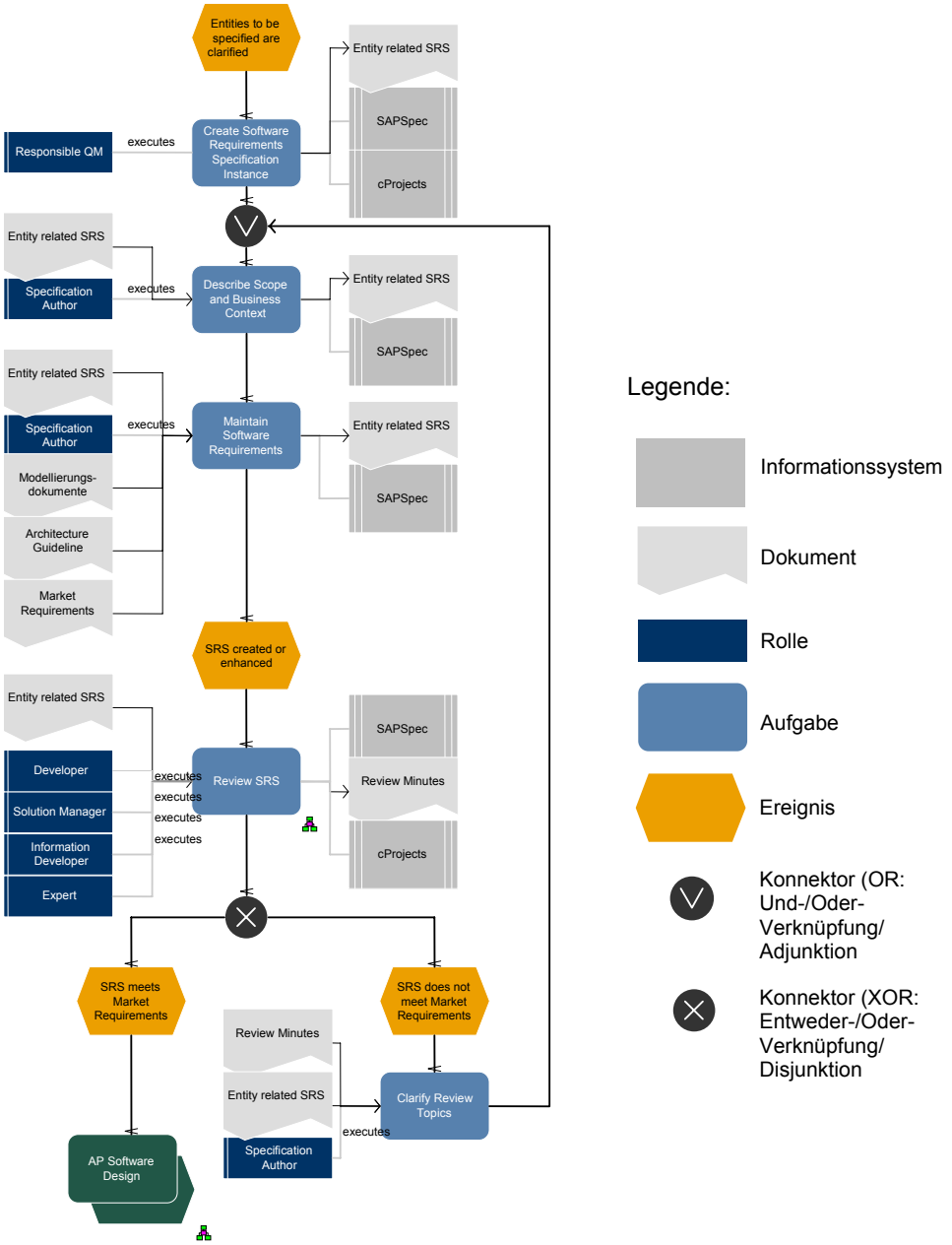

Abbildung 20: Software Requirements Specification (SRS) Prozess im PIL@AP (ARIS-Notation) (Quelle: in Anlehnung an [Gebh07b, S. 9])

4.4.3.2 IST-Situation der Priorisierungsentscheidungen

a) Priorisierungsentscheidungen – Lösung ohne Rationale

Der Requirements Engineering Ansatz bei der SAP AG verfolgt das Ziel, jede Anforderung an die Software als eine identifizierbare Einheit zu erfassen, was Grundlage für die Etablierung von Verfolgbarkeit (*traceability*) über mehrer Verfeinerungsebenen ist. Dies wird ebenfalls dafür genutzt, jede Anforderung mit entsprechenden Testfällen in Beziehung zu setzen.
Die verschiedenen SRS werden in einem Spezifikationswerkzeug (SAPSpec) geschrieben, das die Anforderungen in einer Datenbank persistiert. Eine SRS enthält verschiedene Kapitel, die unterschiedliche Themenbereiche (z.B. Organisation, Geschäftskontext und Umfang, Verweise auf weitere Dokumente, Glossar, Software Requirements) abdecken. Zentral sind die Software Requirements, die jeweils folgende Informationen enthalten [Gebh07b, S. 12f.]:

REQ-ID: Automatisch generierte Nummer zur Identifikation der einzelnen Anforderung.
Trace To: Referenz auf die Anforderung in einer SRS auf einer höheren Ebene, von der diese Anforderung abgeleitet wurde; dient der Verfolgbarkeit.
Software Req: Kurze textuelle Beschreibung der Anforderung.
Remark: Weitere Informationen/Anmerkungen, die zum Verständnis der Anforderung oder des Kontexts beitragen.
S (Scope): Legt fest, in welchem Release die Anforderung umgesetzt werden soll.
BEQ (Bequeath): Referenz auf abgeleitete Anforderungen in einer SRS auf einer tieferen Ebene. Dient der Verfolgbarkeit.

Ein Beispiel für ein Software Requirement ist in Tabelle 11 gegeben.

REQ-ID	Trace to	Software Requirements	Remarks	S	BEQ
…					
010-12	003-24	It shall be possible to get an overview about all received tasks within a certain period (this day, last day, last week, all tasks).		FP1	023
...					

Tabelle 11: Beispiel für ein Software Requirement in einer SRS

In der SRS werden die Priorisierungsentscheidungen (repräsentiert über das *Scope* Attribut) aufgrund verschiedenster implizit vorhandener Kriterien getroffen. Aus Sicht des Solution Management sind das in erster Linie Kriterien wie Kundennutzen und -zufriedenheit sowie die Bedeutung einer Anforderung für den Geschäftskontext.
Während des Review Prozesses ändern sich die angesetzten Kriterien, wenn die Entwicklungsabteilung mit einbezogen wird. Hier rücken Kriterien wie (technische) Machbarkeit, Kosten und vorhandene Ressourcen in den Vordergrund. Die verschiedenen Perspektiven auf die SRS können infolgedessen die ursprünglichen Planungen des Solution Management beeinflussen und damit letztendlich die Implementierungsreihenfolge der Anforderungen ändern. Dies sollte in einem Abstimmungsprozess stattfinden, in dem Konsens Pflicht ist, geschäftsorientierte Anforderungen tendenziell allerdings stärker zu gewichten sind als technische Bedenken, die diesen entgegenwirken [WoAu05].

b) Abgeleitete Probleme und anvisierte Ziele

Wie in eigenen Beobachtungen dieser Abstimmungen festgestellt wurde, bestehen hier verschiedene Probleme:[34]

[34] Diese Probleme bestehen für die SRS, die vom Solution Management erstellt werden und dann mit der Entwicklung abgestimmt werden müssen. In den weiteren Ausführungen dieses Kapitels werden diese SRS als Grundlage für die Ausführungen verwendet.

1. Die verschiedenen Entwicklungsabteilungen (z.B. Front-End oder Back-End) begründen die Verschiebung von Anforderungen in spätere Releases häufig mit fehlenden technischen Voraussetzungen, die die Machbarkeit der Anforderungen einschränken.[35]
2. Diese Diskussionen über Machbarkeit werden häufig wiederholt und sind für das Solution Management nicht immer verständlich und nachvollziehbar.
3. Die Auftretenshäufigkeit der einzelnen technischen Begründungen für das Verschieben von Anforderungen ist nicht nachvollziehbar. Dadurch kann die Dringlichkeit für deren Behebung nicht gut begründet werden.

