Ein Modellbasierter Ansatz zur Lösung Komplexitätsbedingter Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik

Von der Fakultät Wirtschafts- und Sozialwissenschaften der Universität Stuttgart zur Erlangung der Würde eines Doktors der Wirtschafts- und Sozialwissenschaften (Dr. rer. pol.) genehmigte Abhandlung

Vorgelegt von

Torsten Langner

aus Würselen

Hauptberichter:	Prof. Dr. Georg Herzwurm
Mitberichter:	Prof. Dr. Hans-Georg Kemper
Tag der mündlichen Prüfung:	23. Oktober 2007

Betriebswirtschaftliches Institut der Universität Stuttgart

2007

Geleitwort

Banken und Versicherungen stehen oft vor dem Problem, dass ihre IT-Infrastruktur aus einer Vielzahl historisch gewachsener Systeme besteht, die großteils unternehmenskritische Funktionen übernehmen. Folglich werden diese Systeme nicht abgeschaltet, sondern teilweise über Jahrzehnte hinweg weiterentwickelt. Dies lässt den Aufwand für Wartung und Betrieb exponentiell ansteigen.

Wissenschaftler des Betriebswirtschaftlichen Instituts der Uni Stuttgart untersuchten im Rahmen dieses von der Postbank Systems AG in Auftrag gegebenen, zweijährigen Forschungsprojekts, wie Architekturmuster den Entwurf der IT-Infrastruktur vereinfachen und so den Wildwuchs begrenzen können. Die Ergebnisse tragen dazu bei, dass Firmen wie die Postbank mithilfe von Informationstechnologie einerseits kostengünstig Leistungen anbieten und andererseits schnell und flexibel neue Leistungen am Markt einführen können. Mithilfe des Standard-Infrastrukturkatalogs können IT-Architekten einerseits die beschleunigte Entwicklung oder Anschaffung neuer Systeme unterstützen. Andererseits können Altsysteme behutsam auf Standardbausteine umgestellt werden. Somit hat die Entwicklung und kontinuierliche Pflege der Architekturmuster für den IT-Infrastrukturentwurf wesentlich zur qualitativen Verbesserung beigetragen und das Selbstverständnis eines IT-Architekten geschärft.

Dr. Thomas Mangel
Vorstand für Technologiemanagement
Postbank Systems AG, Bonn

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AM	Architekturmuster
CCTA	Central Computer and Telecommunications Agency
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CSP	Constraint Satisfaction Problemen
DoD	Department of Defense
DSL	Domain Specific Language
DSM	Design Structure Matrix
GE	Geldeinheiten
HA	High Availability
ICT	Information and Communications Technology
ICTIM	ICT Infrastructure Management
IEEE	Institute of Electrical and Electronics Engineers
IKT	Informations- und Kommunikationstechnik
IPD-CMM	Integrated Product Development CMM
IPPD	Integrated Product and Process Development
ISO	International Organization for Standardization
IT	Information Technology
ITIL	IT Infrastructure Library
J2EE	Java 2, Enterprise Edition
MDA	Model Driven Architecture
MIM	Module Indication Matrix
OG	Obergruppe
OGC	Office of Government Commerce
PBS	Postbank Systems AG
PSM	Problem Solving Method
PSQM	Prozessorientiertes Software-Qualitätsmanagement
QFD	Quality Function Deployment
RUP	Rational Unified Process
SE	Software Engineering
SECM	Systems Engineering Capability Model

SE-CMM	Systems Engineering Capability Maturity Model
SEI	Software Engineering Institute
SI	Standard Infrastruktur
SI-KK	SI Katalog Konzept
SLA	Service Level Agreement
SPICE	Software Process Improvement and Capability dEtermination
SUV	Sport Utility Vehicles
TCO	Total Cost of Ownership
UML	Unified Modeling Language

Zusammenfassung

Diese Arbeit stellt ein begründetes Gestaltungskonzept vor, das die Infrastrukturarchitekten eines Finanzdienstleisters beim Entwurf neuer IT-Architekturen unterstützt. Das Gestaltungskonzept fokussiert primär die drei Problemdimensionen *Komplexität der Infrastrukturarchitektur*, *Wissens-komplexität und Wissensvolumen* sowie *Historische Entwicklung der Finanzdienstleistungsinformatik*.

Durch die Adaption des Wiederverwendungskonzepts reduziert das vorgestellte Konzept zunächst die Komplexität des Architekturentwurfs im Einzelnen, um später die Komplexität der Infrastruktur im Ganzen reduzieren zu können. Hierzu wird die historisch gewachsene IT-Architekturlandschaft eines Finanzdienstleisters erfasst, dekomponiert und kategorisiert. In einem zweiten Schritt wird aus der Vielzahl der Infrastrukturkomponenten eine Selektion der zukünftig zu verwendenden Komponenten ausgewählt und durch qualitative Eigenschaften beschrieben. Die einzelnen Infrastrukturkomponenten werden in einem dritten Schritt zu qualitativ beschriebenen Architekturmustern komponiert, die den Plattformen des klassischen Maschinenbaus ähneln und der Konfiguration neuer Architekturen dienen. Zur Darstellung der Zusammenhänge der Infrastrukturarchitektur wird ein Meta-Modell entwickelt, dessen Anwendung durch ein Prozessmodell beschrieben wird.

Das Prozessmodell besteht aus den drei entkoppelten Phasen *Einführung*, *Anwendung* und *Pflege* der Methodik. Der Eintritt in eine jeweilige Phase ist Ereignisgesteuert, so dass eine kontinuierliche Pflege der Architekturmuster erfolgt. Das Ergebnis des wiederholten Phasendurchlaufs ist ein Standard-Infrastruktur-Katalog, der zum einen die zu konfigurierbaren Architekturmustern komponierten, vorselektierten Infrastrukturkomponenten qualitativ (d. h. typisch, architektonisch und leistungstechnisch) beschreibt, und der zum anderen den Infrastrukturkomponenten und Architekturmustern strategische Entscheidungsregeln zuweist.

Abstract

This document presents a reasonable methodology with the indent to assist IT infrastructure architects creating new architectural blueprints. The methodology primarily focuses on the following three dimensions of the research problem: *complexity of the IT infrastructure architecture*, *knowledge complexity* and *evolution of information management in Financial Services*.

Adapting the concept of reuse in general the introduced concept reduces the complexity of an individual architectural blueprint first before reducing the complexity of the IT infrastructure enterprise wide. To achieve this goal, the concept identifies the the historical groth of the IT infrastructure of a financial services institute, then decomposes and categorizes it into its most fundamental building blocks.

The second step involves identifying key, sustainable infrastructure components then qualifying and classifying them by quality attributes. Within a third step the concept composes the preselected infrastructure components to architectural patterns that are similar to the platforms known in classical engineering and that are used to configure new architectures. To describe the coherences inside the architectural blueprint the introduced concept creates a meta model which appliance is described by a process model.

The process model consists of three decoupled stages: *initialization*, *utilization* and *maintenance* of the methodology. Entering a stage is event-driven. Thus, the continuous maintance of architectural patterns is guaranteed. The result of a repeated stage passing is a standardized infrastructure catalogue which describes the configurable architectural patterns composed by preselected infrastructure components on a qualitative level (i. e. type based, architecture based and capacity based). Additionally the methodology presented assigns strategic decision rules to infrastructure components and architectural patterns.

1 Einleitung

Der Entwurf von IT-Infrastrukturarchitekturen bei Finanzdienstleistern ist komplex. Es fehlt an einem begründeten Gestaltungskonzept, das zum einen die Entscheidung für oder gegen eine Infrastrukturarchitektur unterstützt, und das zum anderen die Komplexität des Architekturentwurfs im Einzelnen sowie die Komplexität der Infrastruktur im Ganzen reduziert. Im klassischen Maschinenbau werden schon seit langer Zeit Plattformen als Architekturen von Ingenieuren für Ingenieure eingesetzt. Ziel dieser Plattformen ist die Reduktion der Variantenvielfalt der für den Bau verwendeten Komponenten. Da Modularisierung und Komponentenbau im Software Engineering bekannt sind, ist zu überlegen, inwieweit diese Konzepte den Infrastrukturarchitekturentwurf in der Finanzdienstleistungsinformatik beeinflussen können.

1.1 Finanzdienstleistungsinformatik

Informatik ist die Wissenschaft, Technik und Anwendung der systematischen und automatischen Verarbeitung von Informationen.[1] Als Finanzdienstleister wird in Anlehnung an KRUPP ein Unternehmen bezeichnet, das entweder ein Finanzdienstleistungsprodukt, ein Versicherungsprodukt oder ein Bausparprodukt anbietet.[2] Beispiele für Finanzdienstleistungsprodukte sind das Einlagen-, Wertpapier- oder Kreditgeschäft.

Der Markt der Finanzdienstleister erlebt zurzeit einen strukturellen Wandel. Neben den grenzüberschreitenden Bankenfusionen im europäischen Raum finden Übernahmen nicht nur unter Banken statt.[3] Ein Beispiel hierfür ist die Übernahme der Dresdner Bank durch die klassisch als Versicherer aufgestellte Allianz Gruppe.[4] Dieses und weitere Beispiele[5] zeigen, dass das Finanzdienstleistungsgeschäft vom klassi-

[1] Vgl. Moormann (2004)
[2] Vgl. Krupp (1995)
[3] Anmerkung: Nogueira/Reinhard (2000) berichten von ähnlichen Beobachtungen im Jahr 2000 auf dem südamerikanischen Kontinent.
[4] Vgl. Fuchs (2003), S. 8ff
[5] Vgl. Fuchs (2003), S. 8ff

schen Bankgeschäft aus stetig um andere Produkte und Branchen erweitert wird.

Aus diesem Grund wird die in der Literatur[6] als *Bankinformatik* bezeichnete Disziplin der Informatik innerhalb Banken im Rahmen dieser Arbeit als *Finanzdienstleistungsinformatik* bezeichnet. Das Leistungsspektrum der Finanzdienstleistungsinformatik lässt sich in drei Bereiche einteilen:[7] Erstens muss die Informatik eine der Geschäftsstrategie entsprechende und wirtschaftliche Infrastruktur aufbauen und unterhalten. Zweitens muss sie die Bedürfnisse der Kunden im Front und Back Office Bereich qualitativ hochwertig abdecken. Drittens sollen die gewünschten Führungsinformationen frist- und stufengerecht bereitgestellt werden.

Da sich die Wirtschaftsinformatik in Anlehnung an KRCMAR und HEINRICH mit der Entwicklung von Informationssystemen in Form von Mensch-Aufgabe-Technik Systemen befasst,[8] wird die Finanzdienstleistungsinformatik im Rahmen dieser Arbeit als ein Spezialisierungsgebiet der Wirtschaftsinformatik angenommen.

1.2 Zentrale Problemstellung

Die zentrale Problemstellung, mit der sich diese Arbeit auseinandersetzt, setzt sich aus drei Problemdimensionen zusammen. Motiviert werden die im Folgenden auf Basis der Literatur erfassten Problemdimensionen durch eine real-praktische Problemstellung, die es im Rahmen eines Forschungsprojektes durch die Postbank Systems AG zu analysieren und lösen galt. Die real-praktische Problemstellung wird in Kapitel 3 detailliert und dient der Konkretisierung der zentralen Problemstellung.

Die drei Problemdimensionen werden dem weiteren Verlauf dieser Arbeit einen Orientierungsrahmen geben.

---

[6] Vgl. Moormann (2004), Vgl. Passardi (1994)
[7] Vgl. Passardi (1994)
[8] Vgl. Heinrich (2001), S. 15ff; vgl. Krcmar (2005), S. 25

1.2.1 Problemdimension 1: Komplexität der Infrastrukturarchitektur

Die erste Problemdimension betrifft den Architekturentwurf der Infrastruktur: Die Wirtschaftsinformatik befasst sich in Anlehnung an KRCMAR mit der Entwicklung von Informationssystemen (im Folgenden kurz IS). Ein IS ist per Definition ein soziotechnisches Mensch-Maschine-System, das menschliche und maschinelle Komponenten umfasst, die zum Ziel der optimalen Bereitstellung von Information und Kommunikation nach wirtschaftlichen Kriterien eingesetzt werden.[9] IS sind komplexe Systeme.[10] Die bereits 1991 von SCHEER vorgestellte *Architektur integrierter Informationssysteme* (ARIS) teilt ein ebensolches IS in einzelne Komponenten auf. SCHEER betrachtet ein IS aus vier verschiedenen Blickwinkeln: der Datensicht, der Funktionssicht, der Steuerungssicht sowie der Organisationssicht. Die einzelnen Komponenten eines IS werden im Rahmen eines Vorgehensmodells entwickelt.[11] Vier Jahre früher hat ZACHMAN bereits sein *Framework for Information Systems Architecture* vorgestellt:[12] Im Gegensatz zu SCHEER, der mit ARIS die Architektur eines einzelnen IS beschreibt,[13] betrachtet ZACHMAN die Architektur einzelner IS im Zusammenhang der Gesamtarchitektur, die er als eine sog. *Enterprise Architektur* bezeichnet.

Auf Grund der zunehmenden Dynamik der Innovationen im Bereich der Informations- und Kommunikationssysteme nach 1990 ist die Komplexität der Informationssysteme sukzessive angestiegen. MOORMANN identifiziert einen derartigen Komplexitätsanstieg insbesondere bei den von der Finanzdienstleistungsinformatik betrachteten IS.[14] Nicht nur der Informationsanteil eines IS ist mit zunehmender Vernetzung komplexer geworden, sondern auch der Systemanteil eines ebensolchen IS hat einen enormen Komplexitätszuwachs erfahren. Dieser Systemanteil ist eine Komposition von Hard- und Softwarekomponenten. Letztere können Individual- und Standardsoftwareprodukte sein. Häufig besteht ein

[9] Vgl. Krcmar (2005), S. 25
[10] Vgl. Krcmar (2005), S. 26
[11] Vgl. Scheer (1993)
[12] Vgl. Zachman (1987)
[13] Vgl. Krcmar (2005), S. 45
[14] Vgl. Moormann (2004)

IS sowohl aus individuell entwickelter Software als auch aus mindestens einem Standardsoftwareprodukt. Softwareprodukte sind variantenreich. Jede neue Version hat andere Fähigkeiten und andere Systemvoraussetzungen. Aber auch Hardwareprodukte sind variantenreich. Verschiedene Hersteller liefern vergleichbare Produkte mit teilweise minimal abweichenden Fähigkeiten. In der von MOORMANN als „Dekade der vernetzten Informationsverarbeitung“[15] bezeichneten Zeit der neunziger Jahre wurden Soft- und Hardwarekomponenten in den unterschiedlichsten Varianten nicht über eine zentrale Koordinationsstelle eingeführt:[16] Die Folge sind sehr hohe Betriebskosten der einzelnen Varianten und überkomplexe Informationssysteme.

Betrachtet man ein IS analog zu SCHEER und ZACHMAN als ein System mit mehreren Schichten, so bildet die Technik die unterste Schicht. Ein Blick in die Literatur[17] zeigt jedoch, dass die Existenz dieser technischen Schicht lediglich erwähnt wird – eine konkrete Detaillierung fehlt jedoch in Gänze. In seiner 2002 vorgestellten Arbeit *A Practial Framework for Discussing IT Infrastructure* präsentiert LIU eine sehr grobe Strukturierungshilfe zur Komplexitätsbewältigung im Umfeld der IT Infrastruktur. LIU kommt zu dem Schluss, dass die Disziplin Infrastrukturarchitektur bis dato eine aus wissenschaftlicher Sicht nicht erforschte Disziplin des Software-Engineerings ist.[18]

Der Entwurf einer Infrastrukturarchitektur ist Teil des Software-Engineerings.[19] Ein Akteur, der sich mit einem derartigen Entwurf auseinandersetzt, muss im Rahmen seines Entwurfes die Komplexität der jeweils betrachteten Infrastruktur verstehen. Je komplexer das System *Infrastruktur* ist, desto komplexer sind die Entscheidungswege des Software-Engineerings innerhalb dieses Systems.[20]

Charakteristikum der technischen Schicht eines IS eines Finanzdienstleisters ist, dass die Infrastrukturkomponenten keine Eigenentwicklungen sind, sondern von unterschiedlichen Herstellern bezogen werden,

---

[15] Moormann (2004)
[16] Vgl. Penzel (2004) bzw. Dietrich/Schirra (2004), S. 4ff
[17] Beispiele sind: Liu (2002), Dern (2003), Penzel (2004), OGC (2002a), Scheer (1993), Krcmar (2005), Birkhölzer/Vaupel (2003)
[18] Vgl. Liu (2002)
[19] Vgl. Dern (2003), S. 17ff
[20] Vgl. Herzwurm/Langner (2005)

um sie nach ihrer Beschaffung zu individualisieren und zu konfigurieren. Die Fülle unterschiedlicher Standards, die teilweise von Herstellern und teilweise von internationalen Standardisierungsgremien festgelegt werden, implizieren Abhängigkeiten und Kompositionsverbote. Beispielsweise erfordert der Einsatz einer bestimmten Standardsoftware (z. B. ein Applikationsserver) ein vom seinem Hersteller vorgeschriebenes Betriebssystem (z. B. Solaris). Der Einsatz dieses Betriebssystems erfordert wiederum einen bestimmten Prozessortyp (z. B. SPARC). Kombinationsverbote können sowohl von Herstellern vorgegeben werden als auch strategischer Natur sein. Letzteres ist der Fall, wenn die Entscheidungsträger der Finanzdienstleistungsinformatik festlegen, Produkte bestimmter Hersteller nicht mehr einzusetzen. Verkompliziert wird die Strukturierung der Infrastruktur durch die hohe Dynamik der technischen Innovationen, die sich zum einen durch immer neue Varianten existierender Produkte ausdrückt und zum anderen durch die Einführung neuer Produkte hervorgerufen wird.

1.2.2 Problemdimension 2: Wissenskomplexität und Wissensvolumen

Die zweite Problemdimension betrifft das für den Entwurf einer Infrastruktur innerhalb der Finanzdienstleistungsinformatik benötigte Wissen: Das Verständnis des Variantenraums, der die verschiedenen Kombinationsmöglichkeiten eines Systems im Allgemeinen umfasst, wird durch die beiden Größen Wissenskomplexität und Wissensvolumen bestimmt. Während die Wissenskomplexität durch die Abhängigkeiten der Komponenten determiniert wird, beeinflussen die Regeln wann und wie Komponenten kombiniert werden können das Volumen des benötigten Wissens. Das Resultat einer Vernachlässigung einer wissenszentrischen Sichtweise auf den Variantenraum ist die Konzentration dieses Wissens auf wenige Mitarbeiter innerhalb eines Unternehmens.[21] Dabei

[21] Vgl. Pulkkinen et al. (1999); vgl. Bongulielmi (2002), S. 48f

sind die Beherrschung der Wissenskomplexität und des Wissensvolumens nach MÄNNISTÖ ET AL. ein seit Langem bekanntes Problem.[22]
Infrastrukturkomponenten können nach dem Ansatz von LIU in verschiedene Kategorien (z. B. Netzwerkkomponenten, Server etc.) eingeteilt werden. Jede Kategorie besitzt ihre eigene Komplexität. Jede dieser Kategorien ist eine Domäne. Als Domäne wird ein Bereich von Fachwissen bezeichnet, der für spezifische Anforderungen ausgewählt wird, der Konzepte und Begriffe beinhaltet, die durch die Anwender des Fachbereichs verstanden werden und der das Wissen einschließt, wie Softwaresysteme oder Softwaresystemteile für diesen Fachbereich zu bauen sind.[23]
Vorausgesetzt ein einzelner Mensch besitzt eine begrenzte Aufnahmekapazität bzgl. eines komplexen Systems, muss das sich aus einzelnen Domänen und deren Relationen zusammensetzende Wissen über dieses System auf mehrere Einzelpersonen aufgeteilt werden. Die Folge sind Domänenexperten, deren Wissen bzgl. einer einzelnen Domäne umfangreicher ist als das Wissen bzgl. der übrigen Domänen. Ist das zu einer Lösung erforderliche Wissen jedoch domänenübergreifend, müssen mehrere Domänenexperten an der Problemlösung beteiligt werden. Soll beispielsweise ein Datenbanksystem architektonisch entworfen werden, und setzt sich das Gesamtsystem *Datenbank* (vereinfacht) aus den einzelnen Komponenten DBMS[24], Netzwerkkomponente und Storage System zusammen, kann jede dieser Komponenten in Anlehnung an LIU einer Kategorie (und folglich einer Domäne) zugeordnet werden. Der Architekturentwurf ist somit domänenübergreifend. Je komplexer das Wissen jeder einzelnen Domäne ist, desto wahrscheinlicher ist es, dass dieses Wissen auf einzelne Personen verteilt ist. Die Folge ist, dass mehrere Personen an dem Architekturentwurf beteiligt werden müssen.

[22] Vgl. Männistö et al. (1996)
[23] Vgl. Czarnecki/Eisenecker (2000), S. 754
[24] DBMS = Datenbank Management System

1.2.3 Problemdimension 3: Historische Entwicklung der Finanzdienstleistungsinformatik

Die dritte Problemdimension betrifft die Finanzdienstleistungsinformatik selbst:[25] Die deutschen Finanzdienstleister sehen sich mit dem Problem konfrontiert, den Marktumfeldbedingten Ertragseinbrüchen mit einem effektiven Kostenmanagement zu begegnen. DIETRICH und SCHIRRA sprechen von einem ökonomischen Schock, der in den Jahren 2000 bis 2003 zur Senkung der IT-Budgets beigetragen hat.[26] FISCHER und ROTHE stellen fest, dass der Unternehmenserfolg der Finanzdienstleister maßgeblich dadurch determiniert wird, dass die erforderliche IT-Unterstützung intelligenter, schneller, kostengünstiger und in besserer Qualität im Vergleich zu Mitbewerbern zur Verfügung gestellt wird.[27] Eine Ursache für die Problemstellung, mit der sich die Finanzdienstleistungsinformatik aktuell konfrontiert sieht, ist ihre eigene historisch gewachsene Komplexität.[28] Gleichzeitig erlebt der Markt der Finanzdienstleister zurzeit einen strukturellen Wandel.[29]

Diese externen Komplexitätstreiber bestimmen die Ausgestaltung der innerbetrieblichen Abläufe und einzusetzenden Produktionsfaktoren eines jeweiligen Finanzdienstleisters. GRÖßLER ET AL.[30] stellen fest, dass in Märkten mit starken Turbulenzen und Dynamik die Faktoren Flexibilität und Zeit eine wesentliche Rolle spielen.[31]

Auf Grund der Marktinduzierten Einflüsse der Umwelt eines Finanzdienstleisters lastet auf der IT als einer der großen Kostenpositionen in den Bilanzen ein enormer Druck. Zum einen soll die IT die Komplexität der historisch gewachsenen Kernsysteme reduzieren, zum anderen soll die IT mehr Flexibilität aufweisen, um die strategischen Ziele des Geschäfts möglichst ad hoc umsetzen lassen können.[32] Die Reaktion des Systems Finanzdienstleister auf seine Umwelt impliziert eine Aktion

---

[25] Vgl. Herzwurm/Langner (2004)
[26] Vgl. Dietrich/Schirra (2004), S. 1; eine ähnliche Feststellung gibt es auch bei Bubik et al. (2000)
[27] Vgl. Fischer/Rothe (2004), S. 22
[28] Vgl. u. a. Moormann (2004), Hafner et al. (2003) und Marty (2003)
[29] Anmerkung: Nogueira/Reinhard (2000) berichten von ähnlichen Beobachtungen im Jahr 2000 auf dem südamerikanischen Kontinent.
[30] Vgl. Gößler et al. (2003)
[31] Vgl. auch Fischer/Rothe (2004), S. 22
[32] Vgl. Dietrich/Schirra (2004), S. 2ff

desselben innerhalb der Finanzdienstleistungsinformatik. Die Änderung der Umwelt des Systems Finanzdienstleistungsinformatik erfordert eine Anpassung der innerbetrieblichen Abläufe bzw. der Produktionsfaktoren dieses Systems.