Als dringlichste Maßnahme zur Verbesserung der aktuellen Probleme und deshalb als oberstes Ziel für den unterbreiteten Vorschlag wird die Stärkung des Solution Management zur Durchsetzung von geschäfts- und kundenorientierten Anforderungen gesehen.

4.4.3.3 SOLL-Situation der Priorisierungsentscheidungen

a) Priorisierungsentscheidungen – Vorschlag für Aufnahme von Rationale (Diskussion)

Stärkung des Solution Management

Es wird davon ausgegangen, dass Konsens über die Notwendigkeit und Dringlichkeit geschäftsorientierter Anforderungen zwischen den beteiligten Parteien einfacher erzielt werden kann, wenn die Position des Solution Management durch fundierte Informationen über die Gründe für verschobene Anforderungen gestärkt wird. Zwar hängt das Gewicht des Solution Management auch von dessen Akzeptanz in der Organisation ab („akzeptierte Expertise"), kann das Solution Management allerdings belegen, dass geschäftsorientierte Anforderungen mehrmals aufgrund suboptimaler Gründe verschoben wurden, sollte der Forderung, hier Abhilfe zu schaffen, eher entsprochen werden. Eine Dokumentation der Verschiebung ursprünglich vom Solution Management festgelegter Priorisierungen (*Scope* Attribut) kann anhand der Kriterien für die Verschiebung stattfinden. Diese können dann später zur Filterung der Verschiebungen herangezogen werden, um einen Überblick über Verschiebungsgründe zu erhalten. Beispiele für Kriterien sind im Folgenden angeführt, müssen aber gegebenenfalls weiter angepasst werden:

- fehlende Ressourcen bei den Umsetzenden (Zeit, Geld etc.)
- technische Restriktionen
    - im Verantwortlichkeitsbereich der abstimmenden Parteien
    - außerhalb des Verantwortlichkeitsbereichs der abstimmenden Parteien (vgl. Fußnote 35)
- verbundenes Risiko
- Vorzug anderer Anforderungen aufgrund deren
    - (Kunden-) Nutzen
    - Quelle (Bedeutung, Macht, Einfluss)
    - Dringlichkeit
    - Reihenfolgeanforderung/Abhängigkeiten

Der Vorschlag soll an einem konkreten Beispiel erläutert werden:
Das Solution Management hat die Anforderung für ein nächstes Release, dass ein *copy/paste*-Mechanismus für mehrere Tabellenzellen in einer Benutzungsoberfläche vorhanden sein soll. Dies wird aber in einem Review der SRS von Seiten der Entwicklung als nicht machbar zurückgewiesen. Begründet wird dies damit, dass das zugrunde liegende graphische

[35] Das Problem in diesem Zusammenhang besteht auch darin, dass technische Voraussetzungen nicht unbedingt im Verantwortlichkeitsbereich der angesprochenen Entwicklungsteams des Produktes liegen. Denn das Produkt basiert auf einer Infrastruktur (Technologieplattform), die von einem anderen Teilbereich der Organisation entwickelt wird, auf den nur Einfluss über definierte Prozesse genommen werden kann. Dies benötigt häufig etwas Zeit.

Rahmenwerk (*framework*) diese Funktionalität nicht unterstütze und die Entwicklung (Front-End Dev) auf dieses Rahmenwerk keinen direkten Einfluss habe. Daraufhin muss die Anforderung auf unbestimmte Zeit verschoben werden.
Könnte das Solution Management hier belegen, dass auch in anderen SRS diese Anforderung mehrmals vonnöten wäre, so hätte eine potentielle Eskalation größere Erfolgsaussichten.

Nachdem nun die Gründe für eine Dokumentation der Priorisierungsänderungen dargelegt wurden, wird überlegt, **wo** eine solche Dokumentation stattfinden könnte. Möglichkeiten (Lösungsalternativen) für die Dokumentation der Verschiebungsgründe wären:

1. Innerhalb der SRS in der Spalte *Remarks* (vgl. Tabelle 11).
2. Innerhalb der SRS in neuen Spalten.
3. In einem zusätzlichen Dokument des Autors (mit Referenz auf das betroffene Software Requirement).
4. In einem zentralen Dokument des Solution Management (auf Programm- oder Projektebene).

Die Informationen, die in allen Optionen vorhanden sein und somit erfasst werden sollten, sind

- die REQ-ID,
- der ursprüngliche und neue Wert des *Scope* Attributs,
- der Grund der Verschiebung in Form einer Kategorisierung (entlang einer Taxonomie oder Ontologie) z.B. wie oben vorgeschlagen und
- eine Anmerkungsmöglichkeit, in der noch ausführlich der Grund für die Verschiebung angegeben werden kann.

Auf Basis dieser Informationen ist dann eine Aussage über den Verschiebungszeitraum und damit den Grad der Abweichung von der ursprünglichen vom Solution Management für notwendig gehaltenen betriebswirtschaftlichen Umsetzung möglich. Weiter kann u.a. nach den Gründen der Verschiebung gefiltert und die Häufigkeit von Verschiebungsgründen festgestellt werden.

Die genannten Alternativen werden im Folgenden gegen die Kriterien aus Kapitel 4.3 abgewogen:

Selektionsunterstützung. Der Ansatz gibt vor, dass für jede Verschiebung eine Dokumentation der Gründe stattfinden sollte. Dies ist bei allen Alternativen der Fall.

Erfassungsaufwand. Dieser ist ebenfalls für alle Alternativen gleich und als niedrig einzustufen.

Aufdringlichkeit. Für alle als niedrig einzustufen.

Abfragemöglichkeit. Für die 1. Alternative sind keine Filter nach Kategorie möglich, da diese nicht explizit angegeben wird; eine Suche nach Stichworten über den Browser wäre allerdings möglich. Für die 2. Alternative ist sowohl eine Filterfunktion und eine Suchfunktion über den Browser möglich. Da die 3. und 4. Alternative vermutlich in einem Office Programm (z.B. Microsoft Excel) realisiert würden, stehen hier die standardmäßigen Sortier-, Filter- und Suchfunktionen zur Verfügung.

Verständlichkeit. Hier besteht für die verschiedenen Alternative kein Unterschied. Die Nachvollziehbarkeit ist in allen gegeben.

Vollständigkeit/Konsistenz. Ist ebenfalls für alle Alternativen gleich.

Übersichtlichkeit. Hier besteht für die 1. Alternative ein Nachteil, da alle Informationen in einer Tabellenspalte vorhanden sind und auch noch weitere Anmerkungen (*Remarks*) vorhanden sein können.

Rekonstruktionsaufwand. Da die Darstellung des Rationale sich inhaltlich nicht unterscheidet, ist der Rekonstruktionsaufwand aus dieser Sicht identisch. Etwas schlechter wird er für die 1. Alternative sein, da die Informationen hier auch mit anderen Informationen der Spalte *Remarks* vermischt werden können. Aus Sicht eines integrierten Überblicks über alle Verschiebungen sind die 3. und 4. Alternative als geeigneter zu beurteilen. Für die 2. Alternative ist erst mal nur eine Übersicht innerhalb einer SRS möglich. Eine integrierte Übersicht (z.B. Filterung nach gewissen Kriterien) müsste erst implementiert werden.

Implementierungsaufwand. Die 3. und 4. Alternative sind einfach zu implementieren, da vorhandene Mittel (z.B. Microsoft Excel) verwendet werden können. Alternative 1 wäre ebenfalls einfach ohne Änderung umzusetzen. Die 2. Alternative stellt den höchsten Implementierungsaufwand dar, da in dem Anforderungserfassungswerkzeug Änderungen vorgenommen werden müssten (z.B. Änderungen an den Datenstrukturen, Filterfunktionen etc.).

b) Weitere Überlegungen

Überlegungen zur Wahl zwischen der 3. und 4. Alternative betreffen die Verwaltung und den Zugriff der Dokumente. Denn pflegt jeder Autor nur sein eigenes Dokument, hat man wiederum keinen integrierten Überblick über die Verschiebungsgründe auf einer höheren Ebene. Dies könnte z.B. in dem vorher genannten Beispiel dazu führen, dass man doch keine Aussage machen kann, in welchen SRS dieser Verschiebungsgrund ebenfalls aufgetreten ist. Deshalb ist es sinnvoll, das Rationale zumindest auf Projektebene zu erfassen und gegebenenfalls auf Programmebene in einem zentralen Dokument bei Bedarf zu einer umfassenden Auswertung zusammenzufassen. Ist die Struktur der Dokumente gleich, was der Fall sein sollte, ist dies ohne Probleme möglich.

c) Entscheidung – Wahl einer Alternative

Aufgrund dieser Überlegungen erscheint es sinnvoll sich für die 4. Alternative auf Projektebene zu entscheiden, da hier der Implementierungsaufwand am niedrigsten ist und die Flexibilität sowie die Abfragemöglichkeiten am höchsten.

4.5 Zusammenfassung

Das Kapitel 4.4 hat Ansätze aufgezeigt, Rationale Management im Umfeld des PIL@AP (Kapitel 4.1) anzuwenden: einen Rationale Ansatz, der dem Wissensbereich „Produkt-/ Prozesswissen“ (Projektwissen) zuzuordnen ist und sich mit der Rationaleintegration bei der Kommunikation von Projektentscheidungen beschäftigt, einen aus dem Bereich „Produkt-/ Systemwissen“, der im Umfeld der Festlegung von Produktcharakteristika Anwendung findet und einen aus dem Bereich „Produktwissen“ im Kontext von Priorisierungsentscheidungen. Dabei wurde von der Definition der Ziele in Kapitel 4.2 ausgegangen, die für jeden Vorschlag, basierend auf den konkreten Problembereichen, nochmals eingeschränkt und angepasst wurden und deren Umsetzung in den Anwendungsbereichen durch Rationale untersucht wurde. Die Diskussion der Lösungsalternativen wurde anhand der Ziele und der in Kapitel 4.3 aufgestellten Kriterien geleitet. Für die Entscheidungen wurde das entsprechende Rationale dokumentiert.
Resultat sind die genannten drei Vorschläge, die alle vorwiegend deskriptiven Charakter haben. Sie unterstützen die Entscheider also nicht unmittelbar bei einer reflektierten Entscheidungsfindung (**erstes Ziel** in Kapitel 4.2). Man kann aber anführen, dass der Ansatz für die Projektlenkungsentscheidungen gegebenenfalls indirekt eine reflektierte Entscheidungsfindung unterstützt, weil die Wege zu einer Entscheidung transparenter für die (berechtigten) Interessenvertreter zur Verfügung gestellt werden. Hierdurch kann eventuell erreicht werden,

dass die Entscheider motiviert werden, ihre Entscheidungen stärker zu reflektieren und besser zu dokumentieren (in den vorhandenen Dokumenten), da die Entscheidungen nicht mehr in dem Maße eine Black-Box für Außenstehende darstellen. Der erweiterte Vorschlag, bei den Projektlenkungstreffen auch Dialogue Mapping mit seiner präskriptiven Intention einzusetzen und diese Dokumentation dann ebenfalls zur Verfügung zu stellen, würde die deskriptive Ausrichtung des Vorschlags weiter vorangetrieben. Allerdings darf man nicht davon ausgehen, dass ohne weiteres bei allen Treffen diese Methode eingesetzt wird bzw. entstehendes Material daraus (für jeden) zugänglich gemacht wird. Die Gründe hierfür (z.B. Vertraulichkeit, Politik etc.) wurden bereits diskutiert.
Der Vorschlag für Priorisierungsentscheidungen trägt insofern zur Entscheidungsfindung bei, als er Daten (Gründe für Prioritätsänderungen) sammelt, um später fundiert Entscheidungen veranlassen zu können.
Und auch der Vorschlag im Kontext der Product Assumptions unterstützt die Entscheidungsfindung zur aktuellen Gültigkeit von Prämissen und der gegebenenfalls notwendigen Revision der Prämissen, indem er die Gründe und/oder Ziele für deren Existenz erfasst. Damit adressiert er auch das **dritte Ziel** in Kapitel 4.2, nämlich die Unterstützung bei der Entscheidungsrevision.
Zur besseren Nachvollziehbarkeit von Entscheidungen (**zweites Ziel** in Kapitel 4.2) tragen alle drei Vorschläge bei und sie versuchen auch in gewisser Weise ein Bewusstsein für Rationale in der Organisation zu schaffen (**viertes Ziel** in Kapitel 4.2).
Der Verzicht auf ein Schema, der Verzicht auf spezielle Werkzeuge und die Verwendung von vorhandenen Dokumenten beziehungsweise die Integration von Rationale in diese Dokumente ohne aufwendige Änderungen, soll die Aufdringlichkeit der Vorschläge reduzieren und den Zusatzaufwand für die Aufgaben des Rationale Management so gering wie möglich halten. Dies geschieht vor allem vor dem Hintergrund, dass die Organisation noch keine Erfahrung mit explizitem Rationale Management sammeln konnte und bislang noch keine zusätzlich Ressourcen zur Verfügung stehen (**fünftes Ziel** in Kapitel 4.2). Wie bereits erwähnt, ist allerdings zu beachten, dass eine Reduktion des Aufwandes in einem bestimmten Aufgabenbereich (z.B. Rationaleerfassung und -entwicklung) meist dazu führt, dass der Aufwand in einem anderen Aufgabenbereich (z.B. Rekonstruktionsaufwand bei der Rationalenutzung) steigt. Dies wurde besonders offensichtlich im Zusammenhang mit der Kommunikation von Projektlenkungsentscheidungen.
Dieser Ansatz hat aber auch gezeigt, dass bereits durch eine geringe Anreicherung der vorhandenen Mittel und Dokumente eine nachvollziehbare Entscheidungskommunikation möglich ist, die auch das Rationale zur Verfügung stellt und dadurch auch zusätzliche Akzeptanz schaffen kann. Dies geschieht in diesem Ansatz in erster Linie durch die erreichte Nachverfolgbarkeit (*traceability*) zwischen der Entscheidung und weiteren vorhandenen Informationen, die Rationaleelemente enthalten. Hierdurch wird eine integrierte Sicht auf die Entscheidung ermöglicht und eine Rationalerekonstruktion einfacher. Durch eine entsprechende Archivierung des Decision Log und der Dokumente bleibt dieses „Wissensnetzwerk“ erhalten und ist auch nach längerer Zeit noch zugänglich.
Im Hinblick auf die („Reife-“) Stufen (Kapitel 2.2.2.3) bei der Durchführung von Rationale Management sind die Vorschläge nicht eindeutig einzuordnen. Sie schlagen zwar vor, das Rationale parallel zu den Entscheidungen zu erfassen (Stufe 3: Rationaleerfassung), allerdings erfasst z.B. der Vorschlag im Zusammenhang mit den Projektlenkungsentscheidungen das Rationale nicht explizit, sondern stellt nur Verbindungen zwischen den Entscheidungen und Dokumenten her, die Rationaleelemente (teils implizit) enthalten. Somit müsste hier eher von Stufe 1 (Keine explizite Rationaleerfassung) die Rede sein. Auch drücken sie die Beziehungen zwischen den Rationaleelementen nicht explizit aus.
Es wird nun noch kurz diskutiert, welche Aspekte bei selektiver Umsetzung der drei Vorschläge berücksichtigt werden sollten. Dem Vorschlag zur Anreicherung der Kommunikation von Projektlenkungsentscheidungen (Decision Log) ist zueigen, dass er auch höhere Hierarchieebenen mit einbezieht und eine große Anzahl von Mitarbeitern erreicht. Ein