MOORMANN und WÖLFING stellen fest, dass das primäre Problem der Finanzdienstleistungsinformatik struktureller Art ist:[33] Größtes Problem sind die operativen Systeme, deren Existenz und Aufbau durch die historische Entwicklung begründet werden können.

1.3 Zielsetzung

Das Ziel dieser Arbeit ist es, ein begründetes Gestaltungskonzept in Form eines Modells bereitzustellen, das die Komplexitätsbedingten Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik erfasst, um zu deren Lösung beizutragen. Folgende Forschungsfrage steht daher im Mittelpunkt:

Zentrale Forschungsfrage:

Wie muss ein Modellbasierter Lösungsansatz gestaltet und in einem Unternehmen eingeführt werden, der die Komplexitätsbedingten Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik unter der Beachtung der drei vorgestellten Problemdimensionen erfasst und entsprechend methodisch unterstützt?

Hierbei soll insbesondere untersucht werden, in welcher Weise der Infrastrukturanteil komplexer IS dekomponiert werden kann. Ein Modellbasierter Lösungsansatz soll die Dekomposition der komplexen Zusammenhänge strukturieren und auf einer Meta-Ebene beschreiben, um die dekomponierten Elemente zu konfigurierbaren Architektekturmustern komponieren zu können. Bei der Entwicklung dieses Ansatzes geht es im Sinne der wissenschaftlichen Erkenntnisgewinnung um die

[33] Vgl. Moormann/Wölfing (1999); Vgl. auch Moormann (2004), S. 11

Erweiterung des Erkenntnisstandes der Wirtschaftsinformatik in Bezug auf die Infrastrukturarchitektur der Finanzdienstleistungsinformatik. Zudem will die Arbeit dem an die Wirtschaftsinformatik als angewandte Wissenschaft gerichteten Anspruch nach der Umsetzbarkeit wissenschaftlicher Erkenntnisse gerecht werden. Die Beantwortung der zentralen Forschungsfrage und ihre in Kapitel 3 beschriebene realpraktische Problemstellung erfordern daher die Untersuchung folgender Detailfragen:

- Welches sind die die Komplexitätstreiber der Infrastrukturarchitektur der Finanzdienstleistungsinformatik?
- Inwieweit sind existierende Ansätze geeignet, um die Komplexität der Infrastrukturarchitektur in der Finanzdienstleistungsinformatik beherrschbar zu machen und die Entscheidungswege innerhalb eines Architekturentwurfes aufzuzeigen?
- Wie muss die Komplexität, die sich aus der Vielfalt Infrastrukturkomponenten und deren Zusammenhänge ergibt, in Form eines Meta-Modells beschrieben werden?
- Inwiefern kann die Komposition einer dekomponierten Infrastrukturarchitektur die Komplexität reduzieren?
- Welcher Prozess ist notwendig, um einen Modellbasierten Ansatz zur Lösung Komplexitätsbedingter Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik in einem Unternehmen anwenden zu können.
- Welche Notation eignet sich um die Wissenskomplexität und das Wissensvolumen abzubilden?
- Welche Vor- und Nachteile besitzt ein solcher Modellbasierter Ansatz?

1.4 Wissenschaftstheoretische Einordnung

1.4.1 Vorüberlegungen

An die Auseinandersetzung mit der zentralen Problemstellung dieser Arbeit knüpft in diesem Abschnitt die Diskussion über eine geeignete Forschungsmethode. Hierbei stehen folgende Anforderungen im Vordergrund:

- Die Ergebnisse der Arbeit müssen auf die Praxis übertragbar sein, so dass konkrete Handlungsempfehlungen für die Finanzdienstleistungsinformatik aus den Erkenntnissen abgeleitet werden können.
- Die Ergebnisse der Arbeit müssen in der Art verallgemeinert werden können, dass die Handlungsempfehlungen auf die gesamte Finanzdienstleistungsinformatik angewendet werden können.
- Der in dieser Arbeit vorgestellte Lösungsansatz soll einen innovativen Charakter besitzen.

Nach HEINRICH sind IS Mensch-Aufgabe-Technik Systeme, deren Elemente sich wie folgt beschreiben lassen:[34] *Menschen* wirken einzeln oder gemeinsam an der Entwicklung und Einführung von IS mit. Sie betreiben und benutzen diese Systeme, oder sie sind von ihrer Existenz wesentlich berührt. *Aufgaben* beschreiben Einzelprobleme oder Problembereiche in Wirtschaft und Verwaltung. Die *Technik* wird in Form von Informations- und Kommunikationstechniken (IKT) realisiert. Diese ermöglichen die Ein- und Ausgabe, die Speicherung, die Übermittlung sowie die Ver- und Bearbeitung von Daten. Es handelt sich hierbei im Wesentlichen um Hardware- und Softwaresysteme. Nach HEINRICH ist die Wirtschaftsinformatik sowohl eine wirtschaftswissenschaftliche als auch eine ingenieurwissenschaftliche Disziplin. Beide Forschungsorientierungen sind durch zwei unterschiedliche Forschungsrichtungen gekennzeichnet: „In der *theoretischen Forschung* wird die Entwicklung von mehr oder weniger abstrakten Theorien einschließlich Konstruktionslehren und der Umsetzung der Theorien in Konzepte und Prototypische

[34] Vgl. Heinrich (2001), S. 15ff

Produkte gearbeitet; in der *empirischen Forschung* wird an der Überarbeitung der Theorien einschließlich der Konstruktionslehren und der prototypischen Produkte gearbeitet."[35]

Die explizite Beobachtung der Realität findet in der sog. *Aktionsforschung* statt. Nach FRANK ET AL. bezeichnet die Aktionsforschung eine bestimmte Ausrichtung sozialwissenschaftlicher Forschung. Sie basiert auf der Annahme, dass die Untersuchung eines sozialen Systems unter der Berücksichtigung der Systemeigenschaften eine angemessene Vorgehensweise empfiehlt.[36] AVISON ET AL. halten fest: „[action research] is unique in the way it associates research and practice, so research informs practice and practice informs research synergistically."[37]

Ferner heist es: „In action research, the researcher wants to try out a theory with practitioners in real situations, gain feedback from this experience, modify the theory as a result of this feedback, and try it again."[38]

Die Aktionsforschung unterscheidet sich von der traditionellen Forschung insofern, dass ein Aktionsforscher bewussten Einfluss auf das Forschungsfeld nimmt, anstatt als unbeteiligter Beobachter ein Bild über die Realität zu gewinnen. Somit wird in der Aktionsforschung die traditionelle Trennung zwischen beobachtender Forschung und beobachtetem Forschungsobjekt durch die direkte Zusammenarbeit zwischen dem Forscher und den Mitarbeitern des beobachteten Unternehmens ersetzt.[39]

In Bezug auf die Eignung der Aktionsforschung im Problemumfeld der Wirtschaftsinformatik heißt es bei BASKERVILLE und WOOD-HARPER: „The discipline of IS seems to be very appropriate field for the use of action research methods."[40] FRANKE ET AL. stellen fest, dass die wesentlichen Vorteile der Aktionsforschung die Einbindung des Forschers in eine spezifische Situation sowie die Konfrontation des Forschers mit konkreten Aufgabenstellungen sind. Anhand der Aufgabenstellungen kann der

---

[35] Vgl. Heinrich (2001), S. 93
[36] Vgl. Frank et al. (1998)
[37] Avison et al. (1999), S. 94
[38] Avison et al. (1999), S. 94f
[39] Vgl. Frank et al. (1999); vgl. auch Avison et al. (1999), S. 94ff bzw. Argyris et al. (1985), S. 237
[40] Baskerville/Wood-Harper (1998), S. 90, nach Frank et al. (1998)

Forscher einen Lernprozess durchlaufen. Die traditionelle Forschung setzt hingegen voraus, dass Hypothesen über den Forschungsgegenstand vorliegen, die ex ante exakt definiert werden können, um sie später zu verwerfen, zu modifizieren oder zu bestätigen. Der Nachteil der Aktionsforschung ist jedoch, dass sie nur schwer von einem Beratungsprojekt unterschieden werden kann. FRANKE ET AL. weisen darauf hin, dass die Anwendung einer solchen Forschungsmethodik die Gefahr birgt, die gewonnenen Erkenntnisse einer partikulären Praxis vorschnell zu verallgemeinern.[41]

1.4.1 Heuristischer Bezugsrahmen

In Anbetracht der zuvor formulierten Problemdimensionen, mit denen sich diese Arbeit auseinandersetzt, erscheint die Einschlagung der empirischen Forschungsrichtung als ungeeignet. Auf Grund des geringen Kenntnisstands über die Komplexität der Infrastrukturarchitektur der Finanzdienstleistungsinformatik sind lediglich Einzelaspekte dieser Thematik bekannt. Die existierenden Abhandlungen über die erst mit Beginn dieses Jahrhunderts wahrgenommene Disziplin *Architektur* basieren auf unterschiedlichen Konzeptualisierungen.[42] Die empirische Forschungsrichtung erfordert jedoch nach BORTZ wohl definierte und entsprechend strukturierte Theorieengebilde, aus denen sich empirisch überprüfbare Hypothesensysteme ableiten lassen.[43]
Daher muss im Rahmen dieser Arbeit die Realität explizit berücksichtigt werden, um aus ihr wissenschaftliche Aussagen ableiten zu können. In Anlehnung an ALBERT muss versucht werden, die Schwächen bisheriger Problemlösungen aufzuzeigen und eine bessere, revisionsfähige Lösung zu erzielen.[44] Die Konstruktion theoretischer Aussagen auf Basis praktischer Erfordernisse stellt nach KUBICEK das Fortschrittsmedium wissenschaftlicher Untersuchungen dar.[45]

---

[41] Vgl. Frank et al. (1999)
[42] Vgl. Herzwurm/Langner (2005)
[43] Vgl. Bortz (1984), S. 2
[44] Vgl. Albert (1984); nach Herzwurm (1992), S. 13
[45] Vgl. Kubicek (1974) S. 12; nach Herzwurm (1992), S. 13

Festzustellen gilt, dass bei der Verfolgung der traditionellen Forschung eine empirisch fundierte Überprüfung des Gestaltungskonzepts im Rahmen dieser Arbeit nicht möglich ist. Der abgesehen von Einzelaspekten geringe Kenntnisstand über die Infrastrukturarchitektur von Finanzdienstleistern erfordert eine Fokussierung dieser Arbeit auf den Entdeckungszusammenhang in der Theoriebildung. Ausgehend von realen Problemen der Praxis wird ein Modell für die Praxis abgeleitet. Zentrales Problem sind demzufolge nicht die Theorien sondern die Anwendbarkeit und der Nutzen für die Praxis.[46] In Anlehnung an ULRICH unterstützt ein solches Vorgehen die anwendungsorientierte Wissenschaft, die durch den Entwurf neuer Realmodelle auf die Gewinnung praktisch nützlichen Wissens ausgerichtet ist.[47]

Auf der Suche nach wissenschaftlicher Erkenntnis wird im Rahmen dieser Arbeit auf Grund der zuvor diskutierten Rahmenbedingungen des zu untersuchenden Forschungsgebiets eine Kombination aus Aktionsforschung und Sekundärforschung angewendet. Somit verfolgt diese Arbeit nach ULRICH und HILL ein realwissenschaftliches Forschungsziel.[48] Das Intersubjektivitätsproblem[49] realwissenschaftlicher Arbeiten kann in Anlehnung an ULRICH und HILL durch die Klärung von Entdeckungs-, Begründungs- und Verwendungszusammenhang gelöst werden.[50] Im Folgenden gilt daher:

- Der *Entdeckungszusammenhang* dieser Arbeit ist in Form eines gedanklich, heuristischen Bezugrahmens durch die Erfahrungen des Autors in der betrieblichen Praxis der Wirtschaftsinformatik im Allgemeinen und der Finanzdienstleistungsinformatik im Speziellen geprägt. Weitere Einflüsse sind die Systemtheorie[51] und wissenschaftliche Diskurse im Umfeld der Finanzdienstleistungsinformatik.
- *Begründungszusammenhang*: Nach MEYER ET AL. ist forschender Fortschritt durch eine Kombination induktiver (d. h. beobachtender)

[46] Vgl. Ulrich (1984), S. 174
[47] Vgl. Ulrich (1984), S. 35
[48] Vgl. Ulrich/Hill (1976), S. 305
[49] Anmerkung: Das Ergebnis der Forschung ist unabhängig vom Forscher und kann zeitunabhängig von anderen Forschern nachvollzogen werden. Diese würden mit den gleichen Daten und den gleichen Methoden auf das gleiche Ergebnis kommen.
[50] Vgl. Ulrich/Hill (1976), S. 306
[51] Vgl. Seiffert (1992)

sowie deduktiver (d. h. logisch schließender) Schritte gekennzeichnet.[52] Daher besitzt das Vorgehen in dieser Arbeit sowohl einen deduktiven als auch einen induktiven Charakter.

> Der *Verwendungszusammenhang* resultiert aus der Zielsetzung dieser Arbeit.

1.4.2 Forschungsmethodisches Vorgehen

Anhand des zuvor abgesteckten heuristischen Bezugsrahmens kann das forschungsmethodische Vorgehen dieser Arbeit an die Strategie angewandter Forschung nach ULRICH angelehnt werden:[53] Ähnlich dem explorativen Ansatz nach KUBICEK[54] steht ein praktischer Bezug des Forschungsvorhabens im Vordergrund. Der Schwerpunkt der angewandten Forschung nach ULRICH liegt im Anwendungszusammenhang, wobei auf eine Analyse der Praxis und der anschließenden analytisch-induktiven Ableitung der Erkenntnis-zusammenhänge Regeln und Modelle folgen. Nach einer Überprüfung der vorgeschlagenen Lösung des Praxisproblems und deren Implementierung findet der Forschungsprozess sein vorläufiges Ende. Als Resultat beschreibt diese Dissertation nicht den vollzogenen Forschungsprozess, sondern das vorläufige Ergebnis dieses Prozesses: eine Ausgangsbasis theoretischer Aussagen, durch die empirisch überprüfbare Hypothesensysteme zur weiteren Erforschung der Begründungszusammenhänge ableitbar werden.

Die derzeitige Forschung liefert nur abstrakte Aussagen in Bezug auf die Handhabung der Infrastrukturarchitektur in der Finanzdienstleistungsinformatik unter Beachtung der drei Problemdimensionen. Um neue Erkenntnisse erarbeiten zu können und ein neues Konzept bzw. eine neue Sichtweise im Hinblick auf diese Thematik erarbeiten zu können, wird eine Fallstudie nach EISENHARDT und YIN durchgeführt:[55] Um die Ergebnisse dieser Fallstudie validieren zu können, schlagen EISENHARDT und YIN einen iterativen Prozess vor. Dieser Prozess erstreckt sich von der ersten Problemerkenntnis über die Formulierung der For-

[52] Vgl. Meyer et al. (1969)
[53] Vgl. Ulrich (1984), S. 192ff
[54] Vgl. Kubicek (1974)
[55] Vgl. Eisenhardt (1989), Yin (1989)

schungsfragestellung, der Entwicklung einer Lösung, der Validierung dieser Lösung anhand einer Fallstudie bis hin zur kritischen Reflexion der Lösung.

1.4.3 Aufbau der Arbeit

Der Aufbau der Arbeit resultiert aus der zentralen Forschungsfrage und orientiert sich am forschungsmethodischen Vorgehen. Auf der Suche nach wissenschaftlicher Erkenntnis werden zwei Fallstudien bei Finanzdienstleistern durchgeführt. Auf Grund beschränkter Ressourcen wird lediglich eine Fallstudie im Sinne von EISENHARDT und YIN durchgeführt:[56] Sie dient primär der Entwicklung einer Lösung sowie der Validierung und kritischen Reflexion. Die zweite Fallstudie soll vorwiegend die Intersubjektivität der mit der ersten Fallstudie gewonnenen Erkenntnisse reduzieren.

Im Anschluss an diese Einleitung wird in **Kapitel 2** grundlegendes Wissen zum Verständnis dieser Arbeit aufgebaut. Im Vordergrund stehen die Themen Komplexität und Architekturmanagement. **Kapitel 3** stellt den praktischen Bezug dieser Arbeit her, indem in die beiden Fallstudien eingeführt wird und anhand von Befragungsergebnissen die konkrete Ausprägung der drei zentralen Problemdimensionen dargestellt wird.

Nach der Vorstellung des Praxisproblems werden in **Kapitel 4** existierende Modellansätze vorgestellt und auf ihre Eignung in Bezug auf die Lösung Komplexitätsbedingter Entscheidungsprobleme in der Finanzdienstleistungsinformatik diskutiert.

Zur Entwicklung eines Modellbasierten Lösungsansatzes werden in **Kapitel 5** relevante Leittheorien präsentiert, die dem Modellentwurf im Rahmen der Aktionsforschung eine Richtung geben. Anhand dieser Leittheorien kann in **Kapitel 6** das Gestaltungskonzept des Modellbasierten Lösungsansatzes vorgestellt werden, dessen präsentierte Gestalt das Resultat der Aktionsforschung ist.

Der Lösungsansatz umfasst neben einem Meta-Modell auch ein Vorgehensmodell, das primär aus drei Phasen besteht, dem Entwurf (**Kapitel**

[56] Vgl. Eisenhardt (1989), Yin (1989)

7), der Anwendung (**Kapitel 8**) sowie der Pflege (**Kapitel 9**) des sog. *SI-Katalogs*. Die Anwendbarkeit des Modells wird anhand eines Praxisprojekts im Rahmen der Fallstudie 1 überprüft (**Kapitel 10**). Die Evaluierung des Modells erfolgt durch explorativ durchgeführte Expertenbefragungen in **Kapitel 11**.

Aufgrund des erforderlichen Zeitaufwands der Konzeptumsetzung kann eine empirisch verwertbare Ergebnisverifikation erst im Anschluss an diese Arbeit erfolgen.

Strategie der angewandten Forschung	**Kapitel in der Arbeit**
Erfassung und Typisierung praxisrelevanter Probleme	1, 3
Erfassung und Interpretation problemrelevanter Theorien und Hypothesen der empirischen Grundlagenwissenschaften	2, 4
Erfassung und Spezifizierung problemrelevanter Verfahren der Formalwissenschaften	2, 4, 5
Erfassung und Untersuchung des relevanten Anwendungszusammenhangs	3, 7
Ableitung von Beurteilungskriterien, Gestaltungsregeln und –modellen	7
Prüfung der Regeln und Modelle im Anwendungszusammenhang	8, 9
Beratung der Praxis	10, 11

Tabelle *1.1*: Strategie angewandter Forschung nach ULRICH und Aufbau der Arbeit.

1.5 Einführung in die Fallstudien

1.5.1 Fallstudie 1: Postbank Systems AG

Die in dieser Fallstudie betrachtete Postbank Systems AG (im Folgenden kurz *PBS*) ist eine 100%ige Tochter der Postbank AG (im Folgenden kurz *PB*). In der PBS werden alle IT-Aktivitäten der PB konzentriert. Die rund 1000 Mitarbeiter entstammen zum größten Teil aus der früheren DSL Bank, der PB sowie der Postbank Data GmbH. Zwischen der PBS und der PB wird ein klassisches Auftragnehmer/Auftraggeber Verhältnis gelebt. Drittmarktaktivitäten[57] existieren nicht. Die von der PBS

[57] Vgl. König (2000)

in Rechnung gestellten Beträge werden jährlich auf einen Maximalbetrag festgelegt.
Die enorme Markpräsenz der PB mit nahezu 13 Millionen Kunden stellt daher auch ebensolche Ansprüche an die Informationssysteme: Zum Zeitpunkt der Fallstudie verwalteten die Storage-Systeme rund 250 Terrabyte Daten. Alleine der Girokontenbereich erfordert täglich 15 Millionen fachlicher Buchungen, die ca. 60.000 technische Lese-/Schreiboperationen je Sekunde verursachen.

1.5.2 Fallstudie 2: BHW Holding AG

Das in der Fallstudie betrachtete Unternehmen BHW Holding AG (kurz: BHW) mit Sitz in Hameln wurde 1928 gegründet und 1997 an die Börse gebracht. Das Unternehmen ist nach eigenen Angaben im Jahr 2005 mit über 2,7 Millionen Kunden die zweitgrößte Bausparkasse Deutschlands. Im Januar 2006 wurde die BHW in den Postbank Konzern integriert.