weiteres Argument für die Umsetzung dieses Vorschlags ist die Tatsache, dass die Nähe der Entscheidungen zur Operations-Abteilung einen schnelleren Verbreitungseffekt haben könnte. Denn in dieser Abteilung ist auch die *Process Governance* (vgl. Fußnote 26) angesiedelt, die bei Bewährung der Rationale Konzepte verstärkt auf deren Positionierung einwirken könnte.
Auch der Ansatz zur Erfassung von Rationale zu den Product Assumptions betrifft eine sehr große Zielgruppe. Er ist deshalb von Bedeutung, da sich ein fehlerhaftes Verständnis der Prämissen beziehungsweise deren fehlende Aktualität und Gültigkeit ungünstig im Produkt widerspiegeln können. Und beidem soll durch den Vorschlag begegnet werden. Da die Product Assumptions zwar in der täglichen Arbeit als Richtlinien gelten aber sich nicht täglich ändern oder neue hinzukommen, kann dieser Vorschlag ohne viel Aufwand zusätzlich zu erstgenanntem hinzugenommen werden und auch schrittweise praktiziert werden, wenn gerade ausreichend Ressourcen zur Verfügung stehen (vgl. Kapitel 4.4.2.2c)). Eine alleinige Umsetzung bietet sich weniger an, da die damit zu sammelnde Erfahrung in der täglichen Erfassung und Entwicklung von Rationale eingeschränkt wäre.
Der Ansatz zur Dokumentation der Gründe für Priorisierungsverschiebungen erreicht in der operativen Anwendung weniger höhere Hierarchieebenen. Er kann jedoch die Position des Verantwortlichkeitsbereichs stärken, der die Kundensicht zur Anforderungserfassung innerhalb des PIL@AP einnimmt. Aufgrund der größeren Auswirkungen auf die unterschiedlichen Rollen und Aktivitäten im Rahmen des PIL@AP sollte aber angesichts der Einführung von Rationale Management die Umsetzung der ersten beiden Vorschläge höhere Priorität haben.

Kapitel 5

Schlussbetrachtung

Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und zieht ein Fazit. Es adressiert damit vor allem die vierte Leitfrage aus Kapitel 1.2, d.h. die Darstellung der erzielten Erkenntnisse bei der Bearbeitung des Themas. Abschließend wird ein Ausblick gegeben, in dem offene Fragen und Herausforderungen für eine Weiterentwicklung der Ergebnisse, die auf den erzielten Erkenntnissen basieren, aufgezeigt werden.
In der vorliegenden Arbeit wurden die im Kontext des Software Engineering diskutierten Ansätze für Rationale Management zusammengetragen. Sie wurden strukturiert nach verschiedenen Wissensbereichen des Software Engineering aufbereitet und gegenübergestellt. Basierend darauf wurden allgemeine Selektionskriterien für die Bewertung von Rationale Ansätzen erarbeitet und unter Beachtung dieser Kriterien und vorher definierter Ziele die Konzepte des Rationale Management auf einen konkreten Praxisfall angewendet.

Die Analyse der Literatur zum Thema Rationale Management offenbarte, dass die am häufigsten diskutierten Rationale Ansätze zu der Gruppe der argumentations-basierten Ansätze gehören und im Bereich des Software Design Anwendung finden. Gezeigt wurde aber ebenfalls, dass Rationale Management nicht auf den Wissensbereich System-/ Produktwissen beschränkt ist.
Argumentations-basiertes Rationale verwendet im Allgemeinen ein semi-formales Schema als Grundlage für die Erfassung und Entwicklung des Rationale. Hierfür werden verschiedene Schemata vorgeschlagen, den Kern bilden aber die Schemata IBIS, QOC und DRL mit jeweils unterschiedlichen Schwerpunkten z.B. bezüglich Umfang des Schemas oder Anwendungskontext. Diese werden auch in den für die praktische Anwendung diskutierten Ansätzen häufig als Grundlage verwendet und gegebenenfalls um bestimmte anwendungsspezifische Konzepte erweitert.
Neben den statischen Aspekten (Rationalerepräsentation) wurden auch die dynamischen Aspekte des Rationale Management erläutert (Prozessimplementierung). Dabei wurde der Zusammenhang von Rationale und Wissensmanagement aufgezeigt, dessen Aufgaben entsprechend für das Rationale Management angepasst werden. Wie auch im Wissensmanagement verschiedene Rollen für spezialisierte Aufgaben existieren, wird dies ebenso im Rahmen von einigen Rationale Ansätzen vorgeschlagen.
Bei der Untersuchung der für die praktische Anwendung diskutierten Ansätze wurde festgestellt, dass sich die meisten Rationale Ansätze sehr stark auf die Rationalerepräsentation und die Aufgabe der Rationaleerfassung konzentrieren. Da die Erfassung des Rationale aufgrund des meist damit verbundenen Zusatzaufwandes eine wesentliche Hemmschwelle für den Einsatz von Rationale Management darstellt, ist dies erst einmal verständlich. Allerdings tritt die Darstellung wichtiger Antworten für eine sinnvolle und optimierte Nutzung und eine darauf abgestimmte Identifikation und Selektion entsprechenden Rationales in den Hintergrund. Die meisten Ansätze geben zwar vor, wie Rationale erfasst werden soll, aber nicht genau welches

aus der möglichen Menge. Dies ist gerade in großen Softwareprojekten ein sehr wichtiger Punkt, der adressiert werden muss, da die Ansätze sonst nicht skalierbar sind.

Die Übertragung der Konzepte des Rationale Management auf den Anwendungsgegenstand hat Folgendes ergeben:

- Für die Kommunikation von Entscheidungen bei der Projektlenkung (Kapitel 4.4.1) bietet Rationale Management durch den Einsatz eines Decision Log und kleine Änderungen bei den Tätigkeiten einer Rolle die Möglichkeit, Projektlenkungsentscheidungen festzuhalten und mit dem entsprechenden Rationale und dem Kontext der Entscheidungen (in vorhandenen Dokumenten) in Beziehung zu setzen. Damit werden die Entscheidungen nachvollziehbar und bei Archivierung des Decision Log und der Dokumente ebenfalls über längere Zeit zugänglich.
- Rationale Management für die zur Erstellung eines Produktes gesetzten Rahmenbedingungen, die in die Organisation hinein kommuniziert werden müssen (Kapitel 4.4.2), ermöglicht ebenfalls das bessere Verständnis der Vorgaben, dient aber gleichzeitig dazu, die Entscheidungen über die Rahmenbedingungen aktuellen Gegebenheiten anzupassen. Und zwar dadurch, dass überprüft werden kann, ob die Begründungen und Zielsetzungen sowie die zugrunde liegenden Annahmen noch gültig sind und noch der aktuellen Ausrichtung des Unternehmens oder Produktes entsprechen.
- Bei der Planung von Releaseinhalten unter Beteiligung von mehreren Parteien kann Rationale verwendet werden, um die Umsetzung von geschäfts- und kundenorientierten Anforderungen zu forcieren (Kapitel 4.4.3). Dies geschieht dadurch, dass bei Nichtumsetzung die Gründe hierfür dokumentiert und einer kritischen Analyse unterzogen werden.