2 Grundlagen

Ziel dieses Kapitels ist es, die grundlegenden Begriffe Komplexität und Architektur bzw. Architekturmanagement zu erläutern. Die Inhalte dieses Kapitels wurden bereits in verkürzter Form veröffentlicht.[58]

2.1 Komplexität

2.1.1 Komplexität im Allgemeinen

PILLER definiert Komplexität als „[…] das Zusammentreffen einer strukturellen Vielschichtigkeit, resultierend aus der Anzahl und Diversität der Elemente eines Systems sowie deren gegenseitige Verknüpfung und der dynamischen Veränderlichkeit der gegenseitigen Beziehungen der Systemelemente."[59] Nach PATZAK[60] kann Komplexität abstrakt durch die Kombination aus Konnektivität und Varietät beschrieben werden (vgl. Abbildung 2.3). HOFER betrachtet Komplexität als die Kombination von Varianz und Dynamik.[61] GINO definiert ein komplexes System als „[…] a network of many highly interactive and interrelated elements, each performing its own functions."[62] GALLAGHER und APPENZELLER definieren: „Ein komplexes System ist ein System, dessen Eigenschaften sich nicht vollständig aus dem Verständnis seiner einzelnen Bestandteile erklären lassen."[63]

Die Kombination einzelner (unterschiedlicher) Objekte miteinander kann zu einer Vielzahl von Möglichkeiten führen. Die Vielschichtigkeit der dynamischen Veränderlichkeit impliziert sog. *Komplexitätskosten*, die nach MEFFERT als aus der Komplexität des Leistungsprogramms, der Kundenstruktur, der Produkte sowie dem Programm und der Organisation der Leistungserstellung resultierende Faktorverbräuche bezeichnet

[58] Vgl. Herzwurm/Langner (2005)
[59] Piller (2000), S. 179
[60] Patzak (1982)
[61] Vgl. Hofer (2001), S. 14
[62] Gino (2002)
[63] Gallagher/Appenzeller (1999)

werden können.[64] Der Komplexitätsgrad einer Organisation wird durch sog. *Komplexitätstreiber* bestimmt, welche sich aus einer Vielzahl externer und interner Faktoren einer Organisation zusammensetzen.[65]

Externe vs. Interne Komplexität

Sind zwei Organisationen grundsätzlich vergleichbar, so ist diejenige Organisation die leistungsfähigere, der es am besten gelingt, eine Einfachheit mittels strategischer Konzentration sowie durch Schnittstellen- und Komplexitätsmanagement zu erzeugen.[66] Nach MALIK ist die Bewältigung einer existierenden Komplexität durch eine gezielte Entkopplung der externen Varietät von interner Komplexität eine Managementaufgabe.[67]

PILLER bezeichnet die *externe Komplexität* in diesem Zusammenhang als „Komplexität der unternehmerischen Umwelt“[68]. Sie beschreibt die Anzahl und die Verschiedenartigkeiten der relevanten Umweltfaktoren, die bei der Steuerung einer Organisation berücksichtigt werden müssen. Die Komplexitätstreiber dieser externen Komplexität sind beispielsweise die Globalisierung und der hieraus resultierende Wettbewerb, die stetig individueller werdenden Nachfragerwünsche oder die technologischen Entwicklungen.[69]

Das Eingehen auf individuelle Nachfragerwünsche kann nach BÖHMANN und KRCMAR in einer sog. *Komplexitätsfalle* enden, die durch einen Differenzierungs- und Individualisierungsdruck ausgelöst wird. Die sukzessive Erfüllung individueller Nachfragerwünsche erfordert individuelle Lösungen, die immer stärker differenzierte Leistungen voraussetzen. Die Folge dieser Spirale ist, dass die abgesetzten Volumina je Variante sinken.

Die Leistungserstellung erfordert innerbetriebliche Abläufe und einzusetzende Produktionsfaktoren. Die Vielschichtigkeit und Veränderlichkeit dieser Faktoren und Abläufe bestimmt den Grad der *internen Komplexität*.[70] Nach FIRSCHAU ET AL. beschreibt die interne Komplexität „[…] die im Rahmen der Auftragsabwicklung auftretende Vielfalt an

---

[64] Vgl. Meffert (2000), S. 1043
[65] Vgl. Piller/Waringer (1999), S. 5
[66] Vgl. Hofer (2001), S. 17
[67] Vgl. Malik (1996), S. 184 (nach Hofer (2001), S. 17)
[68] Piller/Waringer (1999), S. 5
[69] Vgl. Piller/Waringer (1999), S. 5f
[70] Vgl. Piller/Waringer (1999), S. 6

Bauteilen und –gruppen, Produkten und Prozessen."[71] Diese Vielfalt spiegelt sich nach HOFER in der Varianz verschiedenartiger Produkte zur selben Zeit wieder, deren Dynamik die Intensität des Wechsels zwischen zwei Zeitpunkten der Varianz beschreibt.[72]

Gesetz von Ashby

Wird gemäß der Definition von GALLAGHER und APPENZELLER ein komplexes System sich selbst überlassen, können neue und unerwartete Entwicklungen zutage treten. Das *Gesetz von Ashby*[73] besagt, dass ein System nur mit einem System unter Kontrolle gebracht werden kann, dessen Komplexität mindestens so groß ist, wie die des zu steuernden Systems. Je komplexer die externen Einflüsse auf ein System sind, desto komplexer müssen auch die Maßnahmen sein, welche die Kontrolle gewährleisten sollen. Übertragen auf Varietät bedeutet dies, dass nur Varietät Varietät lenken kann. Weiter besagt dieses Gesetzt, dass zur Regulierung und Kontrolle eines Systems, welches den Einflüssen der Umwelt unterliegt, jede von außen auf das System wirkende Störung durch den Regulator pariert werden muss. Nimmt die Vielfalt der Störungen zu, so muss auch der Regulator mindestens die gleiche Vielfalt aufweisen, um die Kontrolle zu wahren. Ist dies nicht der Fall, kann das System außer Kontrolle geraten.

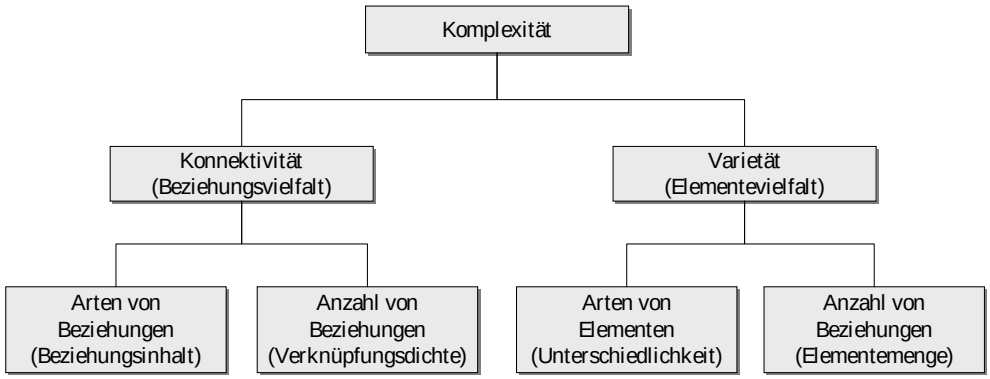


Abbildung 2.1: Komplexität – Merkmale der Systemstruktur (Quelle: Firschau et al. (2002), S. 9)

In Bezug auf die externe und interne Komplexität besagt das Gesetz von Ashby weiter, dass sich die interne Komplexität im Idealfall der externen Komplexität derart anpasst, dass beide Komplexitätsgrade gleich

[71] Firschau et al. (2002), S. 13
[72] Vgl. Hofer (2001), S. 14
[73] Ashby (1956)

sind. Nach LEWIS und STEWART lässt sich Komplexität nicht messen lässt, sondern nur relativ vergleichen.[74]

Im Rahmen einer Untersuchung von Produktionsbetrieben im Jahr 2003 haben GRÖßLER ET AL. festgestellt, dass die externe (= Marktinduzierte) Komplexität keinen direkten Einfluss auf die interne Struktur der Produktionsbetriebe hat. Die Marktinduzierte Komplexität hat jedoch einen signifikanten Einfluss auf die strategische Ausrichtung des Produktionsbetriebs. GRÖßLER ET AL. haben gezeigt, dass in Märkten mit starken Turbulenzen und Dynamik die Faktoren Flexibilität und Zeit eine wesentliche Rolle spielen. Im Fall von stabilen, unveränderlichen Märkten sind diese Faktoren jedoch nur eine untergeordnete Relevanz.[75]

2.1.2 Komplexität im Speziellen

Die externen Komplexitätstreiber bestimmen die Ausgestaltung der innerbetrieblichen Abläufe und einzusetzenden Produktionsfaktoren eines jeweiligen Finanzdienstleisters. Die Feststellung von GRÖßLER ET AL.,[76] dass in Märkten mit starken Turbulenzen und Dynamik die Faktoren Flexibilität und Zeit eine wesentliche Rolle spielen, hat auch einen Einfluss auf die Verhaltensweise eines Finanzdienstleisters am Markt.[77] Wie in Abbildung 2.2 dargestellt, bestimmt ein Regulator die Abbildung der Umwelt auf das mentale Modell (das System Finanzdienstleister). Nach dem Gesetz von Ashby muss die Komplexität dieses Regulators mindestens die Komplexität der Umwelt aufweisen.

Die Komplexität, die auf das an das System Finanzdienstleister gekoppelte System Finanzdienstleistungsinformatik einwirkt, ist abhängig von der Komplexität des Regulators zwischen dem Marktumfeld der Finanzdienstleistung und dem Finanzdienstleister selber. Diese interne Komplexität des Systems Finanzdienstleister bestimmt die Ausgestaltung des Regulators zwischen dem System Finanzdienstleister und dem System Finanzdienstleistungsinformatik. Zur Ausgestaltung des Regulators zwischen der Finanzdienstleistung und der Finanzdienstleis-

[74] Lewis/Steward (2003)
[75] Vgl. Gößler et al. (2003)
[76] Vgl. Gößler et al. (2003)
[77] Vgl. auch Fischer/Rothe (2004), S. 22

tungsinformatik muss daher die externe und interne Komplexität des vorgeschalteten Systems Finanzdienstleister untersucht werden.

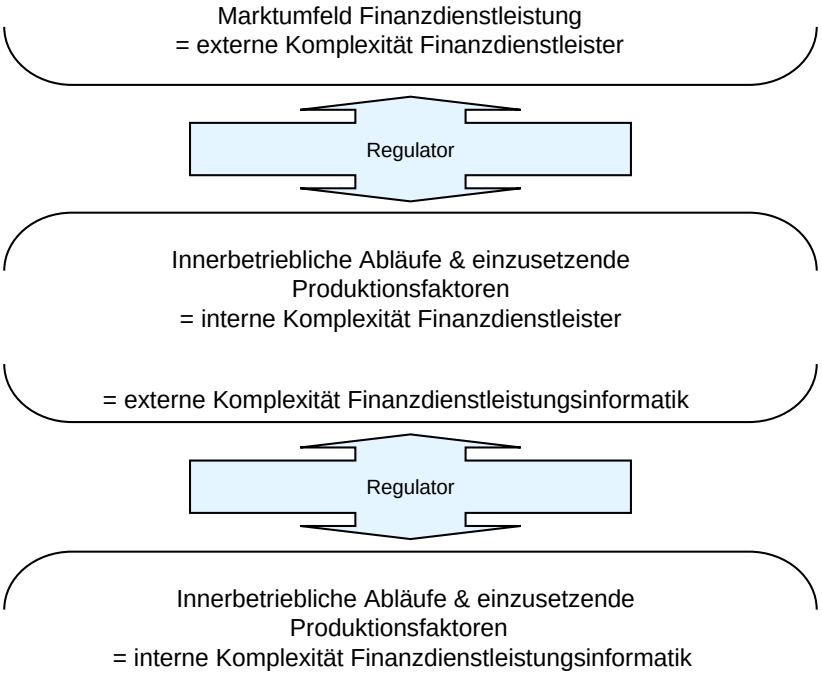


Abbildung 2.2: Ein Regulator zwischen externer Sicht und interner Sicht. (Quelle: Eigene Darstellung)

2.1.2.1 Externe und interne Komplexität der Finanzdienstleister

Finanzdienstleister sind in besonderem Maße abhängig von der Informationstechnologie (IT). Die IT bildet nach MOORMANN und FISCHER das Rückgrat einer jeden Bank:[78] Banken handeln mit Informationen, die mittels Technologien verarbeitet werden.

IT-Anteil am Fertigungsprozess

Dass die Informationstechnologie bis zur heutigen Zeit einen sehr hohen Durchdringungsgrad der fianzdienstleistungstechnischen Fertigungsprozesse erreicht hat, ist mit bloßem Auge erkennbar. So ist es beispielsweise möglich, dass ein Bankkunde eine Aktienorder über das Internet aufgibt, welche direkt oder indirekt vom Online-Bankingsystem seines Kreditinstituts automatisiert an die Handelsplattform XETRA weitergeleitet wird. Das XETRA-System wiederum ist das vollelektronische Handelssystem der Deutschen Börse AG, das den ehemaligen Parkett-

[78] Vgl. Moormann/Fischer (2004), S. V

handel in Punkto Flexibilität, Geschwindigkeit und Kosten bei weitem überbietet.[79]

Der generelle Fertigungsprozess einer Bank untergliedert sich in vier Subprozesse: Akquisition, Vereinbarung von Geschäften, Abwicklung von Geschäften und Bereitstellung von Informationen (siehe Abbildung 2.3):[80]

- Im Rahmen der *Akquisition* wird eine Nachfrage über verschiedenste Vertriebskanäle erzeugt. Dieser Subprozess umfasst alle Aktivitäten, die mit der Vermarktung von bankspezifischen Produkten zusammenhängen.
- Die *Abwicklung von Geschäften* enthält die Betreuung des Kunden, die Unterbreitung von Angeboten sowie den Abschluss von Verträgen im sog. *Front Office* Bereich. Diese Aktivitäten können zum Beispiel in Zusammenarbeit mit einem Kundenberater oder eigenständig vom Kunden via Online-Legitimation durchgeführt werden.
- Die *Vereinbarung von Geschäften* findet im sog. *Back Office* Bereich eines Kreditinstituts statt. Die Kundenauftragsdaten werden elektronisch erfasst, verarbeitet, gebucht und gespeichert.
- Im Rahmen des Subprozesses *Bereitstellung von Informationen* werden zum einen dem Kunden Informationen (z. B. Konto- und Depotauszüge) zur Verfügung gestellt und zum anderen Bankinterne Prozesse (z. B. Meldewesen und Controlling) mit Daten versorgt.

Allen diesen vier Subprozessen ist gemein, dass sie bei der Verarbeitung von Informationen sehr stark durch Technik unterstützt werden. MOORMANN bezeichnet die IT einer Bank daher als „[…] Herzstück der bankbetrieblichen Fertigung.“[81] und kann daher „[…] mit der Produktionsstraße eines Industrieunternehmens […] verglichen werden.“[82]

---

[79] Anmerkung: In Hartenstein (2003), S. 29 wird als Beispiel das Projekt „Calvin“ des Online-Brokers Cortal Consors AG genannt, bei dem eine neue Applikations-Middleware eingesetzt wird, die täglich rund 10 Millionen Transaktionen von 500.000 Kunden verarbeiten kann.
[80] Vgl. Moormann (2004), S. 4; Vgl. SAP (2002) S. 6ff
[81] Moormann (2004), S. 5
[82] Moormann (2004), S. 5

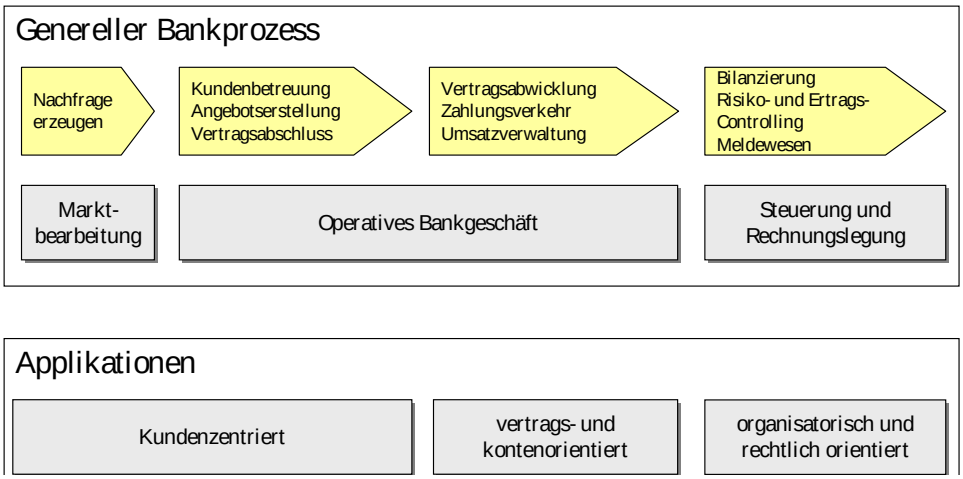


Abbildung 2.3: Bankprozess und Applikationen. (Quelle: SAP (2002), S. 7)

Informations-intensität

PORTER und MILLAR liefern allgemeingültige Kriterien, mit deren Hilfe die Informationsintensität des Wertschöpfungsprozesses und des vom Unternehmen erstellten Produktes bestimmt werden kann. Die Informationsintensität eines Produktes wird durch folgende Merkmale charakterisiert:[83]

- Das Produkt liefert in erster Linie Informationen.
- Das Betreiben des Produkts erfordert in hohem Maße die Verarbeitung von Informationen.
- Die Nutzung des Produkts erfordert die Nutzung umfangreicher Informationen durch den Anwender.
- Das Geschäft mit dem Kunden weist eine hohe Informationsintensität auf.

Ebenso prägen folgende Kriterien den Grad der Informationsintensität bzgl. des Wertschöpfungsprozesses eines Unternehmens: [84]

- Das Unternehmen steht in direktem Bezug zu einer großen Zahl von Kunden und Lieferanten.
- Der Verkauf des Produktes erfordert eine Vielzahl hilfreicher Informationen.
- Es werden viele unterschiedliche Produkte verkauft.
- Ein Produkt setzt sich aus vielen Einzelkomponenten zusammen.

[83] Vgl. Porter/Millar (1985), S. 158ff; vgl. Rothe (2004), S. 37ff
[84] Vgl. Porter/Millar (1985), S. 158ff; vgl. Rothe (2004), S. 37ff

> Zwischen Auftragseingang und Produktauslieferung besteht eine lange Taktzeit.

Wie die Skizzierung des Fertigungsprozesses einer Bank in Abbildung 2.3 zeigt, weisen im Umfeld von Kreditinstituten sowohl die Produkte (z. B. Girokonten und Depots) als auch die Wertschöpfungsprozesse (z. B. zählt die Postbank im Jahr 2005 mehr als 12 Millionen Kunden bei einer Bilanzsumme von 129 Milliarden Euro und 10.000 Mitarbeitern und verarbeitet im Jahr über 2,7 Milliarden Zahlungsverkehrstransaktionen)[85] eine sehr hohe Informationsintensität auf. Zu demselben Schluss kommt auch PETROVIC.[86]

KLOSTERMEIER und SEEGER machen die Informationsintensität von Finanzdienstleistern an einem praktischen Bild deutlich:[87] Im Jahr 2002 betrug der Umlauf von Münzen und Banknoten im Euro-Raum nur noch 4,5% der Geldmenge (Bargeld, Sicht-, Termin- und Spareinlagen, Pensionsgeschäfte bzw. festverzinsliche Wertpapiere). Die restlichen 95,5% müssen also elektronisch (also informationstechnisch) verarbeitet worden sein.

KRÖNUNG weist darauf hin, dass sich Banken stetig stärker als Dienstleister für finanzmarktrelevante Informationen profilieren. Weil die hierbei angebotenen Produkte aus Sicht der reinen Leistungserstellung standardisiert sind, ist die Differenzierung im Wettbewerb über das Produkt an sich nur schwer möglich. Eine Differenzierung kann laut KRÖNUNG folglich nur über die Informationsqualität erfolgen. Auf Basis dieser Feststellung stellt der „bankinterne IT-Einsatz […] einen wesentlichen Hebel zur positiven Differenzierung im Wettbewerb dar und beeinflusst entscheidend die Position der Bank […]“[88].

Auch BUBIK ET AL. heben die bedeutende Rolle der Informationstechnik in der Finanzdienstleistungsbranche hervor:[89] Durch den massiven Einsatz von IT wurden die Geschäftsbedingungen für Finanzdienstleister modifiziert. Wie in Abbildung 2.4 dargestellt, hat die Informationstechnologie auf drei Ebenen eine neue Qualität gewonnen: Zum einen werden

[85] Vgl. Berensmann (2004), S. 61
[86] Vgl. Petrovic (1994), S. 583ff
[87] Vgl. Klostermeier/Seeger (2002), S. 21; siehe auch Rothe (2004), S. 38
[88] Vgl. Krönung (1998), S. 40
[89] Vgl. Bubik et al. (2000), S 103ff

gegebene Geschäftsprozesse umgestaltet, zum anderen Wertschöpfungsketten neu bestimmt. Durch die technologischen Innovationen werden Finanzdienstleister in die Lage versetzt, neue Produkte und Märkte zu schaffen, die ohne den Einsatz der IT nicht denkbar wären. Die Veränderung des Finanzdienstleistungsgeschäfts wird hierbei auf drei Ebenen aufgeteilt. Auf der ersten Ebene findet ein IT-induzierter Wandel statt, der nicht durch die Geschäftsplaner innerhalb einer Bank, sondern durch die IT vorangetrieben wird.

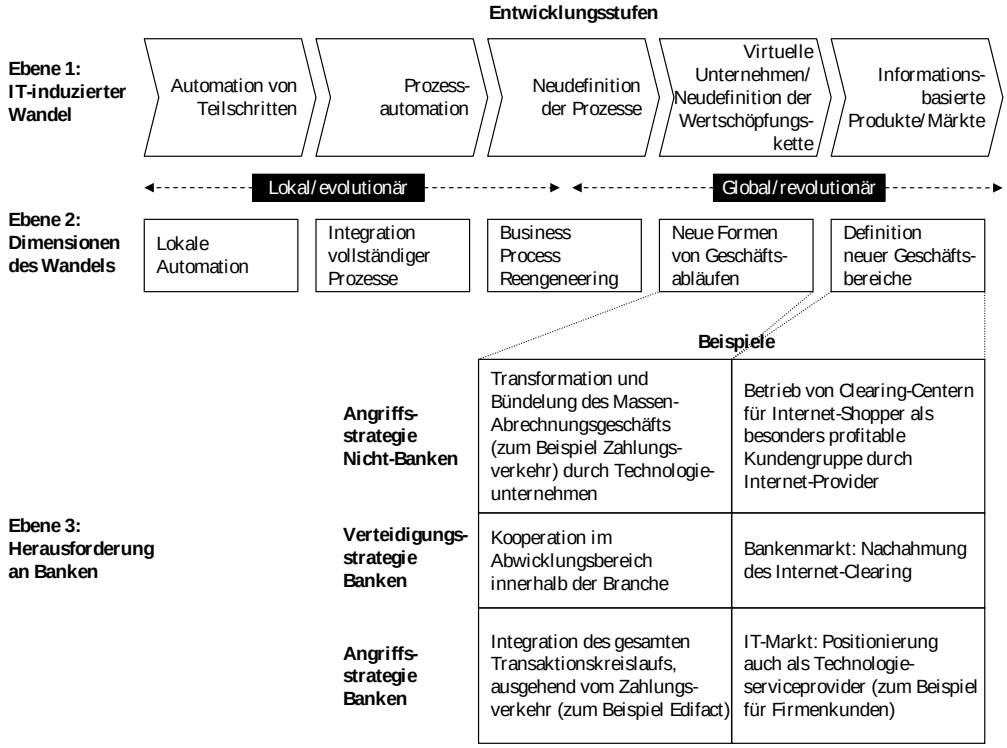


Abbildung 2.4: IT verändert das Finanzdienstleistungsgeschäft – Das Beispiel Banken. (In Anlehnung an Bubik et al. (2000), S. 103)

Auf der zweiten und dritten Ebene verstärkt sich die Bedeutung der Informationstechnologie für Finanzdienstleister. Durch eine Fokussierung der Banken auf das sog. *Kerngeschäft* entstehen neue Kooperationen und Dienstleistungen, die die Informationsintensität enorm erhöhen. So haben beispielsweise die Deutsche Bank und die Dresdner Bank entschieden, ihren Zahlungsverkehr von der Postbank abwickeln zu lassen.[90] Für die Kunden des jeweiligen Instituts ändert sich hierdurch nichts. Sie führen ihre Konten weiterhin bei dem jeweiligen Institut. Die

[90] Dies geht aus einer gemeinsamen Pressemitteilung der drei Kreditinstitute hervor, die am 01.10.2003 veröffentlicht wurde.