Das in der Arbeit verwendete Selektionskriterium für die Auswahl des zu dokumentierenden Rationale beziehungsweise für die Bereiche, in denen Vorschläge gemacht wurden, war die Anzahl der Personen, die in der Organisation mit der Entscheidung beziehungsweise deren Auswirkungen konfrontiert werden. Dieses Kriterium spiegelt eine Dimension zur Identifikation von wichtigen Entscheidungen wider.
Hohen Stellenwert bei den unterbreiteten Vorschlägen hatte die einfache, nicht-aufdringliche Integration von Rationale Konzepten mit vorhandenen Mitteln. Dies resultiert im vorliegenden Fall in einem informalen Charakter der Vorschläge für den ersten Einführungsschritt von Rationale Management. Sie erfassen das Rationale ohne fest definiertes Schema in offenen Vorlagen oder tabellarischer Form, machen dadurch aber nicht alle Beziehungen zwischen den Rationaleelementen und die damit verbundene Bedeutung explizit. Sie konzentrieren sich bislang erst einmal auf die Erfassung der Gründe von Entscheidungen und können bei Bedarf weiter mit Optionen, Kriterien und Evaluationen etc. angereichert werden.
Eine formalere Erfassung von Rationaleelementen und eine explizite Darstellung der Beziehungen zwischen den Rationaleelementen wird durch eine Werkzeugunterstützung vereinfacht, was in Kapitel 3 an einigen Ansätzen sichtbar wurde. Allerdings muss bei allen Werkzeugüberlegungen beachtet werden, dass eine sinnvolle und nutzenbringende Integration von Rationale Management über einen bloßen Vorschlag für die Repräsentation des Rationale und eine Werkzeugunterstützung (d.h. technische Aspekte) hinausgeht. Sie hat wichtige organisatorische, politische und menschliche Aspekte und muss sich deshalb vielmehr im Bewusstsein und den Prozessen der Organisation niederschlagen und darin verankert werden.
Daher spielt bei der Einführung von Rationale Management auch die Stabilität der Prozesslandschaft eine Rolle. Denn sind Prozesse noch im Aufbau, ist es schwierig, vorherzusagen, an welchen Stellen Rationale einen ausreichenden Nutzen im Vergleich zu dessen Aufwand erzielen kann. In diesem Zusammenhang ist es deshalb gegebenenfalls zielführend, erst einmal an lokalen, stabilen Prozessen punktuell Rationale Management einzuführen, den

Nutzen zu untersuchen und von dort dann auszugehen, um Rationale Management weiter zu verbreiten. Nichtsdestotrotz sollte frühzeitig und parallel bei Neueinführung von Prozessen über die Integration nachgedacht werden, so dass Rationale Management ein inhärenter Teil der Prozesse werden kann und nicht nur als „unliebsamer" Zusatz später „nur aufgesetzt" wird. Aus den gewonnenen Erfahrungen heraus muss dann auch bereits bei neuen Prozessen darauf geachtet werden, wie Rationale Management gleich berücksichtigt werden kann.
Vor diesem Hintergrund sind die Untersuchungen der Arbeit als wesentliche Voraussetzung für die weitere Verfolgung der Thematik des Rationale Management im Allgemeinen und auch im Kontext der Anwendungsbeispiele zu sehen. Denn die Vorschläge haben deutlich gemacht, wie verschiedene Ziele des Rationale Management in der industriellen Softwareentwicklung umgesetzt werden und dadurch die Zusammenarbeit und Kommunikation sowie die Entscheidungsfindung und -transparenz unterstützt und verbessert werden können. Nächste Schritte für eine Weiterentwicklung der Vorschläge und Weiterbeschäftigung mit dem Thema, die in der vorliegenden Arbeit nicht erschöpfend behandelt werden konnten, sind im Folgenden genannt:

- In den Prozessen, in denen Rationale Management nachhaltig eingeführt werden soll, sollten die wichtigen und damit kritisch zu dokumentierenden Entscheidungstypen identifiziert werden. Darauf basierend sollten die für die Entscheidungen relevanten Kriterien herausgearbeitet und als Grundmenge zur Alternativenevaluation bei diesen Entscheidungen zur Verfügung gestellt werden.
- Sind die zu dokumentierenden Entscheidungen und Kriterien sowie die Prozesse, in denen die Entscheidungen getroffen und nachträglich verwendet werden, analysiert, sollte verstärkt ein semi-formales Schema erarbeitet werden, welches die Beziehungen der einzelnen Rationaleelemente explizit macht, z.B. in Form einer Vorlage (*template*).
- Um Rationale Management in großem Stil einzusetzen, ist auch eine integrierte Werkzeugunterstützung für die Verwaltung des Rationale und die Ausführung der Aufgaben vorteilhaft. Hier besteht nun – aufbauend auf den Resultaten der Arbeit im Rahmen des angewendeten Softwareentwicklungsprozessmodells – die Chance, Rationale Management Aspekte stärker im Design der Werkzeuglandschaft zu berücksichtigen, denn diese Landschaft befindet sich gerade in einer Umbruchphase hin zu einer verstärkten Integration und Stabilisierung über den gesamten Softwarelebenszyklus. Dennoch muss eine Lösung erarbeitet werden, die auch einer starken Verwendung von Office Anwendungen Rechnung trägt. Ein Vorschlag in diese Richtung wäre eine Datenbankanwendung, die es erlaubt, Rationale, basierend auf den identifizierten Rationaleschemata, zu speichern. Um das Rationale mit den entsprechenden Entscheidungen und Artefakten zu verbinden, zu denen es erfasst wurde, müsste dem gespeicherten Rationale ein Identifikator zugeordnet werden, welcher mit der Entscheidung beziehungsweise dem Artefakt abgelegt wird, um von dort (wieder) zu dem entsprechenden Rationale zu gelangen.

Auch wenn die Weiterentwicklungen die Entscheidungsfindung zusätzlich unterstützen und außerdem weitere Transparenz, Nachvollziehbarkeit und Verständlichkeit im Hinblick auf getroffene Entscheidungen bieten, hat die vorliegende Arbeit gezeigt, dass dies bereits durch einfache Mittel und kleine Änderungen in herkömmlichen Prozessen und Dokumenten erzielt werden kann. Dies gestattet, Rationale Management schrittweise in bestehende und in im Aufbau befindliche Prozesslandschaften einzubeziehen und ein Bewusstsein für Rationale erst einmal zu schaffen und dann weiter auszubauen.

Anhang

User Tasks und Use Cases für die Kommunikation von Projektlenkungsentscheidungen (Decision Log) (Kapitel 4.4.1)

Aufgabenbeschreibung (User Task)	
Name:	Kommunikation von Managemententscheidungen
Verantwortliche Rolle (Initiating Actor):	Entscheidungskommunikator
Beteiligte Rollen (Participating Actors):	Umsetzende (Adressaten der Entscheidungen)
Aufgabenbewertung	
Ziel (Goal):	Umsetzende zur Ausführung ihrer Aufgaben (resultierend aus den Entscheidungen) bewegen
Eingriffsmöglichkeiten:	Nutzung des Kommunikationsmediums (Decision Log, Informations-E-Mail) Filterung/Selektion der Entscheidungen, welche kommuniziert werden sollen, nach folgenden Kriterien: (1) Relevanz für die gesamte (Entwicklungs-) Organisation des Programms (2) Politischer Charakter der Entscheidung (Zielgruppe: manche Entscheidungen sind nur gedacht für die Beteiligten der Treffen → Verantwortlichen)
Ursachen:	Die Umsetzenden sind nicht an den Treffen beteiligt, in denen die Entscheidungen getroffen werden, sollen aber darüber informiert werden und Entscheidungen umsetzen
Priorität:	Hoch, da die getroffenen Entscheidungen der Programm-/Projektsteuerung dienen
Aufgabendurchführung	
Durchführungsprofil (Häufigkeit, Kontinuität, Komplexität):	*Häufigkeit*: wöchentlich bis täglich *Kontinuität*: kann unterbrochen werden *Komplexität*: mittel, da die Information/Entscheidung aus den Treffen für die Kommunikation aufbereitet werden muss
Ausgangssituation (Vorbedingung):	In den Treffen wurden Entscheidungen getroffen und die zu kommunizierenden Entscheidungen liegen vor
Info-In:	Getroffene Entscheidungen Zur Entscheidung vorhandene Dokumente (Präsentationen, Protokolle etc.)
Info-Out:	Kommunizierte Entscheidungen
Ressourcen (z. B. Arbeitsmittel etc.):	Microsoft Office, Publikationswerkzeug für das Intranet