Postbank agiert in diesem Bereich also als ein Dienstleister („Transaktionsbank“) für andere Marktteilnehmer.

Aufwendungen für IT

Eine andere Form der Betrachtungsweise, welchen Stellenwert der IT-Bedarf im Umfeld von Finanzdienstleistern innehat, ist die Betrachtung konkreter Kostenpositionen in den Gewinn- und Verlustrechnungen einzelner Institute.[91] So macht beispielsweise die Dresdner Bank in ihrem Geschäftsbericht[92] von 2004 Gesamtverwaltungskosten von 5.043 Millionen Euro geltend. Diese Kosten teilen sich auf in 3.314 Millionen Euro für Personalaufwendungen und 2.089 Millionen Euro für Sachaufwendungen in denen die Kosten für den Bankbetrieb (die IT) enthalten sind. Analog verhält es sich im Fall des Geschäftsberichts[93] für 2003 der HypoVereinsbank: Im Rahmen von 7.735 Millionen Euro Gesamtverwaltungskosten entfallen allein 1.170 Millionen Euro auf reine IT-Kosten. Dies entspricht etwa 15% der Gesamtverwaltungskosten. Werden die IT-Kosten der Dresdner Bank des Jahres 2004 ebenfalls auf 15% geschätzt,[94] ergibt sich eine Kostenposition von ca. 817 Millionen Euro. Ähnlich verhält es sich auch bei anderen Kreditinstituten:[95] MOORMANN schätzt die IT-Ausgaben im Jahr 2002 der Deutschen Bank auf fast drei Milliarden Euro, die der Commerzbank auf rund 800 Millionen Euro.[96] Die IT-Aufwendungen der Postbank belaufen sich im Jahr 2005 auf mindestens 400 Millionen Euro.[97] NOGUEIRA und REINHARD nennen bei brasilianischen Banken einen Anteil der IT-Kosten an den Gesamtaufwendungen von 6,4%.[98]

KLOSTERMEIER und SEEGER schätzen, dass der Anteil der IT-Ausgaben von Finanzdienstleistern gemessen am Geschäftsvolumen im Durch-

---

[91] Vgl. Rothe (2004)
[92] Siehe Dresdner Bank (2004), S. 36ff
[93] Siehe HypoVereinsbank (2003), S. 30ff
[94] Anmerkung: In den Gewinn- und Verlustrechungen der einzelnen Bankinstitute werden keine expliziten Ausgaben für IT-Kosten aufgeführt. Diese werden in der Regel unter dem Oberbegriff „Verwaltungsaufwendungen“ zusammengefasst. Die Schätzung von 15% entspricht nach Moormann (2004), S. 9ff, in etwa dem durchschnittlichen Verwaltungsaufwand deutscher Filialbanken. Auch in Bubik et al. (2000), S. 102, werden 15-20 % IT-Kostenanteil an den Verwaltungsaufwendungen genannt.
[95] Siehe auch Fischer/Rothe (2004), S. 22
[96] Vgl. Moormann (2004), S. 9ff
[97] Anmerkung: Die realen Ausgaben für IT der Postbank werden nicht veröffentlicht. Die Umsätze der Postbank Systems AG, deren einziger Kunde die Postbank ist, hat laut CW (2005) eine Größenordnung um 384 Millionen Euro. Hinzugerechnet werden müssen alle anfallenden Kosten derjenigen IT-Systeme, die nicht von der Postbank Systems verwaltet werden.
[98] Vgl. Nogueira/Reinhard (2000), S. 1

schnitt viel höher ist als beispielsweise im produzierenden Gewerbe. Wesentliche Ursache dieses Umstands ist, dass „[d]as Vermögen der Banken […] sich nicht mehr in Tresoren, sondern auf ihren Rechnern [befindet].“[99] Als Folge stellen die Autoren fest, dass Finanzdienstleister direkt von der IT abhängen. An IT-intensiven Bereichen der Finanzdienstleister wie zum Beispiel dem Transaktionsbereich haben die IT-Kosten schätzungsweise einen Anteil von 30 bis 40 % der Fixkosten.[100]

Charakterisierung des Marktumfelds der Finanzdienstleister

Der Markt der Finanzdienstleister erlebt während der Niederschrift dieser Arbeit einen strukturellen Wandel. Insbesondere im europäischen Raum sind vermehrt grenzüberschreitende Bankenfusionen erkennbar.[101] Nach Großbankenfusionen in den USA im Jahr 2003 (Bank of America und Fleet Boston sowie JP Morgan Chase und Bank One)[102] hat die Fusionswelle auch europäische Banken erreicht: Die englische Abbey National hat 2004 die spanische Großbank Banco Santander Central Hispano übernommen. Die niederländische ABM Amro hat die italienische Banca Antonveneta akquiriert. Im Jahr 2005 haben die italienische UniCredito Italiano und die deutsche HypoVereinsbank fusioniert. Übernahmen finden aber nicht nur unter Banken statt. Im Jahr 2002 hat die klassisch als Versicherer aufgestellte Allianz Gruppe die Dresdner Bank gekauft. Im Jahr 2003 hat der niederländische Allfinanzkonzern ING Groep die deutsche Diba übernommen und mit der der Nürnberger Direktbank Entrium verschmolzen, die sie zuvor für rund 300 Millionen Euro von der italienischen Bank Capitalia gekauft hatte. Eine andere Form der Konsolidierung und Integration wird beispielsweise von der Commerzbank verfolgt. Diese verfolgt eine grenzüberschreitende Strategie, bei der ein Netzwerk sog. „befreundeter Partnerbanken“ aufgebaut wird: Der italienische Versicherungsanbieter Generali, die italienische Banca Intesa und die spanische Großbank Banco Santander Central Hispano besitzen zusammen rund 11% der Commerz-

[99] Klostermeier/Seeger (2002), S. 21
[100] Vgl. Fuchs (2005), S. 12
[101] Anmerkung: Nogueira/Reinhard (2000) berichten von ähnlichen Beobachtungen im Jahr 2000 auf dem südamerikanischen Kontinent.
[102] Vgl. Franke (2003)

bank-Aktien. Gleichzeitig ist die Commerzbank prozentual an den anderen 3 Instituten beteiligt.[103]

Eine der Ursachen für derartige Fusionen und Übernahmen nicht nur unter Banken sondern auch zwischen Banken und Versicherern ist nach FUCHS die Eroberung neuer Märkte und die eigene Absicherung gegenüber dem weltweit angestiegenen Konkurrenzdruck, der auch als „Globaler Wettbewerb“ bezeichnet wird. Die Übernahmen und Kooperationen zwischen Banken und Versicherern erklärt sich vor allem mit dem Aufbau einer zusätzlichen Einnahmequelle: So vertreibt beispielsweise die Dresdner Bank in ihrem ausgeprägten Filialnetz auch die Versicherungsprodukte der Allianz. Die Commerzbank bietet ihren Kunden die gesamte Angebotspalette der Generali Gruppe (und umgekehrt) an.[104]

Unter den deutschen Finanzdienstleistern hat sich der Wettbewerb enorm verstärkt.[105] Im Streben um ein stetiges Wachstum haben die einzelnen Bankinstitute den deutschen Markt wieder entdeckt, welcher weltweit der viertgrößte Markt für Konsumkredite ist.[106]

Als Unternehmen muss ein Finanzdienstleister ökonomisch arbeiten. Als börsennotiertes Unternehmen steht er zusätzlich in der Pflicht, den größtmöglichen Gewinn für die Anteilseigner zu erzielen. Um den Gewinn steigern zu können, muss ein Unternehmen die Spanne zwischen Einnahmen und Ausgaben vergrößern. Bleiben die Einnahmen konstant oder sinken sie, müssen die Ausgaben gesenkt werden, um den Gewinn zu maximieren.

Insbesondere in den Jahren 2000 bis 2005 ist eine Stagnation des Marktes erkennbar.[107] Nach MOORMANN befindet sich der Bankensektor in einem tief greifenden Veränderungsprozess, dessen Ursachen im Wesentlichen durch folgende drei Felder gekennzeichnet sind:[108]

Problemfelder

> Nach dem Platzen der Börsen-Blase im Jahr 2000 stecken die Banken in einer schweren Ertragskrise. Die Märkte sind weitge-

[103] Vgl. Fuchs (2003), S. 8ff
[104] Vgl. Fuchs (2003), S. 11ff; vgl. Weimer/Wißkirchen (1999)
[105] Vgl. Neumann (2004), S. 97
[106] Vgl. eBanker (2003)
[107] Vgl. Flaig et al. (2005)
[108] Vgl. Moormann (2004), S. 3

hend verteilt und zusätzliche Geschäfte in Nischenmärkten sind – sofern überhaupt möglich – schwierig zu erobern. Um Kunden zu gewinnen haben Banken einen Preiskampf eröffnet, bei dem beispielsweise kostenlose Kontoführungen oder Depotverwaltungen feilgeboten werden.

> Auswirkung dieser Entwicklung ist ein hoher Kostendruck, der insbesondere deutsche Bankinstitute hart trifft, da sie ihre Leistungen im internationalen Vergleich strukturell zu teuer anbieten. KLOSTERMEIER und SEEGER verweisen diesbezüglich auf eine Studie nach der jedes zweite Kreditinstitut mit zu hohen Kosten und jedes vierte Kreditinstitut mit Ertragseinbrüchen zu kämpfen hat.[109] Hiernach entstehen deutschen Großbanken im Durchschnitt für die Erzielung von 100 Euro Einnahmen 80 Euro Kosten – bei einem EU-Schnitt von 60 Euro.[110] HACKETHAL und SCHMIDT provozieren mit der Aussage, dass der Grund hierfür in der Unfähigkeit der deutschen Banken liegt, maximale Gewinne aus ihrem Kerngeschäft zu ziehen.[111] FISCHER und ROTHE sprechen in diesem Zusammenhang von einem „drastischen Rentabilitätsverfall und [einem] bedrohlichen Anstieg der Risikokosten“[112]. Als Gegenmaßnahme wird in vielen Instituten bereits ein stringentes Kostenmanagement praktiziert. Da die IT-Kosten eine wesentliche Kostenposition in den Gewinn- und Verlustrechnungen deutscher Finanzdienstleister einnehmen, ist deren Senkung ein Primärziel der Verantwortlichen.

> Infolge technologisch und geschäftspolitisch begründeter kurzfristiger Richtungswechsel des Managements sind viele unklare Geschäftsmodelle entstanden, die konzeptionellen Handlungsbedarf aufzeigen.[113]

---

[109] Vgl. Klostermeier/Seeger (2002), S. 22

[110] Anmerkung: Die in Kostermeier/Seeger (2002) genannten Studien berücksichtigen jedoch nicht die unterschiedlichen Gesetzeslagen der betrachteten Regionen. Alleine die Archivierung von Belegen über den gesetzlich vorgeschriebenen Zeitraum von 10 Jahren verursacht enorme Kosten, die mit zunehmender Transaktionszahl proportional steigen.

[111] Vgl. Hackethal/Schmidt (2005)

[112] Fischer/Rothe (2004), S. 23

[113] Anmerkung: Im Rahmen der Fallstudie äußerte sich ein IT-Manager diesbezüglich wie folgt: „Wenn die Bank wüsste, was sie will, dann wäre es für uns viel einfacher.“

Die Finanzdienstleister haben die relevanten Problemfelder weitestgehend erkannt und Gegenmaßnahmen eingeleitet. Durch diese Maßnahmen findet nach MOORMANN ein Umbau statt, der sich in folgenden Entwicklungen ausdrückt:[114]

- Die klassischen hochintegrierten Geschäftsprozesse werden nach MARIGHETTI seziert und auf die Prozesse Produktentwicklung, Distribution und Abwicklung aufgeteilt.[115] MOORMANN erkennt in dieser Transformation einen „Trend zur Industrialisierung des Bankensektors“[116].
- Finanzdienstleister konzentrieren sich immer mehr auf ihre eigentlichen Kernkompetenzen. Dies führt vermehrt zu Outsourcing-Projekten und Fusionen bei gleichzeitiger Suche nach der hocheffizienten Abwicklung von Bankleistungen.
- Dadurch dass sich beispielsweise Banken vermehrt auf ihr Kerngeschäft konzentrieren, stoßen Drittanbieter wie AWD oder MLP in den Finanzdienstleistungssektor und kooperieren mit den alteingesessenen Instituten.
- Die Kunden haben im Laufe mehr Erfahrung mit unterschiedlichen Bankinstituten gemacht, wodurch ihre Ansprüche enorm gestiegen sind. Die Folge ist eine verstärkte Anstrengung im Rahmen des CRM[117].
- Durch die Konzentration auf Kernkompetenzen werden die Kernprozesse kontinuierlich optimiert („Business Reengineering“).
- Die Fähigkeiten eines Mitarbeiters müssen aufgrund der immer weiter steigenden Verzahnung von Geschäft und IT nicht mehr nur fachlich sondern immer mehr auch technisch ausgerichtet sein.

[114] Vgl. Moormann (2004), S. 3ff
[115] Vgl. Marighetti (2001)
[116] Moormann (2004), S. 3ff
[117] Customer Relationship Management

2.1.2.2 Externe und interne Komplexität der Finanzdienstleistungsinformatik

Kosten- bzw. Budgetreduktion

Wie bereits gezeigt, sehen sich Finanzdienstleister in Deutschland mit dem Problem konfrontiert, den marktumfeldbedingten Ertragseinbrüchen mit einem effektiven Kostenmanagement zu begegnen.

DIETRICH und SCHIRRA sprechen diesbezüglich von einem ökonomischen Schock, der in den Jahren 2000 bis 2003 zur Senkung der IT-Budgets beigetragen hat:[118] Zum einen haben in ihren Augen die IT-Abteilungen mittels unverständlicher Fachbegriffe und immer neuen Abkürzungen für scheinbar immer innovativere Technologien den Vorständen stetig wachsende Gelder abgerungen.[119] Zum anderen haben die steigenden Budgets nicht dazu geführt, dass die Anwender der IT-Systeme zufriedener wurden. Der erstmals in der japanischen Automobilindustrie in den 80ger Jahren praktizierte Paradigmenwechsel vom „mehr kostet mehr“ zum „bessere Leistung zu niedrigeren Kosten“ hat vor den IT-Abteilungen halt gemacht.

Eine Analyse der dieser Arbeit zugrunde liegenden Literatur macht besonders eins deutlich: ein wesentlicher Anteil aller Autoren sieht die Senkung der IT-Kosten als oberstes Ziel der Managementaktivitäten an. FISCHER und ROHTE begründen diese Fokussierung damit, „dass die IT-Kosten gegenüber den Personalkosten eine geringere Remanenz haben und Sparmaßnahmen […] entsprechend schnellere Wirkung in der Gewinn- und Verlustrechnung […] zeigen.“[120]

Der Markt der Finanzdienstleister ist von Konsolidierungen und Fusionen geprägt. Nach FUCHS sind die „europäischen Bankenfusionen […] wesentlich durch die Suche nach Möglichkeiten zur Kostensenkung und nach neuen Ertragschancen motiviert.“[121] Eine Studie der Investmentbank Morgan Stanley und der Beratungsfirma Mercer Oliver Wyman identifiziert vor allem Einsparungen bei den IT-Plattformen der Finanzdienstleister als Treiber internationaler Bankenfusionen:[122] Banken könnten ihre Erträge alleine durch die Reduktion der IT-Kosten pau-

[118] Vgl. Dietrich/Schirra (2004), S. 1
[119] Eine ähnliche Feststellung gibt es auch bei Bubik et al. (2000)
[120] Fischer/Rothe (2004), S. 22
[121] Fuchs (2005), S. 12
[122] Vgl. Handelsblatt (2005) sowie Fuchs (2005), S. 12

schal um 20% steigern. Nach FUCHS steht diese Aussage im Widerspruch zu der klassischen Auffassung, dass internationale Fusionen nur geringe Kosteneffekte haben.

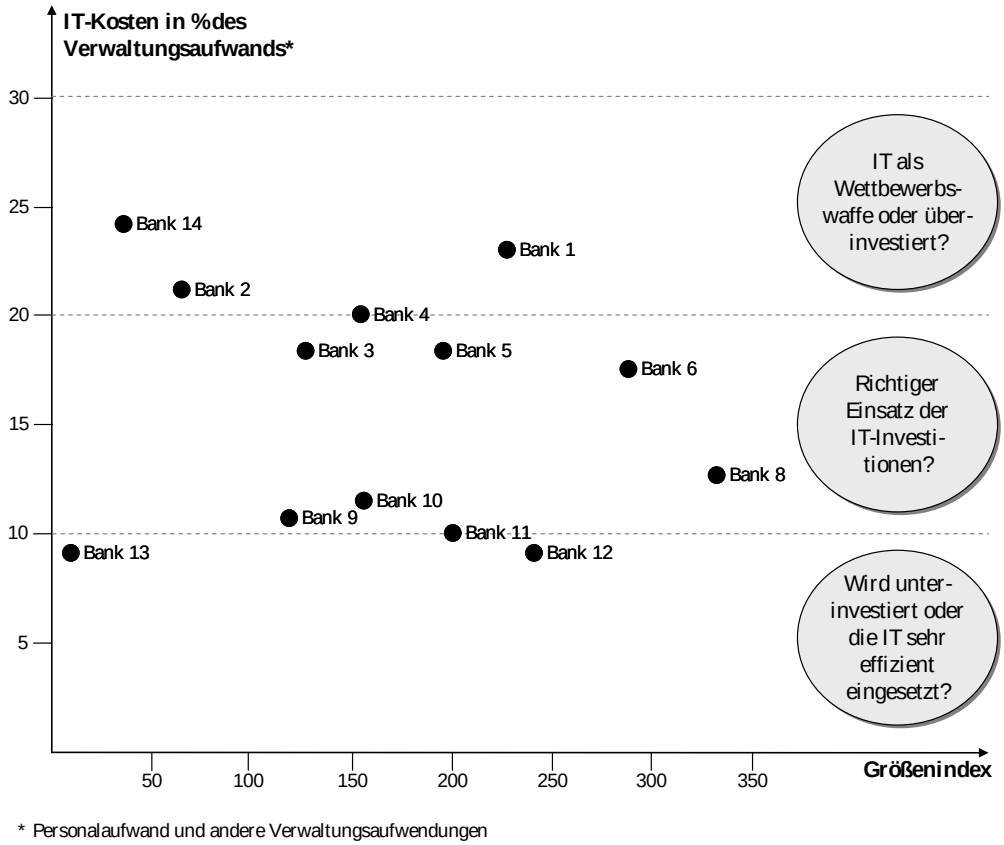


Abbildung 2.5: Die prozentualen Anteile der IT-Kosten am Verwaltungsaufwand im Verhältnis zur Größe der europäischen Großbank. (Quelle: BCG (2001), S. 4)

Eine Studie der Boston Consulting Group von 2001 hat 14 europäische Großbanken bzgl. ihrer IT-Aufwendungen untersucht. Wie das Analyseergebnis in Abbildung 2.5 zeigt, bestätigen sich im Mittel die von MOORMANN geschätzten 15% Anteil der IT-Kosten am Verwaltungsgesamtaufwand.[123] Das Ergebnis der Studie macht deutlich, dass es äußerst schwierig ist nachzuweisen, wann eine Über- oder Unterinvestition in IT stattgefunden hat: Bei Banken, deren IT-Kostenanteil über 20% liegt, stellt sich die Frage, ob diese überinvestiert sind, oder ob sie die IT-Kosten strategisch so eingesetzt haben, dass gegenüber anderen Mitbewerbern im Vorteil sind. Bei denjenigen Banken, deren IT-Kostenanteil unter 10% liegt, stellt sich die Frage, ob sie ihre IT-Kosten derart reduziert haben, dass sie unterinvestiert und somit gegenüber

[123] Vgl. Moormann (2004), S. 9ff

dem Wettbewerb im Nachteil sind. Es könnte jedoch auch sein, dass diese Banken gegenüber den anderen Banken es geschafft haben, ihre IT besonders effizient einzusetzen.

FISCHER und ROTHE stellen fest, dass der Unternehmenserfolg maßgeblich durch die Fähigkeit determiniert wird, „die erforderliche IT-Unterstützung intelligenter, schneller, kostengünstiger und in besserer Qualität als die Mitbewerber zur Verfügung zu stellen."[124] Jedoch zeigt eine gemeinsame Studie SAP (2005) der EFMA[125], der Universität Mannheim und der SAP, bei der mehr als ein Drittel der großen Privatkundenbanken in Europa, dem Mittleren Osten und in Afrika befragt wurden, dass die IT-Systeme der Kreditinstitute eher hinderlich als förderlich in Bezug auf die Umsetzung strategischer Ziele seinen. Dies setzt jedoch voraus, dass die strategischen Ziele bekannt sind und konkret formuliert wurden.

MOORMANN sieht eine geringe Flexibilität der Anwendungslandschaft von Banken in ihrer historischen Entwicklung begründet. Größtes Hindernis für die Umsetzung neuer strategischer Ziele seinen die operativen Systeme:[126] Diejenigen Systeme, die sich um die Grundfunktionen einer Bank (Kontokorrent, Zahlungsverkehr etc.) kümmern, sind bei den meisten Kreditinstituten veraltet. Neue Anforderungen sind auf Grund der starr einprogrammierten Geschäftsprozessabläufe nur mit sehr hohem Aufwand umzusetzen. Die Anwendungssysteme einer Bank sind Spartenorientiert aufgebaut und sind zueinander meist inkompatibel. Wird ein Geschäftsprozess über zwei oder mehr Systeme verschiedener Sparten benötigt, ist eine durchgängige Bearbeitung oft nicht sichergestellt. Bei den Systemen steht das Konto und nicht der Kunde im Zentrum. Interoperable Systeme haben zu einer heterogenen Systemlandschaft geführt, deren Betrieb hohe Kosten produziert. Jedes Altsystem (Legacy-System) verfügt meist über eine eigene Datenbasis. So kommt es zu Dateninkonsistenzen und enormen Aufwenden bei der Datenzusammenführung.

---

[124] Fischer/Rothe (2004), S. 22
[125] European Financial Management & Marketing Association
[126] Vgl. Moormann (2004), S. 10ff

Ein weiteres Ergebnis der Studie SAP (2005) ist, dass für neun von zehn Banken Strategien zur Umsatzsteigerung wichtiger sind als die Reduktion von Kosten. Um diesen Schritt zu vollziehen gibt ein Drittel der befragten Kreditinstitute an, die IT gezielt auszubauen, um einen entscheidenden Wettbewerbsvorsprung erzielen zu können. Zusammenfassend hebt die Studie folgende Kernaspekte hervor:[127]

- Wachstum und Gewinnmaximierung stehen weitestgehend im Vordergrund. Ein bewährtes Mittel dieser Gewinnmaximierung ist die Reduktion der Kosten.
- Der Bereich der sog. *Business Intelligence*[128] wird weiter fokussiert. Die befragten Kreditinstitute sind der Ansicht, dass sich das Sammeln, Aggregieren und Analysieren von kundenbezogenen Daten positiv auf die kundenbezogenen Strategien auswirkt.
- Das sog. *Multikanalbanking*[129] wird weiter ausgebaut. Zum einen soll das Multikanalbanking die Automation der Geschäftsprozesse erhöhen und somit für kürzere Reaktionszeiten sorgen. Zum anderen sollen hierdurch die Kunden gezielter angesprochen werden können.
- Die Fähigkeiten der einzelnen Mitarbeiter sowie die gesamte organisatorische Struktur werden an die neuen Gegebenheiten angepasst.
- Dadurch dass die meisten IT-Systeme nicht geeignet sind, um die anvisierten Strategien (Wachstum, Gewinnmaximierung, Kundenorientierung etc.) in die Realität umzusetzen, ist ein Drittel der befragten Kreditinstitute gewillt in neue Technologien zu investieren, die das sog. *Alignment* zwischen den Geschäftsprozessen und den IT-Systemen zu lösen versprechen. Als Trends können die Abkehr von der eigenen IT-Entwicklung und der vermehrte Einsatz von Standard-Software ausgemacht werden.

---

[127] Vgl. SAP (2005), S. 6ff

[128] Anmerkung: Allgemein umfasst der Begriff die analytischen Konzepte, Prozesse und Werkzeuge, um Unternehmens- und Wettbewerbsdaten in entscheidungsrelevantes Wissen zu transformieren. Es werden unternehmensinterne und -externe Daten als Quellen herangezogen. Der Begriff wird dem Fachgebiet der Wirtschaftsinformatik zugerechnet.

[129] Anmerkung: Das Multikanalbanking ist eine Umschreibung für die Möglichkeiten eines Kunden, auf unterschiedliche Arten und Weisen (z. B. per Post, Telefon oder Internet) Bankgeschäfte zu tätigen.

BUBIK ET AL. stellen fest, dass die IT in der Finanzdienstleistungsbranche zwar eine immer wichtigere Rolle spielt, „dennoch […] hat IT jene kritische Bedeutung, die sie im Kampf um mehr Wertschöpfung und Marktanteile schon jetzt einnehmen könnte, bei weitem nicht erreicht.“[130] BUBIK ET AL. schätzen das Potential der IT zur Schaffung strategischer Wettbewerbsvorteile gegenüber anderen Finanzdienstleistern als eher gering ein. Sofern Wettbewerbsvorteile errungen werden, resultiert dies weniger auf konkreten Technologien, sondern eher auf einem strategisch richtigen Einsatz der IT.

In Anlehnung an PORTER führen der bloße Einsatz von Technologien sowie die von FISCHER und ROTHE geforderte schnellere, kostengünstigere und qualitativ hochwertigere IT-Unterstützung, die einen Vorteil gegenüber Wettbewerbern schaffen, lediglich zu einer reinen operativen Effektivität und stößt irgendwann an eine Produktivitätsgrenze, nach deren Erreichen keine Kosten mehr gesenkt werden können oder eine Qualität sich weiter steigern lässt. Diese operative Effektivität ist jedoch nur von kurzfristiger Dauer, und zwar solange, bis die Konkurrenz die Technologien adaptiert hat. Unternehmen sollten nach PORTER daher die strategische Positionierung des Unternehmens überdenken.[131]

In Anlehnung an NIPPA sowie WARD und PEPPARD ist der von PORTER initiierte Fokus auf Wettbewerbsrelevanz von Industrie- und Branchenstrukturen durch die Konzentration auf die ursprünglichen Unternehmenspotentiale abgelöst worden. Das Ziel der Kostenführerschaft ist beispielsweise strategischer Natur.[132]

Strukturelles Problem der Informatik

MOORMANN ET AL. stellen fest, dass das primäre Problem der Bankinformatik struktureller Art ist:[133] Laut MOORMANN „ist die IT-Situation in vielen Instituten alles andere als zufrieden stellend.“[134] Größtes Problem seinen die operativen Systeme, deren Existenz und Aufbau durch die historische Entwicklung in der Bankinformatik begründet werden

[130] Bubik et al. (2000), S. 102
[131] Vgl. Porter (1996)
[132] Vgl. Nippa (1999), S. 6; vgl. Ward/Peppard (2002), s. 98ff; vgl. auch Herzwurm/Langner (2005)
[133] Vgl. Moormann/Wölfing (1999); Vgl. auch Moormann (2004), S. 11
[134] Moormann (2004), S. 10

können. MOORMANN identifiziert insgesamt 5 Phasen, die eine Strukturänderung der Anwendungslandschaft von Banken beeinflusst haben:[135]

- In Phase 1 (1960-1970) ist der Grad der Komplexität des Bankgeschäfts sehr niedrig. Im Vordergrund steht die sog. *Batch-Datenverarbeitung*, bei der die Kernbankensysteme primär das Privatkundengeschäft automatisieren. Buchungssysteme verarbeiten große Datenmengen insbesondere der Kontoführung. Das Umfeld der Banken war von Stabilität gekennzeichnet.
- In Phase 2 (1970-1980) verändert sich das Marktumfeld für Banken und die Komplexität des Bankgeschäfts erreicht einen mittleren Grad. Im Vordergrund steht die sog. *Time-Sharing-Datenverarbeitung*, bei der die Anwendungslandschaft der Banken um Dialogverarbeitung erweitert wird. Ebenfalls wird in dieser Phase der Begriff der „zentralen EDV-Abteilung“ etabliert.
- In Phase 3 (1980-1990) halten PCs Einzug in die Bankinformatik, wodurch die personalisierte Informationsverarbeitung in den Vordergrund rückt. Mitarbeiter in Fachabteilungen sind erstmals in der Lage auf Großrechnerdaten zuzugreifen und diese Informationen auszuwerten. Die vorher zentral geprägten Systeme werden immer dezentraler. Das Marktumfeld für Finanzdienstleister zeigt erste Züge einer Instabilität auf. Der Grad der Komplexität des Bankgeschäfts ist hoch.
- In Phase 4 (1990-2000) steht die Anwendungslandschaft der Finanzdienstleister im Zeichen der sog. *vernetzten Informationsverarbeitung*. Im Vordergrund stehen Client/Server-Applikationen, die weltweit Daten elektronisch austauschen lassen. Die IT wird erstmals als Dienstleistung verstanden. Das Marktumfeld für Finanzdienstleister weist Überraschungen[136] auf. Der Grad der Komplexität des Bankgeschäfts ist sehr hoch.
- In Phase 5 (2000-2010) hat die Komplexität des Bankgeschäfts in Anbetracht des Marktumfeldes einen Grad erreicht, der zum einen

[135] Vgl. Moormann (2004), S. 7ff

[136] MOORMANN erläutert nicht, welche „Überraschungen“ in dieser Phase zum Vorschein kommen. Es lässt sich jedoch vermuten, dass die durch Technologie-Euphorien getriebenen Vorstellungen des Internet-Booms gemeint sind.

sehr hoch ist und zum anderen von einem globalen Wettbewerb bzw. Existenzkampf gezeichnet ist. Als Trend ist die sog. *Web-basierte Informationsverarbeitung* auszumachen, bei der versucht wird sämtliche Geschäftsprozesse zu digitalisieren. Die in Phase 3 dezentralisierten Systeme werden in dieser Phase zentralisiert. Es wird vermehrt auf Standardisierung von Technologien und Infrastrukturen gesprochen. WESTPHAL erkennt in der Softwareentwicklung einen Trend zur Industrialisierung, durch die Softwareentwickler nicht mehr ein breit gefächertes sondern ein spezialisiertes Wissen aufbauen müssen.[137]

Isolierte Projektarbeit, fehlende Architekturvorgaben

Dass die Anwendungssysteme von Finanzdienstleistern so gebaut wurden, wie sie heute zum größten Teil noch existieren, liegt vor allem in der historischen Entwicklung der Bankinformatik begründet. Wie die zuvor von MOORMANN beschriebenen Entwicklungsphasen demonstrieren, haben die einzelnen Dekaden unterschiedliche Anforderungen an die Kernsysteme hervorgebracht. Durch eine Dezentralisierung konnten die einzelnen Fachbereiche ihre eigenen Applikationslandschaften aufbauen, deren Zusammenarbeit heutzutage für große Probleme sorgt.

MARTY hebt hervor, dass diese Applikationen – abgesehen von Programmiervorschriften auf Codierungsebene – weitestgehend ohne Architekturvorgaben gebaut wurden.[138] Je komplexer das Bankgeschäft wurde, desto komplexer und heterogener wurden Applikationslandschaften aufgebaut und nur unzulänglich miteinander gekoppelt.

Auch KLAWA stellt fest, dass das Ergebnis der Praxis, die Fachbereiche eines Finanzdienstleisters selbst über ihre IT-Systeme bzw. die Auswahl von konkreten Produkten und Herstellern für diese Systeme entscheiden zu lassen, die heute als „starr“[139] bezeichneten Applikationslandschaften sind.

Ein Kernproblem der Applikationserstellung ist die Tatsache, dass Projektteams (meist durch Fachabteilungen getrieben) zum Ziel gesetzt wird, die Applikation mit den vereinbarten Funktionen innerhalb der veranschlagten Kosten und der vereinbarten Zeit auszuliefern. Wie HAFNER

[137] Vgl. Westphal (2005)
[138] Vgl. Marty (2004), S. 47
[139] In Anlehnung an SAP (2005)

ET AL. feststellen, werden hierdurch „Fragen der langfristigen Entwicklung der betriebenen Applikationslandschaft […] nicht explizit behandelt.“[140]

Eine organisatorisch-strategische Komponente fehlt

Auch DERN stellt fest, dass in der Praxis von Finanzdienstleistern eine hohe Diskrepanz zwischen der Forderung nach einer strategisch ausgerichteten und gleichzeitig hoch flexiblen IT-Architektur und der alltäglichen Projektpraxis zu Tage tritt.[141] Hauptgrund dieser Diskrepanz ist, so DERN, die „[…] scheinbar unüberwindbare Kluft zwischen der Unternehmens- und der Projektsicht […]“[142].

BIRKHÖLZER und VAUPEL vergleichen dieses Phänomen mit Beobachtungen anderer Bereiche: Einsiedler haben ihre eigenen Gesetze und finden sich in Großstätten nicht zurecht. Ähnlich ist es mit IT-Systemen. Durch das Zusammenrücken und Zusammenwachsen von Einzelapplikationen zu interagierenden Applikationslandschaften ergeben sich neue Anforderungen und Eigenschaften: „isolierte Elemente werden im Wesentlichen von ihren eigenen inhärenten Eigenschaften bestimmt, Elemente eines Ensembles von ihrer Fähigkeit zur Interaktion.“[143]

Als Folge für die Applikationslandschaften muss ein höheres Organ institutionalisiert werden, dass langfristige Ziele vor Augen hat und die Architektur bzw. Interaktion von Einzelapplikationen überwacht und steuert.

2.2 Architekturmanagement und die Komplexität der Infrastruktur

Nach der Diskussion der kausalen Zusammenhänge für die Ausgestaltung der innerbetrieblichen Abläufe und Produktionsfaktoren der Finanzdienstleistungsinformatik, soll nun die als *Architekturmanagement* bekannt gewordene Disziplin als Regulator zwischen dem System Finanzdienstleistungsinformatik und seiner Umwelt Finanzdienstleister vorgestellt werden.

---

[140] Hafner et al. (2004), S. 54
[141] Vgl. Dern (2003), S. VII
[142] Dern (2003), S. VII
[143] Birkhölzer/Vaupel (2003), S. 9

Dass die sog. *Architektur* einen bedeutenden Einfluss auf den Lebenszyklus eines Systems hat, wurde lange Zeit beobachtet. In den Jahren vor 2000 dominierten vorwiegend Hardwarebezogene Architekturaspekte. Durch eine starke Zunahme der Komplexität von Softwaresystemen wuchs jedoch der Bedarf der Integration architektonischer Aspekte in den Softwareentwicklungsprozess.[144]

2.2.1 Architekturmanagement als Regulator zwischen Finanzdienstleistung und Finanzdienstleistungsinformatik

2.2.1.1 Architektur

Es existiert eine Vielzahl von Definition in Bezug auf den Begriff *Architektur* im Kontext der Informationstechnologie. IEEE Std 1471-2000 definiert eine Architektur als „[t]he fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design evolution.“[145] DIEZ ET AL. betrachten Architektur als Menge von Leitlinien innerhalb eines Entwicklungsprozesses.[146]

BIRKHÖLZER und VAUPEL orientieren sich bei der Herleitung des Begriffs vorwiegend an der gleichnamigen Disziplin des Häuserbaus. Um beide Disziplinen voneinander unterscheiden zu können, grenzen sie den Begriff durch die Umbenennung in *IT-Architektur* von der klassischen Disziplin Baukunst wie folgt ab: „IT-Architektur ist die Strukturierung eines Informationsverarbeitenden Systems in eine Summe von Elementen, die Beschreibung des Zusammenspiels und der gegenseitigen Abhängigkeiten der Elemente. IT-Architektur bezeichnet dabei sowohl die Tätigkeit als auch die Elemente.“[147] Die zentrale Frage der Disziplin IT-Architektur ist, wie die Komplexität von IT-Umgebungen mit Hilfe einer geeigneten Strukturierung beherrscht werden kann. BIRKHÖLZER und VAUPEL unterteilen die Disziplin in die drei Teildisziplinen *Intra-*

[144] Vgl. IEEE (2000), S. iii
[145] IEEE (2000), S. 3
[146] Vgl. Diez et al. (1999)
[147] Birkhölzer/Vaupel (2003), S. 19

Komponenten-Architektur, *Intra-System-Architektur* und *Inter-System-Architektur*.[148]

WINTER bzw. LICHTENEGGER ET AL. unterscheiden prinzipiell vier Formen von Architektur im Umfeld von Informationssystemen:[149] Eine *Geschäftsarchitektur* stellt den Gesamtzusammenhang der Leistungsverflechtungen in einem Wertschöpfungsnetzwerk dar. Eine *Prozessarchitektur* stellt den Gesamtzusammenhang der Leistungsentwicklung, -erstellung und des –vertriebs in einer Organisation dar. Eine *Applikationsarchitektur* stellt den Gesamtzusammenhang der informatorischen Verflechtung von Applikationen einer Organisation dar. Eine *IT-Architektur* übernimmt die Darstellung der gesamten Zusammenhänge der Verflechtung zwischen Software und Datenbankkomponenten.

KRÜGER und SEELMANN-EGGEBERT stellen fest, dass es keine etablierte Definition für den Begriff Architektur im Umfeld von Informationssystemen gibt. Sie benennen unterschiedliche Ausprägungen des Begriffs und führen u. a. folgende Begriffsvariationen vor: Software-/Anwendungsarchitektur, Componentware-Architektur, Informationsarchitektur, Groupware-Architektur, Datenarchitektur. In diesem Zusammenhang definieren KRÜGER und SEELMANN-EGGEBERT: „Die IT-Architektur eines Unternehmens stellt die Gesamtheit aller Komponenten, Technologien und organisatorischen Maßnahmen dar, die im Unternehmen vorkommenden Funktionen, Prozesse und Daten abbilden und deren Zusammenspiel ermöglichen.“[150]

DERN unterteilt Architektur ebenfalls in vier Bereiche (vgl. Abbildung 2.6), die ineinander verzahnt die Voraussetzung für ein integriertes und methodisch fundiertes Vorgehen zur übergreifenden Planung von IT-Architekturen bilden:[151]

Business-Architektur

> Die sog. *Business-Architektur* repräsentiert die formalisierte Beschreibung der geschäftlichen Ausrichtung eines Unternehmens oder Geschäftsfeldes. Sie teilt sich in eine Prozessarchitektur und eine Organisationsarchitektur auf. Erstere definiert und strukturiert die Geschäftsprozesse eines Unternehmens sowie deren Informations-

[148] Vgl. Birkhölzer/Vaupel (2003), S. 21ff
[149] Vgl. Winter (2003); vgl. Lichtenegger et al. (2003)
[150] Krüger/Seelmann-Eggebert (2003), S. 29
[151] Vgl. Dern (2003), S. 11ff

bedarf. Die Organisationsarchitektur befasst sich mit der organisatorischen Ausgestaltung des Unternehmens, so dass die Geschäftsprozesse der Prozessarchitektur optimal ausgeführt werden können.

Informationsarchitektur

> Die sog. *Informationsarchitektur* definiert die strukturierten Prinzipien und Regeln, die für die Gestaltung der Anwendungslandschaft wegweisend sind. Elemente der Informationsarchitektur sind das sog. *IS-Portfolio*, das eine systematische Aufstellung der Informationssysteme eines Unternehmens enthält, die sog. *Technologiestrategie*, die die Leitlinien für die Entwicklung der IT-Basisinfrastruktur festlegt, die sog. *Architekturstrategie*, die den zu erreichenden Zielzustand der Prozesse und der IT-Systeme beschreibt, und die sog. *Architekturprinzipien*, die DERN als „[…] alle architekturbezogenen Grundsätze und übergreifenden Standardisierungen, die für die Weiterentwicklung des IS-Portfolios […] und der zugehörigen Architekturen gelten“[152] versteht.

IT-Architektur

> Die sog. *IT-Architektur* ist die strukturierte Abstraktion existierender oder geplanter Informationssysteme. Mittels ihrer Abstraktion soll die IT-Architektur für alle an der Gestaltung von Informationssystemen Beteiligten eine gemeinsame Kommunikationsplattform bilden. Zur Herbeiführung eines Verständnisses der IT-Architekturen werden eine statische und eine dynamische Sicht auf die Architekturen benötigt. Die statische Sicht wird durch die sog. *Anwendungsarchitektur* beschrieben, die die sich in die Unterdisziplinen *Fachliche Architektur* (fachliche Funktionalitäten und Daten), *Softwarearchitektur* (Art, Struktur und Zusammenwirken der Softwarebausteine) und *System- und Sicherheitsarchitektur* (Abbildung auf die IT-Basisinfrastruktur) aufteilt. Das Zusammenspiel der drei Elemente der Anwendungsarchitektur wird im Sinne einer dynamischen Sicht als Modell des Softwareentwicklungsprozesses betrachtet.

IT-Basisinfrastruktur

> Die sog. *IT-Basisinfrastruktur* umfasst die Menge aller Hard- und systemnahen Softwarekomponenten, die die Laufzeit- und Managementumgebung für Entwicklung, Test und Produktion von Informationssystemen bilden. Sie ist ein selbständiges Handlungsfeld des IT-

[152] Vgl. Dern (2003), S. 26f

Infrastruktur-Managements und wird über die in der Informationsarchitektur definierte Technologiestrategie mit der Business-Architektur verknüpft.[153] Nach IEEE 1471-2000 tragen Architekturbeschreibungen der IT-Basisinfrastruktur dazu bei, die Ausgestaltung und Verwaltung der Infrastruktur, das Konfigurationsmanagement sowie den Neuentwurf und die Pflege der Systeme, Subsysteme und Komponenten dieses Bereichs zu unterstützen.[154]

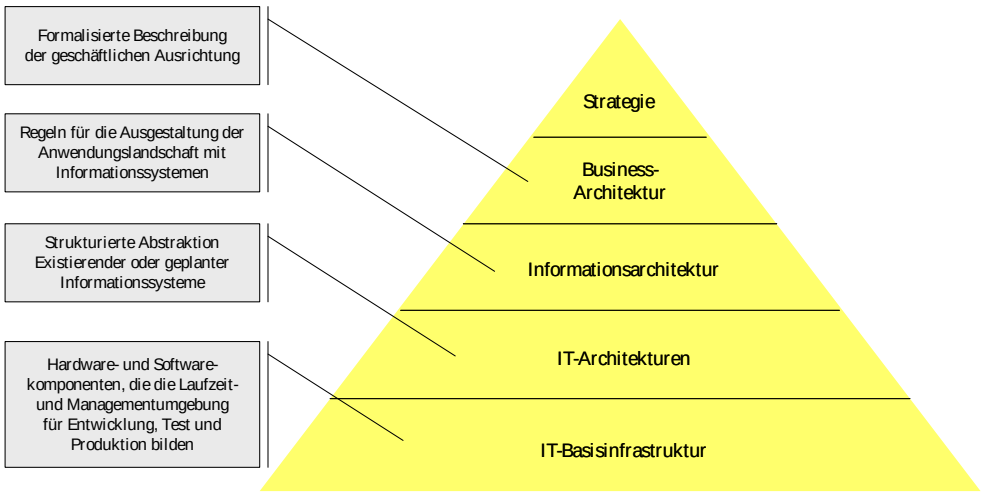


Abbildung 2.6: Elemente der Architekturpyramide. (In Anlehnung an Dern (2003))

Zusammenfassend kann nun auf Basis der gewonnenen Erkenntnisse definiert werden:

Begriffsdefinitionen: *Infrastruktur, Infrastrukturkomponente, Infrastruktursystem, Variante bzw. Infrastrukturkomponentenvariante*

Eine *Infrastrukturkomponente* ist entweder eine Hardware- oder eine systemnahe Softwarekomponente, die eine spezifische Aufgabe im Rahmen der IKT übernimmt. Eine Infrastrukturkomponente ist ein Teil der die Laufzeit- und Managementumgebung für Entwicklung, Test und Produktion von Informationssystemen bildenden Komponentenmenge.

Infrastruktur umfasst die Menge aller Infrastrukturkomponenten. Diese Menge ist die Gesamtheit aller der zur Speicherung, Verar-

[153] Anmerkung: Diese Definition ist in einem hohen Maße deckungsgleich mit dem in Krcmar (2005) als *Infrastruktur* verwendeten Begriff.
[154] Vgl. IEEE (2000), S. 8

beitung, Administration und Kommunikation zur Verfügung stehenden Ressourcen.

Ein *Infrastruktursystem* ist eine Aggregation von Infrastrukturkomponenten, die Teil der Infrastruktur sind, die in Summe eine spezifische Aufgabe für mindestens eine Applikation übernimmt.

Eine *Variante* einer Hardware- bzw. Software-technischen Komponente, kurz *Infrastrukturkomponentenvariante*, ist eine andere Hardware- bzw. Software-technische Komponente gleichen Zwecks, die sich in mindestens einer Eigenschaft oder einer Relation zu einer anderen Komponente unterscheidet.