Tabelle 12: User Task „Kommunikation der Managemententscheidungen“

Interaktionsbeschreibung (Use Case)		
Name:	Verfassen einer Informations-E-Mail	
Verantwortliche Rolle (Initiating Actor):	Entscheidungskommunikator	
Beteiligte Rollen (Participating Actors):	-	
Ziel (Goal):	Umsetzende mit der E-Mail über ihre Aufgaben informieren und sie zur Ausführung der Aufgaben bewegen So genannten „Push-Mechanismus“ etablieren, der die Umsetzenden schnell über die Entscheidungen informiert	
Vorbedingung (Preconditions):	Die zu kommunizierenden Entscheidungen aus den Treffen liegen vor	
Beschreibung (Flow Of Events):	**Aktor**	**System**
	(1) Selektieren/Filtern der Entscheidungen, die kommuniziert werden sollen (2) Formulierung der Entscheidungen als E-Mail (3) Versand der E-Mail	
Ausnahmefälle (Exceptions):	-	
Nachbedingungen (Postconditions):	E-Mail wurde versandt und ist bei den Adressaten angekommen	
Regeln (Rules):	-	
Qualitätsanforderung (Quality Constrains):	Kurze und prägnante Formulierung der Entscheidungen Verständlichkeit der Entscheidungen und des Kontextes	

Tabelle 13: Use Case „Verfassen einer Informations-E-Mail“

Interaktionsbeschreibung (Use Case)		
Name:	Pflegen des Decision Log	
Verantwortliche Rolle (Initiating Actor):	Entscheidungskommunikator	
Beteiligte Rollen (Participating Actors):	-	
Ziel (Goal):	Integrierte (chronologische) Sicht auf Entscheidungen „Pull-Mechanismus“ etablieren, durch den „Interessierte“ sich über Entscheidungen informieren können und ggf. weitere Informationen erhalten	
Vorbedingung (Preconditions):	Die zu kommunizierenden Entscheidungen aus den Treffen liegen vor	
Beschreibung (Flow Of Events):	**Aktor**	**System**
	(1) Selektieren/Filtern der Entscheidungen, die kommuniziert werden sollen (2) Formulierung der Entscheidungen (3) Aufnahme der Formulierung und der weiteren Informationen in die Tabelle (4) Aufnahme einer Informations-E-Mail auf die Intranetseite des Decision Log (optional) (5) Publikation des Decision Log	
Ausnahmefälle (Exceptions):	-	
Nachbedingungen (Postconditions):	Der Decision Log wurde publiziert, enthält alle aktualisierten Daten und ist für Adressaten (Umsetzende) zugänglich	
Regeln (Rules):	-	
Qualitätsanforderung (Quality Constrains):	Kurze und prägnante Formulierung der Entscheidungen Verständlichkeit der Entscheidungen und des Kontextes Vollständigkeit der Informationen im Decision Log	

Tabelle 14: Use Case „Pflegen des Decision Log“

Beispiel für die Abbildung/Erfassung einer Diskussion mittels des Compendium Werkzeugs (Kapitel 4.4.1.2)

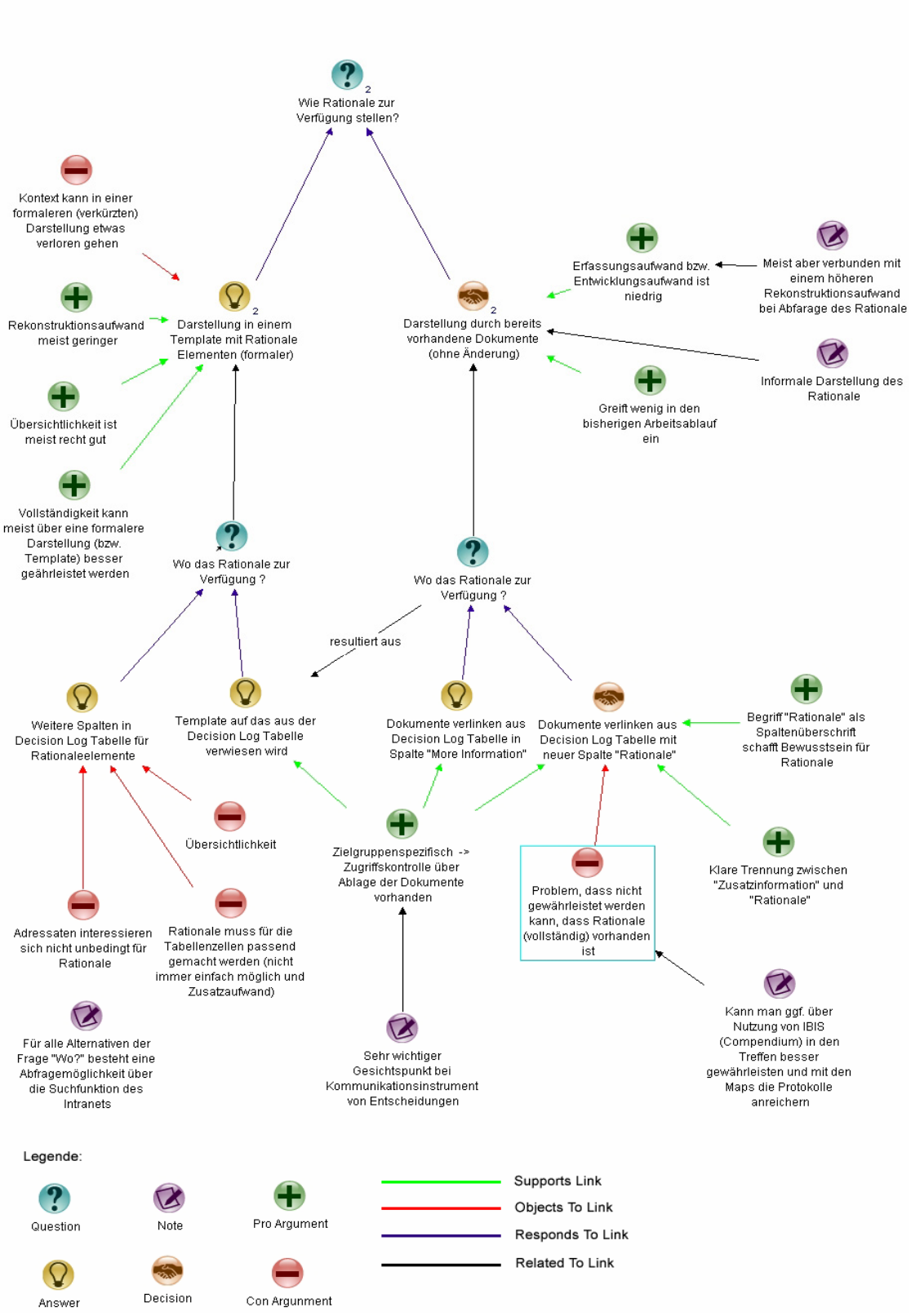

Abbildung 21: Diskussion aus Kapitel 4.4.1.2 mittels des Compendium Werkzeugs dargestellt

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Balz98 BALZERT, H. (1998): Lehrbuch der Software-Technik. Band 2: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Spektrum. Heidelberg, Berlin.

Balz00 BALZERT, H. (2000): Lehrbuch der Software-Technik. Band 1: Software-Entwicklung. 2. Auflage. Spektrum. Heidelberg, Berlin.

Boeh88 BOEHM, B. W. (1988): A Spiral Model of Software Development and Enhancement. In: *Computer*, Volume 21, Issue 5, (May 1988), S. 61-72. (http://www.sce.carleton.ca/faculty/ajila/4106-5006/Spiral%20Model%20Boehm.pdf am: 13.04.2007)

BrJR00 BRATTHALL, L.; JOHANSSON, E.; REGNELL, B. (2000): Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution. In: *Proceedings of the Second international Conference on Product Focused Software Process Improvement* (PROFES'00) (Oulu, Finland, June 20-22, 2000, LNCS 1840), S. 126-139.

BSS+06 BUCKINGHAM SHUM, S. J.; SELVIN, A. M.; SIERHUIS, M.; CONKLIN, J.; HALEY, C. B.; NUSEIBEH, B. (2006): Hypermedia Support for Argumentation-Based Rationale: 15 Years on from gIBIS and QOC. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 111-132. (http://kmi.open.ac.uk/publications/pdf/KMI-05-18.pdf am: 13.04.2007)

BuBr03 BURGE, J. E.; BROWN, D. C. (2003): Rationale Support for Maintenance of Large Scale Systems. In: http://citeseer.ist.psu.edu/burge03rationale.html am: 19.03.2007

BuBr06 BURGE, J. E.; BROWN, D. C. (2006): Rationale-Based Support for Software Maintenance. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 273-296.

BuBroJ BURGE, J. E.; BROWN, D. C. (o.J.): An Integrated Approach For Software Design Checking Using Design Rationale. In: http://web.cs.wpi.edu/~dcb/Papers/DCC-paper-04.pdf am: 19.03.2007

Burg05 BURGE, J. E. (2005): Software Engineering Using design RATionale. Ph.D. thesis, Worcester Polytechnic Institute. In: http://www.wpi.edu/Pubs/ETD/Available/etd-050205-085625/unrestricted/BurgeDissertation.pdf am: 19.03.2007

Carr06 CARROLL, J. M. (2006): Foreword. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. VII-VIII.

Chan01 CHANG, S. K. (Hrsg.) (2001): Handbook of Software Engineering and Knowledge Engineering. Volume 1. Fundamentals. World Scientific Publishing Company. Singapore.

Das Handbuch ist auch im Internet zugänglich:
http://www.ksi.edu/seke/hand.html am: 20.03.2007

CoBe88 CONKLIN, J.; BEGEMAN, M. L. (1988): glBIS: A Hypertext Tool for Exploratory Policy Discussion. In: *Proceedings of the 1988 ACM Conference on Computer-Supported Cooperative Work* (CSCW '88) (Portland, Oregon, United States, September 26-28, 1988), S. 140-152.

DeLo Decision Log. (SAP Intranetseite).
(Internes SAP Dokument)

DISZ04 DILLINGER, A., ILLGNER, H., SIEVI, O., ZIMMERMANN, D. (2004): Product Innovation Lifecycle. From Ideas to Customer Value. Whitepaper Version 1.1. July 2004. External Version. Released for Customers & Partners.
(SAP Dokument)

DMMP06a DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.) (2006): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg.

DMMP06b DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (2006): Rationale Management in Software Engineering: Concepts and Techniques. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 1-48.

DuPa00 DUTOIT, A. H.; PAECH, B. (2000): Supporting Evolution: Rationale in Use Case Driven Software Development. International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'2000), Stockholm, June, 2000. In:
http://citeseer.ist.psu.edu/dutoit00supporting.html am: 20.03.2007

DuPa01 DUTOIT, A. H.; PAECH, B. (2001): Rationale Management in Software Engineering. In: CHANG, S. K. (Hrsg.): Handbook of Software Engineering and Knowledge Engineering. Volume 1. Fundamentals. World Scientific Publishing Company. Singapore. S. 787-815.
(ftp://cs.pitt.edu/chang/handbook/20.pdf am: 13.04.2007)

FlVi01 FLURRY, G.; VICKNAIR, W. (2001): The IBM application framework for e-business. In: *IBM Systems Journal* Volume 40, Issue 1 (Jan. 2001), S. 8-24.

Fowl05 FOWLER, M. (2005): The New Methodology. In:
http://www.martinfowler.com/articles/newMethodology.html am: 20.02.2007

GDS+07 GEBHARD, M.; DILLINGER, A.; SALZMANN, D.; TUSHARA, R.; ILLGNER, H.; GREINER, E.; RUDOLF, E. (2007): Product Innovation Lifecycle at AP. Process Model for SAP's Enterprise Service-Oriented Architecture based Product Suite – From Breakthrough Innovation to Standardized Product Development. Whitepaper. Version 0.9 – For Review. March 16, 2007.
(Internes SAP Dokument)

Gebh07a GEBHARD, M. (2007): Software-Anforderungsbasierte Nachverfolgung von Produkteigenschaften in einer modelbasierten Entwicklungslandschaft. Umsetzung der Verfolgung von Kundenanfragen bis zur Verifikation der Implementierung bei der Entwicklung der SAP Applikationsplattform. Preprint April 2007.
(SAP Dokument)

Gebh07b GEBHARD, M. (2007): Software Requirements Specification. Process Description. Version 1.6.0.
(Internes SAP Dokument)

GeDi06 GEBHARD, M.; DILLINGER A. (2006): Requirements Inspections – how to proceed. Februar 2007. Version 1.2.0. February 26, 2007.
(Internes SAP Dokument)

GHJV95 GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. (1995): Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley. Amsterdam.

Grud96 GRUDIN, J. (1996): Evaluating Opportunities for Design Capture. In: MORAN, T. P.; CARROLL, J. M. (Hrsg.): Design Rationale. Concepts, Techniques, and Use. Lawrence Erlbaum Associates. Mahwah, New Jersey, S. 453-470.

Haef05 HÄFELE, P. (2005): Softwareentwicklung mit dem TRAIN-Prozess. Bachelor Thesis. Institut für Informatik, Lehrstuhl Software Systeme, Ruprecht-Karls-Universität Heidelberg. In:
http://www-swe.informatik.uni-heidelberg.de/research/publications/TRAIN.pdf am: 04.04.2007

HePa06 HERRMANN, A.; PAECH, B. (2006): Lernen aus dokumentierten Architektur-Entscheidungen. In: Softwaretechnik-Trends 26:4, Nov. 2006, S. 22-27.
(http://pi.informatik.uni-siegen.de/stt/26_4/01_Fachgruppenberichte/ORA2006/06_herrmann-final.pdf am: 20.03.2007)

Herr06 HERRMANN A. (2006): Wissensmanagement und Entscheidungen im Software Engineering (SWE IIc). Teil 1 – Einführung, Folie 28f.
Skript zur Vorlesung "Wissensmanagement und Entscheidungen im Software Engineering (SWE IIc)" gehalten im Wintersemester 2006/2007 an der Ruprecht-Karls-Universität Heidelberg, Institut für Informatik, Lehrstuhl für Software Engineering.

HHLP06 HAGGE, L.; HOUDEK, F.; LAPPE, K.; PAECH, B. (2006): Using Patterns for Sharing Requirements Engineering Process Rationales. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 409-427.

HoAt06 HORNER, J.; ATWOOD, M. E. (2006): Effective Design Rationale: Understanding the Barriers. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 73-90.

Hohm03 HOHMANN, L. (2003): Beyond Software Architecture. Creating and Sustaining Winning Solutions. Addison-Wesley. Amsterdam.

HoRM02 HOWARD, R.; RICHARDSON, J.; METCALFE, D. (2002): Negotiation analysis. The science and art of collaborative decision making. Belknap. Cambridge, Massachusetts.

Howa68 HOWARD, R. (1968): Decision analysis. Introductory lectures on choices under uncertainty. Addison-Wesley. Reading, Massachusetts.

ISO9126 ISO (2001): ISO/IEC 9126-1:2001. Software engineering – Product quality – Part 1: Quality model. International Organization for Standardization. Geneva.

Kars96 KARSENTY, L. (1996): An empirical evaluation of design rationale documents. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Common Ground* (Vancouver, British Columbia, Canada, April 13-18, 1996). S. 150-156.

KRKO03 KARLSSON, L.; REGNELL, B.; KARLSSON, J.; OLSSON, S. (2003): Post-Release Analysis of Requirements Selection Quality – An Industrial Case Study. International Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ'03), Klagenfurt/Velden, Austria, June 2003. In: http://serg.telecom.lth.se/research/publications/docs/187_Post-release.pdf am: 20.03.2007

KuRi70 KUNZ W., RITTEL H. W. J.: (1970): Issues as elements of information systems. Working Paper No. 131. Reprinted 1979. Center for Urban and Regional Development, University of California, Berkeley. In: http://www-iurd.ced.berkeley.edu/pub/WP-131.pdf am: 20.03.2007

KyKa99 KYARUZI, J. K.; VAN KATWIJK, J. (1999): Beyond Components-Connections-Constraints: Dealing with Software Architecture Difficulties. In: *Proceedings of the 14th IEEE international Conference on Automated Software Engineering* (Cocoa Beach, Florida, United States, October 12-15, 1999). S. 235.