2.2.1.2 Architekturmanagement im Allgemeinen

Eine konkrete Definition für den Begriff *Architekturmanagement* liefern HAFNER ET AL.:[155] Ziel des Architekturmanagements im allgemeinen Sinne ist es, die klassischen Aufgaben des Informationsmanagements[156] zu unterstützen. Dies umfasst die Schaffung und Aufrechterhaltung einer geeigneten Informationsinfrastruktur zur wirtschaftlichen Erreichung strategischer Unternehmensziele. Architekturmanagement verfolgt zur Vermeidung weiterer organisatorischer Komplexität keine eigenständigen Zielsetzungen. Weiterhin integriert das Architekturmanagement kontinuierlich neue technologische und fachliche Herausforderungen und gewährleistet die Beherrschbarkeit des unternehmensweiten Informationssystems.

DERN liefert keine explizite Definition des Begriffs Architekturmanagement, umschreibt jedoch implizit Architekturmanagement als alle diejenigen Aktivitäten, die für die organisatorische Implementation der von ihm vorgestellten Architekturpyramide[157] notwendig sind. Diese Implementation beinhaltet eine koordinierte und systematische Weiterentwicklung der Anwendungslandschaft und fokussiert nicht einzelne Informationssysteme, sondern betrachtet die Gesamtheit aller Informationssysteme eines Unternehmens. Dadurch, dass jedes neue System

[155] Vgl. Hafner et al. (2004)
[156] Vgl. Heinrich (2002), S. 21 bzw. Hildebrand (2001), S. 14ff
[157] Vgl. Dern (2003), S. 11ff; siehe *Abbildung 2.6*

die interne Komplexität des Systems Finanzdienstleistungsinformatik erhöht, muss es Aufgabe des Architekturmanagements sein, diese Komplexität nicht weiter ansteigen zu lassen bzw. diese Komplexität sukzessiv zu reduzieren.[158]

2.2.1.3 Architekturmanagement als Regulator

Wird das Architekturmanagement auf Basis seiner Definition als Regulator betrachtet, der das Verhältnis zwischen interner und externer Komplexität der Finanzdienstleistungsinformatik optimiert, kann dieser Regulator die interne Komplexität derselben reduzieren.[159] Dieser Regulator muss jedoch nicht nur die Einflüsse seiner direkten Umwelt regulieren, sondern auch die der Umwelt der technologischen Neuerungen (Abbildung 2.7).

Die Problemstellungen, mit denen sich die Finanzdienstleistungsinformatik auseinandersetzen muss, sind sowohl technischer als auch organisatorischer Natur. Die Informationssysteme eines Finanzdienstleisters sind über Jahrzehnte aufgebaut, modifiziert und zum Teil auch wieder abgebaut worden. Die von MOORMANN als „strukturell“ bezeichneten Probleme resultieren aus dem Umstand, dass das System Finanzdienstleistungsinformatik sich über die Jahre an seine Umwelt angepasst hat. Ändert sich die Umwelt der Finanzdienstleister, versucht dieser sein System möglichst schnell an die neue Umwelt anzupassen. Da die Finanzdienstleistungsinformatik als ein Kernbereich eines Finanzdienstleisters Teil der Ausgestaltung des Systems Finanzdienstleister ist, muss auch diese sich ihrer lokalen Umwelt anpassen. Ist dies aus technologischen oder organisatorischen Gründen auf direktem Weg nicht in der gleichen Zeit möglich, wie dies im System Finanzdienstleister vonstatten geht, sorgt der Druck des dominierenden Systems Finanzdienstleister auf das untergeordnete System Finanzdienstleistungsinformatik, dass indirekte Wege gegangen werden. Diese indirekten Wege fördern den Aufbau neuer Komplexität.

---

[158] Vgl. Dern (2003), S. 11ff

[159] Zur Komplexitätsreduktion durch Prozessregulation siehe Firchau et al. (2002)

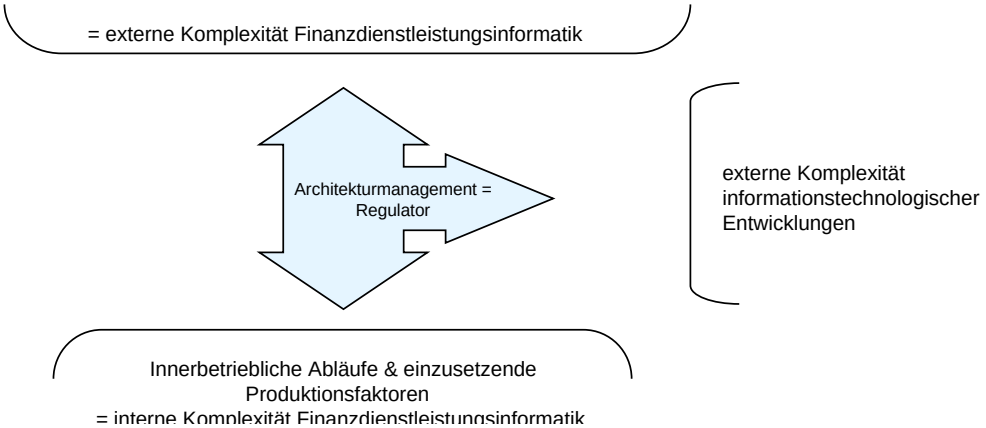


Abbildung 2.7: Architekturmanagement als Regulator der Finanzdienstleistungsinformatik. (Quelle: Eigene Darstellung)

2.2.2 Komplexität der Infrastruktur

Für die von PILLER als „[…] das Zusammentreffen einer strukturellen Vielschichtigkeit, resultierend aus der Anzahl und Diversität der Elemente eines Systems sowie deren gegenseitige Verknüpfung und der dynamischen Veränderlichkeit der gegenseitigen Beziehungen der Systemelemente“[160] definierte Komplexität gibt es eine vielfältige Anzahl von Beispielen. Jedes System ist komplex. Im Vergleich zu anderen Systemen kann es komplexer, gleich komplex oder weniger komplex sein. Nach FRANKE bestimmen folgende Treiber den Grad der Komplexität:[161]

- Die Anzahl der Systemelemente.
- Die Anzahl der unterschiedlichen Systemelemente und die Ungleichmäßigkeit ihrer Aufteilung (= *Partition*).
- Die Anzahl der Relationen und die topologische Unsymmetrie ihrer Lage.
- Die Anzahl der unterschiedlichen Relationen, die Ungleichmäßigkeit ihrer Aufteilung und die topologische Unsymmetrie ihrer Anordnung.
- Das Vorkommen von gerichteten Relationen.
- Die Anzahl von Zyklen und ihre Ungleichheit im Durchmesser sowie die Unsymmetrie ihrer topologischen Lage.

[160] Piller (2000), S. 179
[161] Vgl. Franke (1998)

> Das Vorkommen von mehrstelligen Relationen.

Komplexität entsteht bereits durch die Kombination verschiedener Objekte auf Basis von Regeln. Eine „unnötige“ interne Komplexität wird nach FRANKE oft bereits dadurch erzeugt, dass unterschiedliche Organisationseinheiten andersartige Bezeichnungen und Klassifikationen für gleiche Objekte, Regeln oder Sachverhalte verwenden.[162]

Die von FRANKE aufgezählten Komplexitätstreiber lassen sich im Bereich der Infrastruktur von Informationssystemen wieder finden:[163] Als Beispiel für die Komplexität nennt KRCMAR den Aufbau eines Enterprise Resource Planning Systems (ERP-Systems), das auf Basis einer Client/Server-Architektur implementiert ist. Neben geeigneten Hardware-Infrastrukturkomponenten wie den Rechnern mitsamt ihrer CPUs werden auch Software-Infrastrukturkomponenten wie beispielsweise ein Betriebssystem benötigt. Die Kommunikation zwischen Client und Server erfolgt über ein spezifisches Protokoll. Die Anzahl der Kombinationsmöglichkeiten der technischen Realisation wächst mit der Elementanzahl, der Anzahl der Elementtypen und den Regeln ihrer Kombinierbarkeit.

Die von KRCMAR verwendete Bezeichnung *Elementtyp* wird im Kontext des Komplexitätsmanagements als *Variante* bezeichnet. Nach DIN 199 umfasst eine Variante Gegenstände ähnlicher Form und/oder Funktion mit in der Regel hohem Anteil identischer Gruppen oder Teile.[164] Da der in dieser Arbeit fokussierte Anwendungsbereich jedoch die Infrastruktur der Finanzdienstleistungsinformatik ist, und diese per Definition nicht nur aus Hardware-Komponenten (Gegenständen) sondern auch aus systemnahen Software-Komponenten besteht, ist diese Definition einer Variante unzulänglich. Daher wird eine Variante in Anlehnung an FRANKE und FIRCHAU wie folgt definiert:

Begriffsdefinition: *Variante*

Eine *Variante* einer Hardware- bzw. Software-technischen Komponente ist eine andere Hardware- bzw. Software-technische Kompo-

[162] Vgl. Franke (1998)
[163] Vgl. Krcmar (2005), S. 213
[164] Vgl. DIN 199

nente gleichen Zwecks, die sich in mindestens einer Eigenschaft oder einer Relation zu einer anderen Komponente unterscheidet.

Ein Beispiel für eine Hardware-Komponentenvariation ist ein Server, der entweder eine Sparc- oder eine Intel-basierte Prozessorarchitektur verwendet. Der Zweck beider Varianten ist der technische Betrieb eines Softwaresystems. Die Eigenschaften, die sich bei beiden Varianten unterscheiden, sind beispielsweise der Hersteller oder die Prozessorarchitektur.

Ein Beispiel für eine Software-Komponentenvariation ist ein Java Runtime Environment, kurz JRE, das entweder für das Betriebssystem Solaris oder das Betriebssystem AIX erhältlich ist. Der Zweck beider Varianten ist die Übersetzung sog. *Byte Codes* in die Maschinensprache des verwendeten Betriebssystems. Die Eigenschaften beider JREs sind identisch, jedoch unterscheiden sie sich in jeweils in einer Beziehung zu einem Betriebssystem (und somit implizit zu einer Prozessorarchitektur).

Welche Kombinationsmöglichkeiten sich theoretisch bereits bei drei unterschiedlichen Elementen mit jeweils zwei Varianten und zwei Relationsvarianten ergeben, wird in Abbildung 2.8 angedeutet. Ausgehend von der reinen Kombinierbarkeit der Elemente a, b und c gibt es bereits vier mögliche Kombinationen auf Basis der unterschiedlichen Relationsvarianten. Die Komplexität des Systems wächst in die vertikale Richtung mit der Varietät der jeweiligen Elementtypen a’, a’’, b’, b’’, c’ und c’’. Die Komplexität des Systems wächst in die horizontale Richtung mit der Anzahl der Relationsvarianten.

- Sei a ein Betriebssystem und gebe es zwei Betriebssystemvarianten (Linux, Solaris).
- Sei b ein J2EE Applikationsserver und gebe es zwei Applikationsserver-varianten (BEA WebLogic und JBoss).
- Sei c ein Webserver und gebe es zwei Webservervarianten (Tomcat, Java Web Server).

- Gebe es zwischen b und a sowie zwischen c und a eine direkte Abhängigkeit.
- Gebe es für die Relation zwischen b und c zwei Relationsvarianten (Netzübergang interner Partner, Netzübergang externer Partner).

Dieses System besitzt bereits 16 verschiedene Kombinationsvarianten. Erhöht sich die Anzahl der Varianten, ihrer Kombinationsregeln und der Relationsvarianten bereits minimal, wächst die Zahl der Kombinationsmöglichkeiten exponentiell. Die Folge ist eine Komplexitätszunahme. Der Anzahl und Verschiedenartigkeit dieser Komplexitätstreiber ergibt sich aus der Komplexität des jeweils vorangestellten Systems.

PENZEL charakterisiert die Vielfalt der in einem Rechenzentrum eines Finanzdienstleisters auffindbaren sog. *Betriebsumgebungen* als „fast unüberblickbar". Er unterteilt eine Betriebsumgebung in sieben logische Schichten: Präsentation, Anwendung, Kommunikation/Schnittstellen, Datenbank, Betriebssystem, Hardware und Netzwerk. In jeder Schicht werden mehrere Software- und Hardware-Produkte verwendet. Jedes dieser Produkte besitzt wiederum eine Vielfalt unterschiedlicher im Betrieb befindlicher Versionen. Die durch eine Ist-Aufnahme identifizierten Betriebsumgebungen der HVB besitzen 150 Varianten. PENZEL schätzt, dass diese Varianten auf 30 reduziert werden können, ohne den Betrieb der Informationssysteme einzuschränken.[165]

In Anlehnung an Abschnitt 2.1.2 (insbesondere Abbildung 2.2) ist die Komplexität der Infrastruktur eines Finanzdienstleisters das Resultat der Regulation der Marktinduzierten Komplexität innerhalb des Systems Finanzdienstleister und dessen Regulation auf das System Finanzdienstleistungsinformatik. Eine Ursache hierfür ist nach MARTY der Umstand, dass die Informationssysteme von Finanzdienstleistern ohne umfassende Architekturvorgaben gebaut wurden.[166]

[165] Penzel (2004), S. 124f
[166] Vgl. Marty (2004), S. 47

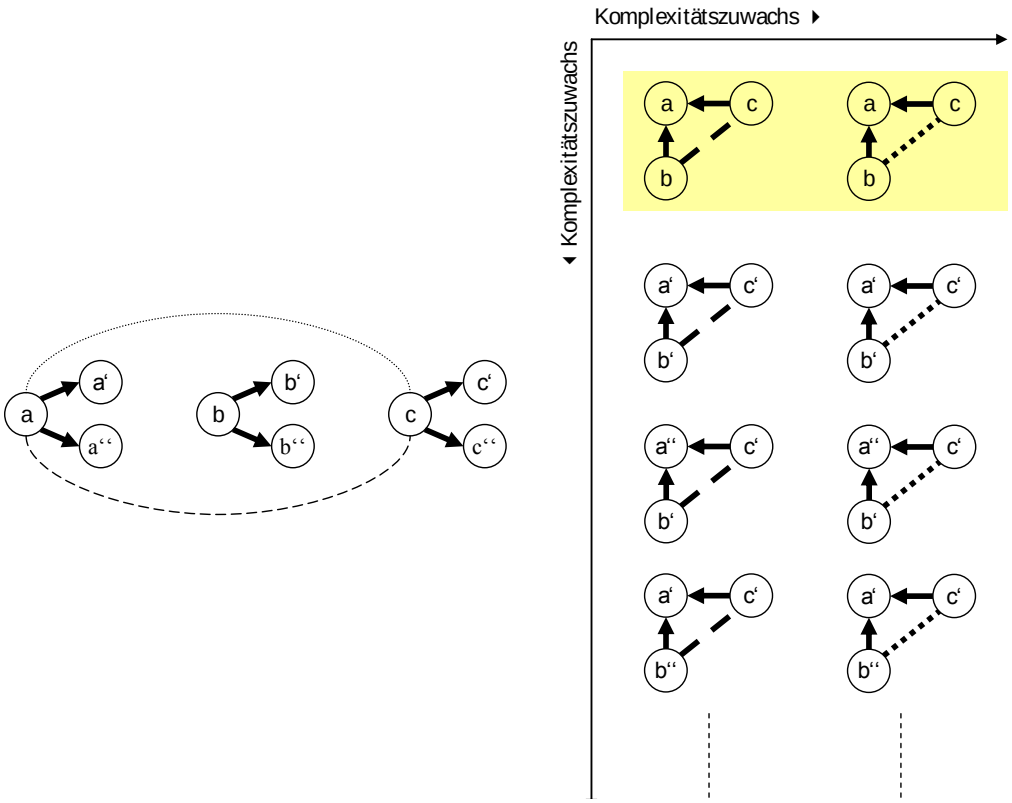


Abbildung 2.8: Komplexitätswachstum. (Quelle: Eigene Darstellung)

3 Praktischer Bezug

Das forschungsmethodische Vorgehen in dieser Arbeit stellt einen praktischen Bezug in den Vordergrund. Gemäß ULRICH wird die Analyse der Praxis benötigt, um aus ihr Regeln und Modelle zur induktiven Ableitung der Erkenntniszusammenhänge entwickeln zu können.[167] Daher wird in diesem Kapitel in die beiden Fallstudien eingeführt, die maßgeblich für die Ausgestaltung des Lösungsvorschlags dieser Arbeit in Bezug auf die zentrale Problemstellung verantwortlich sind.

3.1 Fallstudie 1

Die Erfassung der organisatorischen Gegebenheiten erfolgte im Zeitraum Oktober 2004 bis Januar 2005. Bis zum Juni 2006 fanden lediglich kleine organisatorische Änderungen statt, die keine Auswirkungen auf die im Folgenden präsentierte Organisationsform haben. Die Erhebung der konkreten Problemidentifikation fand im Zeitraum Januar 2005 bis April 2005 im Rahmen von anonymisierten Einzelinterviews statt, da nur so gewährleistet werden konnte, dass der Forscher die komplexen organisatorischen Zusammenhänge in Verbindung mit den konkreten Problemausprägungen verstehen konnte. Insgesamt wurden 18 Mitarbeiter des befragt.

3.1.1 Erfassung der organisatorischen Gegebenheiten

Analog zu dem in Abbildung 3.1 dargestellten Aufbau gliedert sich die ca. 1000 Personen beschäftigende PBS vornehmlich in drei sog. *Ressorts*: Technologiemanagement, Projekte und Betrieb. Ein Ressort ist ein organisatorischer Teil innerhalb der PBS, dem ein Vorstandsmitglied vorsteht.

Ressort ***Technologie-management***

Das Ressort *Technologiemanagement* ist mit ungefähr 40 Mitarbeitern zuständig für die Verwaltung der für die Pflege und Weiterentwicklung der Informationssysteme eingesetzten Technologien. Hierunter fällt

[167] Vgl. Ulrich (1984), S. 192ff

auch das Architekturmanagement dieser Informationssysteme. Die Festlegung und Vermarktung der Zielsetzung der Komplexitätsreduktion und der langfristigen Kostensenkung ist eine der Grundvoraussetzungen für die Ausgestaltung des Architekturmanagements. Analog zu ZACHMAN wird Architekturmanagement als Investition betrachtet.[168] Wie in Abbildung 3.1 dargestellt, existieren in der nun aktuellen Ausgestaltung vier Domänen. Eine *Domäne* ist eine thematisch fokussierte Gruppe von Mitarbeitern, die jeweils Teil einer von drei Abteilungen sind. Insgesamt gibt es drei Informationssystem-Domänen und eine Infrastruktur-Domäne. Eine *Informationssystem-Domäne* verwaltet eine logische Zusammenführung von verschiedenen fachlich gleichen Informationssystemen, die auf Grund ihres Einsatzgebietes (z. B. Gesamtbanksteuerung oder Abwicklungssysteme) thematisch zusammengefasst werden können. Der Grund für diese Aufteilung ist, dass jeder Themencluster von Informationssystemen nahezu die gleichen qualitativen Ansprüche besitzt. So müssen beispielsweise die Informationssysteme der Domäne Abwicklungssysteme sehr hohe Datenvolumina in kurzen Zeiten verarbeiten und dürfen praktisch nie ausfallen. Diese qualitativen Anforderungen sind letztendlich Richtungsgeber für die technische Architektur der Informationssysteme. Als weitere Erkenntnis wurde gewonnen, dass den Informationssystemen einer Domäne gemeinsame Strategien zugrunde liegen. Beispielsweise ist eine der Geschäftsstrategien der PBS das sog. *Insourcing* von Zahlungsabwicklungen. Hierbei werden die Zahlungsabwicklungen der Dresdner Bank und der Deutschen Bank von der Postbank übernommen. Diese Strategie hat Auswirkungen auf die IT und deren Strategien. Letztendlich ergibt sich ein qualitativer Anspruch an die Architektur der Informationssysteme der Domäne Transactionbanking in der Form, dass die Systeme möglichst flexibel („mandantenfähig“) gestaltet werden.

Neben den drei Informationssystem-Domänen wurde eine vierte sog. *Infrastruktur-Domäne* geschaffen. Die Existenzberechtigung dieser Domäne beruht auf der Feststellung, dass sämtliche Informationssysteme, die von den restlichen drei Domänen verwaltet werden, auf dem-

[168] Vgl. Zachman (2001)

selben Infrastrukturportfolio aufsetzen. Unbestritten ist, dass die hohen Kosten für den Betrieb von Systemen auf die Vielzahl unterschiedlicher Plattformvarianten zurückzuführen ist.[169] Die Aufteilung der Fachbereiche bzw. deren Anfragen auf die drei Domänen bedeutet eine Regulation von Informationen. Das Clustern der Informationen ermöglicht Teilausschnitte der IT-Architektur (so wie sie z. B. von DERN definiert wird) detaillierter zu analysieren als dies mit dem Gesamtsystem möglich wäre. Diese Regulation basiert auf dem Modularisierungsgedanken[170] und ist die Grundvoraussetzung für eine Komplexitätsreduktion.

Die Kernaufgaben dieses Ressorts werden von zwei Rollen abgedeckt: Die *Domänenarchitekten* sind für denjenigen architektonischen Teil eines Informationssystems zuständig, der nicht die Infrastruktur desselben betrifft. Die Infrastrukturarchitektur wird von der Rolle des *Infrastrukturarchitekten* abgedeckt. Einer oder mehrere Infrastrukturarchitekten haben sind auf ein bestimmtes Infrastrukturthema spezialisiert. Um die Zusammenarbeit mit dem Ressort Betrieb zu vereinfachen, entspricht die thematische Spezialisierung der einzelnen Architekten der organisatorischen Aufteilung des Ressorts Betrieb in Form der Plattformen.