LeDo03 LEITE, J.; DOORN, J. (Hrsg.) (2006): Perspectives on Requirements Engineering, Kluwer Academic Publishers. Dordrecht.

Lee90 LEE, J. (1990): SIBYL: a tool for managing group design rationale. In: *Proceedings of the 1990 ACM Conference on Computer-Supported Cooperative Work* (CSCW'90) (Los Angeles, California, United States, October 07-10, 1990). S. 79-92.

Lee97 LEE, J. (1997): Design Rationale Systems: Understanding the Issues. In: *IEEE Expert: Intelligent Systems and Their Applications*. Volume 12, Issue 3 (May 1997), S. 78-85.

LeFe06 LEHMAN, M. M.; FERNÁNDEZ-RAMIL, J (2006): The Role and Impact of Assumptions in Software Engineering and its Products. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 313-328.

LeLa91 LEE, J.; LAI, K.-Y. (1991): What's in Design Rationale? In: *Human-Computer Interaction*, Volume 6, Issue 3&4 (1991), S. 251-280.

LoLo00 LOURIDAS, P.; LOUCOPOULOS, P. (2000): A Generic Model for Reflective Design. In: *ACM Transactions on Software Engineering and Methodology* (TOSEM), Volume 9, Issue 2 (April 2000), S. 199-237. (http://delivery.acm.org/10.1145/360000/350895/p199-louridas.pdf?key1=350895&key2=8576646711&coll=GUIDE&dl=GUIDE,ACM&CFID=11111111&CFTOKEN=2222222 am: 13.04.2007)

MaBr02 MALAN, R.; BREDEMEYER, D. (2002): Less is More with Minimalist Architecture. In: *IT Professional* Volume 4, Issue 5 (Sep. 2002), S. 46-47. (http://www.bredemeyer.com/pdf_files/MinimalistArchitecture.PDF am: 20.03.2007)

McCa91 MCCALL, R. J. (1991): PHI: A Conceptual Foundation for Design Hypermedia. In: *Design Studies*, Volume 12, Issue 1 (January 1991), S. 30-41.

Meye73 Artikel Entscheidung. In: Meyers Enzyklopädisches Lexikon, Bd. 7. Bibliographisches Institut. Mannheim, Wien, Zürich. 1973. S. 845-846.

MoCa96a MORAN, T. P.; CARROLL, J. M. (Hrsg.) (1996): Design Rationale. Concepts, Techniques, and Use. Lawrence Erlbaum Associates. Mahwah, New Jersey.

MoCa96b MORAN, T. P.; CARROLL, J. M. (1996): Overview of design rationale. In: MORAN, T. P.; CARROLL, J. M. (Hrsg.) (1996): Design Rationale. Concepts, Techniques, and Use. Lawrence Erlbaum Associates. Mahwah, New Jersey, S. 1-19.

MYBM91 MACLEAN, A.; YOUNG, R. M.; BELLOTTI, V. M.; MORAN, T. P. (1991): Questions, Options, and Criteria: Elements of Design Space Analysis. In: *Human-Computer Interaction*, Volume 6, Issue 3&4 (1991), S. 201-250.

Paec06 PAECH, B. (2006): Rationale for Organizing Bodies of Knowledge. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 349-352.

PaKo03 PAECH, B.; KOHLER, K. (2003) Task-driven Requirements in object-oriented Development. In: LEITE, J.; DOORN, J. (Hrsg.): Perspectives on Requirements Engineering, Kluwer Academic Publishers. Dordrecht.

PaRi06 PALYAGAR, B.; RICHARDS, D. (2006): Capturing and Reusing Rationale Associated with Requirements Engineering Process Improvement: A Case Study. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 391-408.

PAss Application Platform. Product Assumptions. Version 1.25 – Revised Final Version.
(Internes SAP Dokument)

PoBr88 POTTS, C.; BRUNS, G. (1988): Recording the reasons for design decisions. In: *Proceedings of the 10th International Conference on Software Engineering* (Singapore, Apr. 11-15), S 418-427.

Pott89 POTTS, C. (1989): A generic model for representing design methods. In: *Proceedings of the 11th International Conference on Software Engineering* (Pittsburgh, PA, May 15-18), S 217-226.

RaDh92 RAMESH, B.; DHAR V. (1992): Supporting Systems Development by Capturing Deliberations During Requirements Engineering. In: *IEEE Transactions on Software Engineering*, Volume 18, Issue 6 (June 1992), S. 498-510.

REPARE The Requirements Engineering Pattern Repository, „Exchanging Requirements Engineering Experience", http://repare.desy.de

RoSP06 ROOKSBY, J.; SOMMERVILLE, I.; PIDD, M. (2006): A Hybrid Approach to Upstream Requirements: IBIS and Cognitive Mapping. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 137-154.

Royc70 ROYCE, W. W. (1970): Managing the Development of large Software Systems. In: *Proceedings, IEEE WESCON* (August 1970), S. 1-9. (http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf am: 13.04.2007)

RuLS01 RUS, I.; LINDVALL, M.; SINHA, S. S. (2001): Knowledge Management in Software Engineering. A State-of-the-Art-Report. Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland. In: http://www.cebase.org:444/umd/dacs_reports/kmse_-_nicholls_final_edit_11-16-01.pdf am: 20.03.2007

Schn06 SCHNEIDER, K. (2006): Rationale as a By-Product. In: DUTOIT, A. H.; MCCALL, R., MISTRÍK, I., PAECH, B. (Hrsg.): Rationale Management in Software Engineering. Springer-Verlag. Berlin, Heidelberg, S. 91-109.

ShMc96 SHIPMAN III, F. M.; MCCALL, R. J. (1996): Integrating Different Perspectives on Design Rationale: Supporting the Emergence of Design Rationale from Design Communication. In: http://www.csdl.tamu.edu/csdl/trpubs/csdl96001.pdf am: 02.02.2007

Somm01 SOMMERVILLE, I. (2001): Software Engineering. 6. Auflage, Pearson Studium. München.

SPICE SPICE, Software Process Improvement and Capability dEtermination Website, http://www.sqi.gu.edu.au/spice/

Stac73 STACHOWIAK, H. (1973): Allgemeine Modelltheorie. Springer-Verlag. Wien, New York.

TyAk05 TYREE, J.; AKERMAN, A. (2005): Architecture Decisions: Demystifying Architecture. In: *IEEE Software.* Volume 22, Issue 2 (March 2005), S. 19-27.

UML UML Resource Page, http://www.uml.org/

WoAu05 WOHLIN, C.; AURUM, A. (2005): What is important when deciding to include a software requirement in a project or release? In: *Proceedings of the International Symposium on Empirical Software Engineering* (ISESE'2005), (Noosa Heads, Australia, November 17-18, 2005), S. 237-246.

WoDu04 WOLF, T.; DUTOIT, A. H. (2004): A Rationale-based Analysis Tool. 13th International Conference on Intelligent and Adaptive Systems and Software Engineering, July 1-3, Nice, France. (http://wwwbruegge.in.tum.de/static/publications//pdf/136/wolf2004rat.pdf am: 13.04.2007)

WoDuoJ WOLF, T.; DUTOIT, A. H. (o.J.): Sysiphus: Combining System Modeling with Collaboration and Rationale. In: http://pi.informatik.uni-siegen.de/stt/24_4/01_Fachgruppenberichte/11wolf.pdf am: 15.03.2007

WoMa06 WOODS, D.; MATTERN, T. (2006): Enterprise SOA – Designing IT for Business Innovation. O'Reilly. Beijing, Köln.

WSHE Worksheet in Product Assumptions In: Application Platform. Product Assumptions. Version 1.25 – Revised Final Version. S. 95. *(Internes SAP Dokument)*

Erklärung

Hiermit versichere ich, Philipp Häfele, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Heidelberg, den 30.04.2007