[169] Siehe auch Rausch/Rothe (2004)
[170] Vgl. Baldwin/Clark (1998)

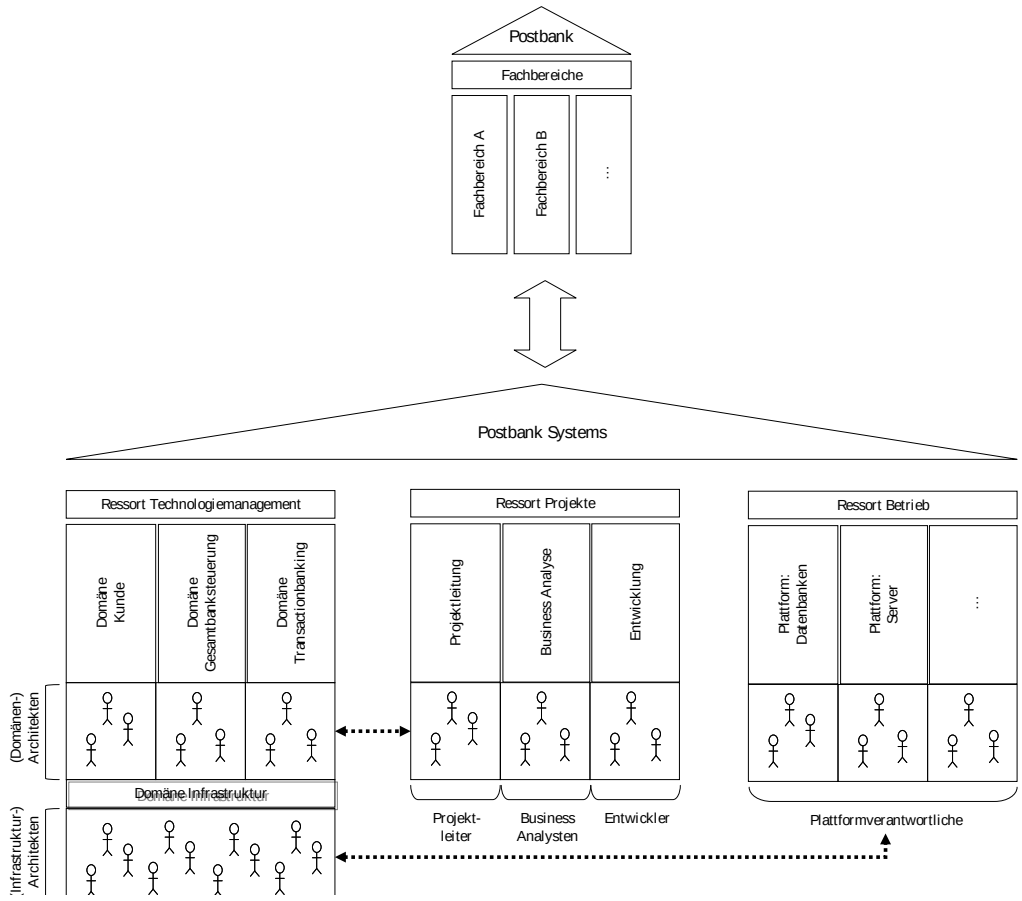


Abbildung 3.1: Grober organisatorischer Aufbau der Postbank Systems. (Quelle: Eigene Darstellung)

Ressort ***Projekte***

Die Durchführung eines Projekts erfordert unterschiedliche Rollen. Die Steuerung der Projekte wird von gleichnamigem Ressort übernommen. Für die Ressourcenplanung und –überwachung ist die Projektleitung zuständig. Auf Basis einer konkreten Projektanfrage fordert der *Projektleiter* die notwendigen Ressourcen an und reserviert diese für das jeweilige Projektvorhaben. Die Analyse der Kundenanforderungen sowie deren Spezifikation wird von den *Business Analysten* übernommen, die die Anforderungsspezifikation in einer Form aufbereiten, die von anderen Projektbeteiligten verstanden wird. Die Implementierung von Software innerhalb eines Softwareprojekts wird von den *Entwicklern* übernommen.

Ressort ***Betrieb***

Der technische Anteil eines Informationssystems wird von den Mitarbeitern des Ressorts *Betrieb* überwacht, administriert und gesteuert. Die einzelnen technischen Komponenten sind in sog. *Plattformen* eingeteilt. Beispiele für Plattformen sind Unix Server oder Storage Systeme. Die Einführung und Überwachung technischer Komponenten ist die Aufga-

be der *Plattformverantwortlichen*. Die Überwachung der sich im Betrieb befindlichen technischen Komponenten ist Aufgabe der *Administratoren*.

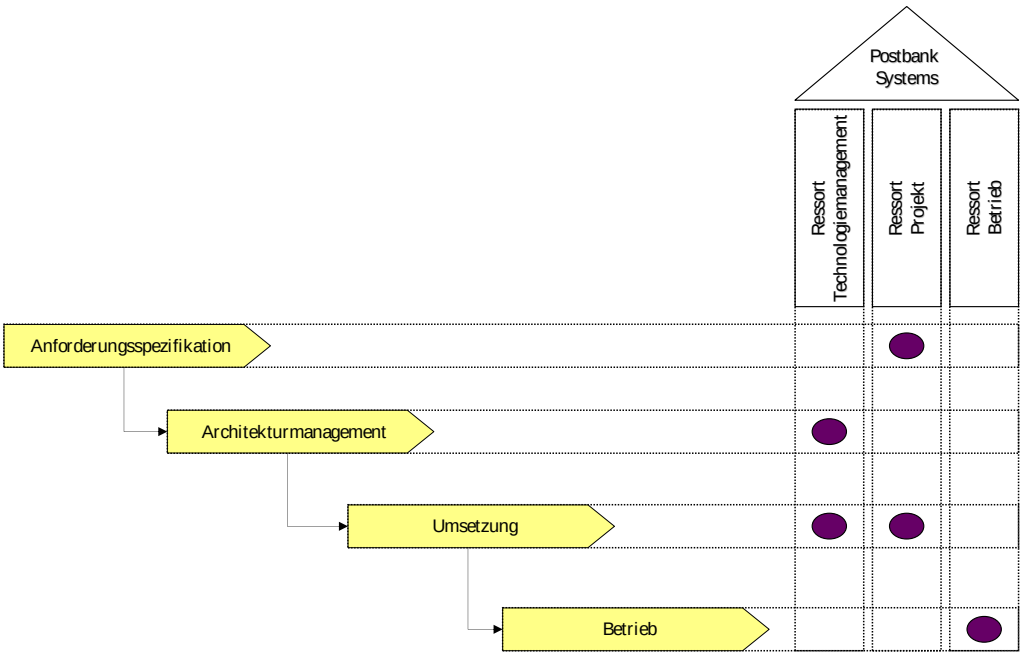


Abbildung 3.2: Integration des Architekturmanagements in die typische Vorgehensweise bei Projekten der Postbank Systems. (Quelle: Eigene Darstellung)

Vorgehensmodell

Der Prozess, der bei der Erstellung neuer Applikationen bzw. der Wartung und Weiterentwicklung existierender Applikationen verfolgt wird, ist in Abbildung 3.2 auszugsweise dargestellt: Durch die Aufnahme der Anforderungsspezifikation ist das Architekturmanagement in der Lage, das Aufgabengebiet einer Domäne zuzuweisen. Die Domänen erkennen eigenständig, ob Interaktionen mit Systemen anderer Domänen notwendig sind, berücksichtigen technische Neuerungen und streben einen Komplexitätsabbau der Systeme an. Nach dem Architekturentwurf kann die Umsetzung und letztendlich der Betrieb der Applikation erfolgen. Diese Vorgehensweise ist unternehmensweit durch das sog. *Postbank Vorgehensmodell* (PBVGM) vorgeschrieben. Das PBVGM ist eine Anpassung des *Rational Unified Process* (RUP) an die organisatorischen Gegebenheiten der PBS.

3.1.2 Problemidentifikation

3.1.2.1 Problemdimension 1: Komplexität der Infrastrukturarchitektur

Problemausprägungen im Allgemeinen

Die Eigenart und Vielfältigkeit der von der PB angebotenen Finanzdienstleistungen ist ein Komplexitätstreiber der Infrastruktur. Die große Anzahl der Kunden sowie die Abwicklung des Zahlungsverkehrs anderer Kreditinstitute im Rahmen des Insourcings erfordert hochleistungsfähige und Desaster Recovery fähige Systeme. Im Fall des Ausfalls eines Rechenzentrums muss ein zweites Rechenzentrum diesen Ausfall abfangen. Die Erfüllung dieser Anforderungen erfordert eine Vielzahl unterschiedlicher, hochqualitativer Infrastrukturkomponenten wie zum Beispiel Serversysteme, Datenbanken oder Storage Systeme. Kunden der PB haben neben dem Filialbesuch u. a. auch die Möglichkeit ihre Bankgeschäfte über das Internet abzuwickeln. Diese Form des Dienstleistungsangebots erfordert wiederum andere Infrastrukturkomponenten wie beispielsweise Web Server oder Applikationsserver. Auch diese Komponenten müssen den extremen Qualitätsanforderungen bzgl. der Stabilität und Performance gerecht werden. Neben diesen extremen Anforderungen an Infrastrukturkomponenten, die primär seitens des Zahlungsabwicklungs- bzw. Privatkundenbereichs gestellt werden, existieren auch geringere Anforderungen an Infrastrukturkomponenten. Beispiele hierfür sind Informationssysteme von Fachbereichen, die lediglich von Mitarbeitern einzelner Abteilungen verwendet werden. Auch diese geringerqualifizierten Anforderungen weisen unterschiedliche Varianten auf. Sie reichen von Systemen, die nur wenige Stunden nachts arbeiten, bis hin zu Systemen, die nur wenige Stunden tagsüber verwendet werden, die jedoch in kurzen Zeitintervallen starken Belastungen ausgesetzt sind.

Die Variantenvielfalt der in den diversen Informationssystemen eingesetzten Infrastrukturkomponenten ist sehr groß. Nahezu jede Infrastrukturkomponente – ganz gleich ob Software- oder Hardware – wird von unter-schiedlichen Herstellern bezogen. So werden beispielsweise verschiedene Betriebssysteme (Windows, Linux, Unix, Solaris etc.), Serversysteme (HP9000, Alpha-Server, pSeries etc.), Datenbanksysteme (Oracle RAC, DB2, SQL Server etc.) sowie Applikationsserver (Net-

Weaver, WebLogic, WebSphere, JBoss etc.) eingesetzt. Diese Variantenvielfalt wird zusätzlich vergrößert, indem jede Variante in verschiedenen Untervarianten eingesetzt wird, die sich durch die von den Herstellern vorgegebenen Versionen ergeben.

Problemausprägungen im Speziellen

Die Beantwortung der spezifischen Fragen ergab die in Tabelle 3.1 abgedruckten Ergebnisse. Die Enthaltungen im Fall der Frage 1 sind darauf zurückzuführen, dass 3 Interviewpartner bisher noch nie aktiv mit Softwarearchitektur in Kontakt gekommen sind. Bei der Beantwortung der Frage 2 konnte festgestellt werden, dass je nach Themengebiet, mit dem sich der jeweilig Befragte täglich auseinandersetzt (z. B. Storage Systeme, Netzkomponenten etc.), die Dynamik unterschiedlich eingeschätzt wird. Während die Dynamik im Fall der Netzwerkkomponenten eher gering bis mittel stark ausgeprägt ist, wurde die Dynamik aus Sicht derjenigen Mitarbeiter, deren Hauptaufgabengebiet sich auf die Bereiche Server und Netzwerke konzentriert, beispielsweise als hoch eingestuft.

Im Fall der Frage 4 fällt die sehr konzentrierte Fokussierung auf die Bewertung „wenig“ auf. Ein genannter Grund hierfür ist die Unkenntnis in Bezug auf existierende Ansätze.

Relevant für die Gestaltung eines Modells, das sich mit der Komplexität der Infrastruktur auseinandersetzt, ist die Erfassung und Analyse vorhandener Tools oder Methoden, welche im untersuchten Unternehmen eingesetzt werden. Die Mehrheit der Befragten setzt keine speziellen Verfahren ein, die die Infrastruktur beherrschbar machen. Diejenigen, die ein spezielles Verfahren einsetzen, setzen dies jedoch nur für den Eigengebrauch ein.

#	Frage	Ergebnis
1	Wie beurteilen Sie den Grad der Komplexität der Infrastrukturarchitektur im Vergleich zur klassischen Softwarearchitektur?	Weniger komplex: 2x gleich komplex: 11x komplexer: 2x
2	Wie beurteilen Sie den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betreffen?	gering: 3x mittel: 5x hoch: 10x
3	Wie beurteilen Sie den Grad der Variantenvielfalt einzelner von der Infrastrukturarchitektur erfasster Komponenten?	gering: 0x mittel: 10x hoch: 8x
4	Inwieweit helfen die Ihnen bekannten Ansätze	mir sind keine Ansätze be-

	die Komplexität der Infrastrukturarchitektur zu beherrschen?	kannt: 5x überhaupt nicht: 3x wenig: 8x ausreichend: 2x bestens: 0x
5	Setzen Sie spezielle Tools, Methoden, etc. ein, die die Komplexität der Infrastrukturarchitektur in Ihrem Unternehmen beherrschbar machen? Wenn ja, verwenden Sie diese nur alleine oder werden sie von Ihren Kollegen ebenfalls eingesetzt?	nein: 14x ja, ich verwende spezielle Tools, Methodiken, etc.: 4x ja, ich verwende spezielle Tools, Methodiken, etc. die auch von meinen Kollegen eingesetzt werden: 0x
6	Wie wichtig ist die Beherrschung der Komplexität der Infrastrukturarchitektur in Ihrem Unternehmen?	nicht wichtig: 0x wichtig: 4x sehr wichtig: 14x

Tabelle *3.1*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 1 (Fallstudie 1).

3.1.2.2 Problemdimension 2: Wissenskomplexität und Wissensvolumen

Problemausprägungen im Allgemeinen

Das Ergebnis des vorherigen Abschnitts ist die Feststellung, dass die Infrastrukturarchitektur der PBS komplex ist. In Anlehnung an KELLER und WENDT ist die Rolle eines Architekten im Allgemeinen die eines Mediators.[171] Er ist Ansprechpartner des Kunden, des Managements und der Technik. Er muss demzufolge technische Zusammenhänge verstehen können, sie mit einem Techniker diskutieren können, und die Ergebnisse dem Management bzw. den Kunden verständlich machen.

Die Rolle eines Architekten im Speziellen bei der PBS zeigt ein ähnliches Bild: Der Unterschied ist, dass diese Rolle in Form der Infrastruktur- und Domänenarchitekten zweifach spezialisiert ist. Der Kunde eines Infrastrukturarchitekten ist ein Domänenarchitekt, sein technischer Ansprechpartner ist ein Plattformverantwortlicher. Das Management, dem er Bericht erstatten muss, ist der Vorstand.

Die Wissenskomplexität sowie das Wissensvolumen einzelner als Plattformen bezeichneter Themengebiete ist sehr hoch. Dies führt dazu, dass das notwendige Wissen, das für eine fundierte Beratung und Konsultation der mit einem Infrastrukturarchitekten kommunizierenden Rollen benötigt wird, von immer nur einer Person vollständig beherrscht wird. Somit ist ein Abruf dieses Spezialwissens begrenzt auf die Verfügbarkeit dieser Person im Sinne einer Resource.

[171] Vgl. Keller/Wendt (2003)

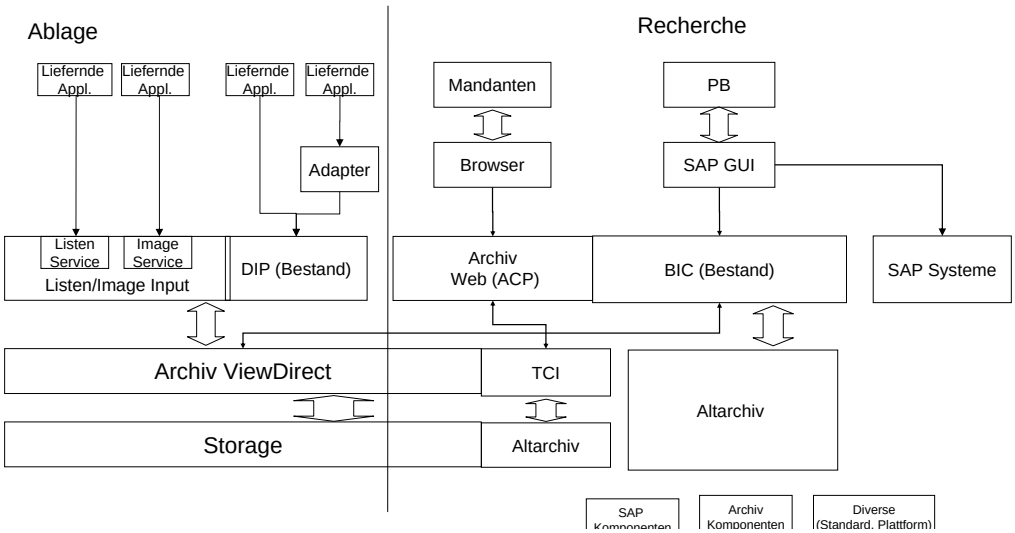


Abbildung 3.3: Ein Beispiel für eine Dokumentationsnotation. (Quelle: PBS interne Darstellung)

Bei der Untersuchung der PBS konnte festgestellt werden, dass die Architekten größtenteils keine standardisierte Sprache zur Dokumentation von Architekturentwürfen verwenden. Wesentliche Designelemente wurden den Produkten Visio und Powerpoint entnommen. Somit unterschieden sich Notationen von Architekt zu Architekt. Sofern beispielsweise ein Architekt, dessen Spezialgebiet Datenbanken sind, seinen Architekturentwurf mit dem eines Architekten kombinieren muss, dessen Spezialgebiet J2EE Applikationsserver sind, wurden jeweils zwei verschiedene Notationen zu einer neuen Sprache kombiniert. Diese Sprache ist jedoch solange unverständlich für die Kommunikationspartner der Architekten, solange diese ihren Entwurf nicht zusätzlich textuell dokumentieren. Ebenfalls konnte beobachtet werden, dass sich die Notation eines Architekten von Entwurf zu Entwurf ändern kann, sodass zwei Entwürfe zwar die gleichen Informationen enthalten, die Notationen jedoch erheblich voneinander abweichen. Ein Beispiel für unterschiedliche Notationsformen ist in den Abbildungen Abbildung 3.3 und Abbildung 3.4 abgedruckt.[172] Ohne detaillierter auf beide Abbildungen einzugehen sei angemerkt, dass dies zwei Darstellungen desselben Informationssystems sind, die jedoch von zwei verschiedenen Personen mit Power Point erstellt wurden.

[172] Anmerkung: Zur Veröffentlichung dieser Grafiken wurden selbige leicht verändert.

Hinzu kommt, dass ein Architekt auf Grund des Wissens über sein Spezialgebiet bestimmte, für ihn selbstverständliche und damit nicht erwähnenswerte, für andere Beteiligte jedoch unbekannte Informationen nicht in seinen Architekturentwürfen darstellt.

Die für Außenstehende intransparenten Entscheidungswege, wie ein Infrastrukturarchitekt zu seiner Lösung gekommen ist, tragen nicht zu einer Beherrschung von Wissensvolumen und Wissenskomplexität bei.

Problemausprägungen im Speziellen

Die Beantwortung der spezifischen Fragen ergab die in Tabelle 3.2 abgedruckten Ergebnisse. Bei der Beantwortung der Frage 1 gab es 3 Enthaltungen auf Grund unzureichender Kenntnisse im Bereich der Softwarearchitektur.

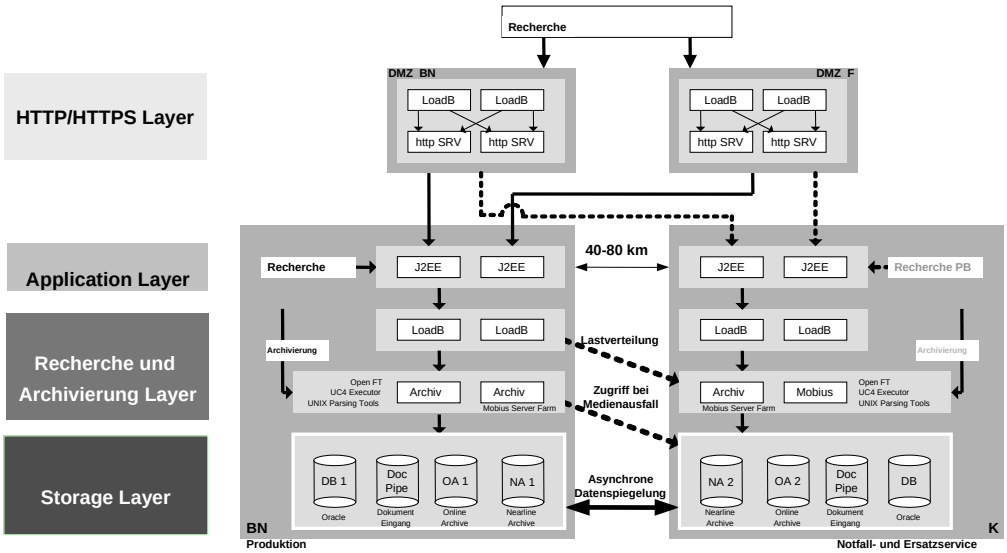


Abbildung 3.4: Dasselbe Beispiel aus anderer Perspektive und mit anderer Dokumentationsnotation. (Quelle: PBS interne Darstellung)

Die Ergebnisse der Frage 3 reflektieren im Wesentlichen die organisatorische Struktur des Unternehmens. Da die Wissenskomplexität sowie das Wissensvolumen einzelner als Plattformen bezeichneter Themengebiete von Einzelpersonen beherrscht wird, entstehen sog. *Kopfmonopole*. Um dieses auf einzelne Personen beschränkte Spezialwissen allgemein zugänglich zu machen, wird eine einheitliche, von allen Beteiligten verstandene Sprache benötigt, die die Dokumentation dieses Wissens ermöglicht. Tests mit ausgewählten Dokumentationen in Form von Visio Zeichnungen oder Power Point Präsentationen haben ergeben, dass selbst dokumentiertes Wissen von anderen Beteiligten nicht verstanden wird.

#	Frage	Ergebnis
1	Wie beurteilen Sie den Grad der Wissenskomplexität bzw. den Grad des Wissensvolumens im Umfeld der Infrastrukturarchitektur im Vergleich zur klassischen Softwarearchitektur?	Weniger komplex: 4x gleich komplex: 9x komplexer: 2x
2	Inwieweit helfen die Ihnen bekannten Ansätze die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur zu beherrschen?	mir sind keine Ansätze bekannt: 5x überhaupt nicht: 3x wenig: 9x ausreichend: 1x bestens: 0x
3	Ist das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme in mehr als 60% der Fälle auf einzelne Personen beschränkt („Kopfmonopole“) oder ist das Wissen allgemein zugänglich dokumentiert?	Kopfmonopole: 12x allgemein zugänglich: 6x
4	Ist allgemein zugängliches Wissen mit Hilfe einer einheitlichen Notation dokumentiert?	ja: 5x nein: 13x
5	Sind die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar dokumentiert, d. h. können Sie in mehr als 60% aller Fälle nachvollziehen weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A entschieden hat und gegen Typ B?	ja: 8 nein: 5 von Fall zu Fall unterschiedlich: 5
6	Setzen Sie spezielle Tools, Methoden, etc. ein, die die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur in Ihrem Unternehmen beherrschbar machen? Wenn ja, verwenden Sie diese nur alleine oder werden sie von Ihren Kollegen ebenfalls eingesetzt?	nein: 10x ja, ich verwende spezielle Tools, Methodiken, etc.: 4x ja, ich verwende spezielle Tools, Methodiken, etc. die auch von meinen Kollegen eingesetzt werden: 4x
7	Wie wichtig ist die Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur Ihrem Unternehmen?	nicht wichtig: 0x wichtig: 5x sehr wichtig: 13x

Tabelle *3.2*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 2 (Fallstudie 1).

Die Zustimmung, dass allgemein zugängliches Wissen mit Hilfe einer einheitlichen Dokumentation festgehalten wird (Frage 4), ist in 3 von 5 Fällen darauf zurückzuführen, dass die Interviewpartner zur Dokumentation ihres Wissens, die Dokumentation ihrer Kollegen wiederverwenden. Praktisch bedeutet dies das Kopieren der Folien eines Kollegen in die eigene Power Point Präsentation.

Die organisatorische Aufteilung in Themenverantwortliche wird als Methode verstanden, die Wissenskomplexität bzw. das Wissensvolumen beherrschbar zu machen.
Insgesamt ergibt sich ein konsistentes Ergebnis zur Erhebung in Abschnitt 3.1.2.1.

#	Frage	Ergebnis
1	Wie beurteilen Sie den Grad der Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in Ihrem Unternehmen auswirkt?	gering: 1x mittel: 4x hoch: 9x sehr hoch: 4x
2	Wie beurteilen Sie den Grad des Kostendrucks, der auf Ihr Unternehmen ausgeübt wird?	gering: 0x mittel: 1x hoch: 6x sehr hoch: 2x

Tabelle *3.3*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 3 (Fallstudie 1).[173]

3.1.2.3 Problemdimension 3: Historische Entwicklung der Finanzdienstleistungsinformatik

Problemausprägungen im Allgemeinen

Die allgemeine Problemausprägung der Finanzdienstleistungsinformatik hängt im Wesentlichen von der historischen Entwicklung des Umsystems Bank ab. Die strategische Ausrichtung der Bank mit der Anforderung die Vormachtstellung als größte Privatkundenbank Deutschlands weiter auszubauen kombiniert mit der IT-strategischen Ableitung, die Kostenführerschaft im Bereich der finanzdienstleistungsspezifischen Verwaltungskosten zu übernehmen, prägen die Entscheidungswege innerhalb des Systems Finanzdienstleistungsinformatik. Primäres Ziel des operativen Geschäfts ist es, wirtschaftlich zu arbeiten und die Kosten für die Bereitstellung der diversen IT-Dienstleistungen ohne Qualitätsverlust weiter zu senken.

Problemausprägungen im Speziellen

Diese allgemeinen Problemausprägungen stimmen mit den in Tabelle 3.3 abgedruckten Problemausprägungen im Speziellen überein. Da nahezu die gesamte IT der Bank in dem untersuchten Unternehmen ausgelagert wurde, entsprechen die anfallenden Gesamtkosten für die Bereitstellung der IT-Dienstleistung den dem Auftraggeber in Rechnung gestellten Beträgen. Somit sind die Kosten ein wichtiges Instrument der

[173] Anmerkung: Bei der Beantwortung der Frage 2 gab es mehrere Enthaltungen.

Finanzdienstleistung zur Steuerung der Finanzdienstleistungsinformatik.

3.2 Fallstudie 2

Die Erfassung der organisatorischen Gegebenheiten mitsamt der Erhebung der konkreten Problemidentifikation erfolgte im Februar 2006. In Form eines eintägigen Workshops wurde eine Gruppenbefragung der 5 Teilnehmer durchgeführt. Die BHW wurde offiziell im Januar 2006 in den Postbank Konzern integriert. Die Strukturen und Abläufe der im Rahmen dieser Arbeit fokussierten Organisationseinheiten wurden erst nach Mai 2006 denen der PBS angepasst.

3.2.1 Erfassung der organisatorischen Gegebenheiten

Das betrachtete Unternehmen unterscheidet primär die beiden Organisationseinheiten *Fachbereiche* und *IT* (Abbildung 3.5). Wesentlicher Unter-schied zur PBS ist, dass die IT in diesem Fall nicht in ein separates Unternehmen ausgegliedert ist. Während sich jeder Fachbereich auf ein am Markt angebotenes Produktsegment konzentriert (z. B. Lebensversicherungen, Baudarlehen etc.) unterteilt sich die ca. 400 Personen starke IT grob in die organisatorischen Untereinheiten *Architektur*, *Entwicklung* und *Betrieb*.

Der Organisationseinheit *Architektur* gehören lediglich 5 Mitarbeiter an. Das Hauptaufgabengebiet dieser Architekten ist die logische Enterprise Architektur wie sie von ZACHMAN verstanden wird.[174] Die Infrastrukturarchitektur ist jedoch nur unwesentlich repräsentiert.

Ferner gibt es Software Architekten in der Organisationseinheit *Entwicklung* sowie Technologiespezialisten für ausgewählte Themenbereiche (z. B. Datenbanken, Netzwerke etc.) in der Organisationseinheit *Betrieb*.

Das im weiteren Verlauf dieser Arbeit abstrakt definierte Rollenkonzept kann auf die BHW wie folgt übertragen werden:

[174] Vgl. Zachman (1987)

- Die Rolle *Kunde* wird bei der BHW von den Fachbereichen wahrgenommen.
- Das *Management* wird durch den CIO und dessen Management Team repräsentiert.
- Die Rolle *Infrastrukturarchitekt* wird von den Mitarbeitern der Organisationseinheiten *Architektur*, *Entwicklung* und *Betrieb* wahrgenommen, da je nach Themengebiet der betreffenden Infrastrukturkomponente das domänenspezifische Wissen entweder durch einen Mitarbeiter der Architektur (z. B. JBoss Applikationsserver), der Entwicklung (z. B. Oracle RAC) oder des Betriebs (z. B. SAP R/3) zur Verfügung gestellt werden kann.

Des Weiteren kann festgestellt werden, dass die Entwickler lediglich organisatorisch der IT angehören. Rein physisch werden sie eher Fachbereichsnah eingesetzt. Neue Softwarearchitekturen müssen mit einem der 5 Enterprise Architekten diskutiert werden, bevor sie in die Realität umgesetzt werden können. Die Enterprise Architekten kommunizieren ebenfalls intensiv mit den Technologiespezialisten des Betriebs, um neu einzuführende Infrastrukturkomponenten bzw. Änderungen an den bestehenden Systemen abzustimmen.
Während die Architektur als solche bei der PBS durch ein sehr stark formalisiertes Vorgehensmodell geprägt ist, agiert das Architekturteam der BHW weitestgehend frei und Situationsgetrieben, sodass ein agiler Charakter sichtbar wird.

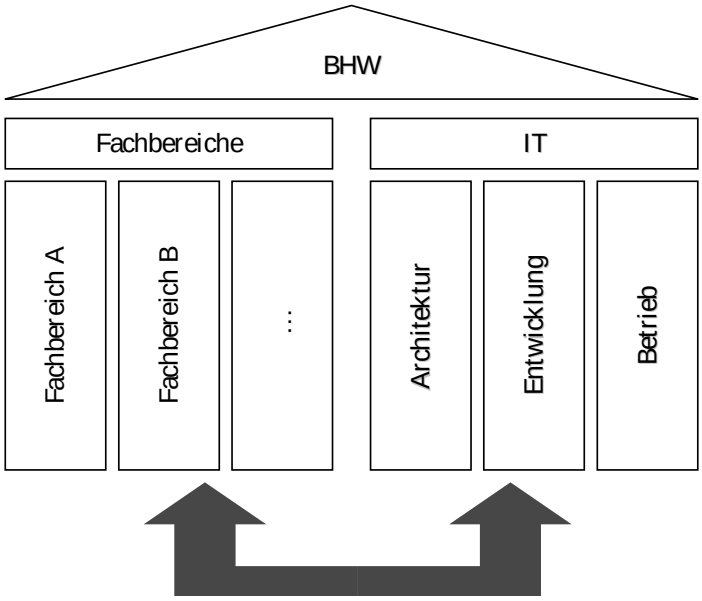


Abbildung 3.5: Grober organisatorischer Aufbau der BHW. (Quelle: Eigene Darstellung)

3.2.2 Problemidentifikation

3.2.2.1 Problemdimension 1: Komplexität der Infrastrukturarchitektur

Problemausprägungen im Allgemeinen

Das Unternehmen BHW wurde 1928 gegründet. Seither hat das Unternehmen Informationssysteme eingeführt, modernisiert und abgeschafft. Die technische Entwicklung über die Jahrzehnte hinweg entspricht im Wesentlichen der allgemein von MOORMANN als „Entwicklungsstufen der Bankinformatik“ bezeichneten Evolution der Finanzdienstleistungsinformatik:[175] In den 60ger Jahren wurden erste Buchungssysteme auf Mainframe-Systemen eingeführt. Später wurden diese Systeme um Dialoganwendungen ergänzt, die nach dem sog. *Time-Sharing-Verfahren* arbeiten. In den 80ger Jahren wurden zudem PC Systeme eingeführt die in den 90ger Jahren mit neuen Client/Server-basierte Technikwelten kommunizierten. Mit dem Ende der 90ger Jahre wurde zunehmend auf die Web-basierte Technologie zurückgegriffen.

Die Komplexität der Infrastrukturarchitektur im betrachteten Unternehmen ergibt sich aus der Einführung einer Vielzahl heterogenster Technologien und Technologievarianten. Wie bereits von PENZEL bzw. DIET-

[175] Vgl. Moormann (2004)

RICH und SCHIRRA beschrieben,[176] wurden neue Technologien auch bei der BHW meist nicht über eine zentral koordinierende Stelle eingeführt. Vielmehr konnten Projekte unter kurzfristig angelegten Kosten-/Nutzengesichtspunkten selbst entscheiden, welche Technologiekombinationen verwendet werden. Das sukzessive Hinzufügen neuer Funktionen sowie die Einführung neuer Systeme, die auf andere, bestehende Systeme zugreifen, sind Komplexitätstreiber der Infrastrukturarchitektur.

Die über die Jahre gewachsene IT-Infrastruktur besitzt eine vielschichtige Architektur, bei der die Altsysteme größtenteils durch immer weitere Zugriffsschichten (Middleware-Systeme) abgekapselt wurden. Kombiniert mit einer weitestgehend unkoordinierten Einführung neuer Technologien ist die Variantenvielfalt der unterschiedlichen Infrastrukturkomponenten sehr hoch. So laufen bspw. die verschiedensten Datenbanksysteme auf den unterschiedlichsten Betriebssystemen.

Ein Beispiel für die Komplexität, die sich aus der Vielzahl eingesetzter Technologievarianten und –Kombinationen ergibt, resultiert aus der Schwierigkeit sog. *Patches* einzuspielen. Angenommen die Datenbank vom Typ *X* greift auf das Storage System vom Typ *Y* zu. Auf selbiges Storage System greift ebenfalls eine in einem anderen Projekt eingeführte Datenbank vom Typ *X'* zu, wobei *X'* eine Variante von *X* ist. Wenn nun der Hersteller des Storage Systems einen Patch ausliefert, der Fehler in der Software des betriebenen Produkts behebt, kann es u. U. dazu kommen, dass *X'* nicht mehr auf *Y* zugreifen kann, da der Hersteller von *Y* den Support für *X'* aufgekündigt hat.

Problemausprägungen im Speziellen

Die spezifische Befragung der Workshopteilnehmer ergab das in Tabelle 3.4 abgedruckte Ergebnis: Auf Basis der Befragung wird auch in diesem Fall die Infrastrukturarchitektur als komplex mit hoher Änderungsdynamik der variantenreichen Komponenten eingestuft.

Als bekannter Ansatz, der die Komplexität beherrschbar macht, wird die Plattformstrategie[177] angeführt. Seine praktische Umsetzung findet dieser Ansatz in der Bildung von fixen Kombinationen speziell selektierter

[176] Vgl. Penzel (2004) bzw. Dietrich/Schirra (2004), S. 4ff
[177] Vgl. Hofer (2001)

Infrastrukturkomponenten – vergleichbar mit den Technologie-Sets von PENZEL.[178]

#	Frage	Ergebnis
1	Wie beurteilen Sie den Grad der Komplexität der Infrastrukturarchitektur im Vergleich zur klassischen Softwarearchitektur?	Weniger komplex: 0x gleich komplex: 3x komplexer: 2x
2	Wie beurteilen Sie den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betreffen?	gering: 0x mittel: 1x hoch: 4x
3	Wie beurteilen Sie den Grad der Variantenvielfalt einzelner von der Infrastrukturarchitektur erfasster Komponenten?	gering: 0x mittel: 0x hoch: 5x
4	Inwieweit helfen die Ihnen bekannten Ansätze die Komplexität der Infrastrukturarchitektur zu beherrschen?	mir sind keine Ansätze bekannt: 0x überhaupt nicht: 0x wenig: 1x ausreichend: 3x bestens: 1x
5	Setzen Sie spezielle Tools, Methoden, etc. ein, die die Komplexität der Infrastrukturarchitektur in Ihrem Unternehmen beherrschbar machen? Wenn ja, verwenden Sie diese nur alleine oder werden sie von Ihren Kollegen ebenfalls eingesetzt?	Nein: 1x ja, ich verwende spezielle Tools, Methodiken, etc.: 1x ja, ich verwende spezielle Tools, Methodiken, etc. die auch von meinen Kollegen eingesetzt werden: 3x
6	Wie wichtig ist die Beherrschung der Komplexität der Infrastrukturarchitektur in Ihrem Unternehmen?	nicht wichtig: 0x wichtig: 1x sehr wichtig: 4x

Tabelle *3.4*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 1 (Fallstudie 2).

Diese als Plattformen bezeichneten Kombinationen von Infrastrukturkomponenten sind zum Zeitpunkt der Befragung auf wenige Themengebiete beschränkt. Lediglich der sog. *Online-Bereich* (Webserver, J2EE Server) verfügt über vordefinierte Plattformen. Eine Plattform schreibt vor, welcher Servertyp (Webserver oder J2EE Server) mit welchem Betriebssystem auf welcher Hardware für zukünftige Projekte verwendet werden muss. Eine detailliertere Aufteilung in Architekturmuster sowie eine Attributierung bzgl. qualitativer Eigenschaften existieren nicht. Vielmehr wird Projekten die Vorgabe gemacht, für Neuentwicklungen bspw. nur noch den J2EE Server JBoss AS 4.0.4 auf Red Hat Enterprise Linux 4 mit x86 Prozessoren zu verwenden. Die detail-

[178] Vgl. Penzel (2004), S. 124; vgl. Abschnitt 5.4

lierte Konfiguration auf Basis der qualitativen Anforderungen an diese Plattform erfolgt erst im Projekt.

3.2.2.2 Problemdimension 2: Wissenskomplexität und Wissensvolumen

Problemausprägungen im Allgemeinen

Auch im Fall der BHW bestimmt die hohe Komplexität der Infrastrukturarchitektur den Grad der Wissenskomplexität bzw. des Wissensvolumens. Ähnlich wie bei der PBS ist das Wissen bzgl. der technischen Informationssystemarchitekturen im Einzelnen das Kopfmonopol einer begrenzten Anzahl von Mitarbeitern.

Die Architektur hat im betrachteten Unternehmen lediglich eine beratende und keine Weisungsbefugte Funktion. Als Berater des Managements, der Entwicklung und des Betriebs sind sie vom alltäglichen Projektgeschäft entkoppelt und werden in ausgewählten Projekten tätig. Ihr Wissen bzgl. der Informationssystemarchitekturen ist daher eher oberflächlich anzusehen. In Bezug auf einzelne Informationssysteme ist es jedoch umso tiefgründiger. Relevant ist das Wissen über die Verknüpfung der einzelnen Informationssysteme untereinander. Das Wissen der Entwicklung ist Projekt-spezifisch und konzentriert sich auf die Softwarearchitektur einzelner Informationssysteme. Die Realisierung der Infrastrukturarchitektur eines einzelnen Informationssystems ist Aufgabe des Betriebs. Details über konkrete Ausprägungen kennen nur die Betreuer einer Applikation. Hinzu kommt, dass sich die Architektur primär auf die „neue Welt“ konzentriert hat, d. h. dass Web-basierte Systemlandschaften und Systeme zur Schaffung einer Service-Orientierten Architektur fokussiert werden.

Die Vorgehensweise innerhalb derartiger Projekte wird zwar formal in einem Vorgehensmodell vorgeschrieben, die tatsächliche Aktion innerhalb eines Projekts wird jedoch agil an die Projektsituation angepasst. Ein Erfolgsfaktor der Wissensreproduktion innerhalb eines konkreten Projekts ist, diejenigen Personen (Kopfmonopolisten) innerhalb der Entwicklung bzw. des Betriebs ausfindig zu machen, die die tatsächliche Ausgestaltung des Informationssystems kennen. Die Zeit wird in Projekten immer als kritischer Faktor angesehen. Daher werden Dokumentationen in den meisten Fällen „auf das Nötigste“ beschränkt.

In diesem Umfeld agieren die Architekten in ausgewählten Projekten als Mediatoren zwischen Management, Entwicklung und Architektur. Sie versuchen Vorgaben bzgl. der Ausgestaltung neuer bzw. abzuändernder Systeme aktiv durch Beratung in die Projekte hineinzutragen. Änderungen an der Infrastruktur werden durch den Betrieb an das Management herangetragen.

Auch in diesem Fall kann festgestellt werden, dass das Wissensvolumen und die Wissenskomplexität der Infrastrukturarchitektur nicht in Gänze beherrscht wird. Die in den jeweiligen Dokumentationen verwendeten Notationen (insbesondere in Bezug auf die Infrastrukturarchitektur) sind nicht standardisiert. Dies erschwert das Verständnis des festgehaltenen Wissens anderer Personen.

#	**Frage**	**Ergebnis**
1	Wie beurteilen Sie den Grad der Wissenskomplexität bzw. den Grad des Wissensvolumens im Umfeld der Infrastrukturarchitektur im Vergleich zur klassischen Softwarearchitektur?	weniger komplex: 0x gleich komplex: 2x komplexer: 3x
2	Inwieweit helfen die Ihnen bekannten Ansätze die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur zu beherrschen?	mir sind keine Ansätze bekannt: 3x überhaupt nicht: 0x wenig: 2x ausreichend: 0x bestens: 0x
3	Ist das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme in mehr als 60% der Fälle auf einzelne Personen beschränkt („Kopfmonopole“) oder ist das Wissen allgemein zugänglich dokumentiert?	Kopfmonopole: 4x allgemein zugänglich: 1x
4	Ist allgemein zugängliches Wissen mit Hilfe einer einheitlichen Notation dokumentiert?	ja: 2x nein: 3x
5	Sind die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar dokumentiert, d. h. können Sie in mehr als 60% aller Fälle nachvollziehen weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A entschieden hat und gegen Typ B?	ja: 3 nein: 0 von Fall zu Fall unterschiedlich: 2
6	Setzen Sie spezielle Tools, Methoden, etc. ein, die die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur in Ihrem Unternehmen beherrschbar machen? Wenn ja, verwenden Sie diese nur alleine oder werden sie von Ihren Kollegen ebenfalls eingesetzt?	nein: 2x ja, ich verwende spezielle Tools, Methodiken, etc.: 0x ja, ich verwende spezielle Tools, Methodiken, etc. die auch von meinen Kollegen eingesetzt werden: 3x

7	Wie wichtig ist die Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur Ihrem Unternehmen?	nicht wichtig: 1x wichtig: 4x sehr wichtig: 0x

Tabelle *3.5*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 2 (Fallstudie 2).

Problemausprägungen im Speziellen

Die in Tabelle 3.5 abgedruckten Ergebnisse der Befragung bzgl. der speziellen Problemäusprägungen der Problemdimension 2 zeigen eine Tendenz, die bereits in den Problemausprägungen der Problemdimension 1 beobachtet wurde: Obwohl Plattformen zur Komplexitätsreduktion eingesetzt werden, zeigt das Ergebnis der Frage 3, dass Kopfmonopole einen wesentlichen Anteil an der Infrastrukturarchitektur haben. Die Begründung der Interviewpartner ist der Umstand, dass die neu geschaffenen Plattformen nur einen geringen Anteil an der gesamten Infrastrukturarchitektur haben. Wesentlich schwieriger schätzen sie die Aufgabe ein, eine detaillierte Analyse der aktuell betriebenen Variantenvielfalt zu erstellen. Oftmals kennen nur einzelne Personen die exakte Kombination von Infrastrukturkomponenten, die eine Applikation betreibt.

Sofern das Wissen bzgl. der Infrastrukturarchitektur im Allgemeinen dokumentiert wurde, ist es von Fall zu Fall unterschiedlich mit einer einheitlichen Notation dokumentiert worden. Einer der Interviewpartner wies darauf hin, dass insbesondere die jüngeren Kollegen eine ähnliche wenn nicht gleiche Notation zur Dokumentation verwenden. Auch in diesem Unternehmen werden hauptsächlich Power Point und Visio als Dokumentationstools im Infrastrukturbereich verwendet.

Für ihren eigenen Verantwortungsbereich haben die Interviewpartner eigene Methoden entwickelt, um das notwendige Wissensvolumen beherrschbar zu machen. Im Fall der Architektur wird innerhalb der Gruppe der Architekten versucht ein Abbild der Systemlandschaft zu erstellen. Die dabei angewendete Methodik ist eine Eigenkreation. Im Fall der Entwicklung wird UML als einheitliche Notation zur Dokumentation der Infrastrukturarchitektur verwendet. Allerdings wurde darauf hingewiesen, dass lediglich das sog. *Deployment-Diagramm*, das als Standard nur rudimentäre Möglichkeiten der Dokumentation von Infrastrukturkomponenten bietet, hierfür verwendet wird.

3.2.2.3 Problemdimension 3: Historische Entwicklung der Finanzdienstleistungsinformatik

Die dritte Problemdimension ist bei der BHW eher mittel stark ausgeprägt. Das 1928 als Beamtenheimstättenwerk gegründete Unternehmen hat im Laufe der Jahre viele Veränderungen durchgemacht. Hierzu zählen zum einen die markttechnischen Veränderungen der Unternehmensstruktur (z. B. Ausgliederung diverser Geschäftsbereiche in Tochterunternehmen) sowie die Marktbedingten Veränderungen technischer Natur (z. B. Nutzung des Internets als Vertriebskanal). Neben diesen Einflüssen haben auch Gesetzesänderungen sowie neue, an den Markt angepasste Produkte die Dynamik erhöht. Auch die Finanzdienstleistungsinformatik der BHW steht unter dem Druck, die Kosten für die adäquate Bereitstellung der IT kontinuierlich zu optimieren.
Diese Tendenzen belegen auch die in Tabelle 3.6 abgedruckten Befragungsergebnisse.

#	**Frage**	**Ergebnis**
1	Wie beurteilen Sie den Grad der Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in Ihrem Unternehmen auswirkt?	gering: 1x mittel: 3x hoch: 1x sehr hoch: 0x
2	Wie beurteilen Sie den Grad des Kostendrucks, der auf Ihr Unternehmen ausgeübt wird?	gering: 0x mittel: 0x hoch: 3x sehr hoch: 2x

Tabelle *3.6*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 3 (Fallstudie 2).

3.3 Konkretisierung der zentralen Problemstellung

Nachdem nun der praktische Bezug dieser Arbeit bekannt ist kann auf Basis der gewonnenen Erkenntnisse die in Abschnitt 1.2 allgemein formulierte zentrale Problemstellung mit Hilfe real-praktischer Anforderungen konkretisiert werden. Die nachfolgend aufgelisteten Kriterien (Tabelle 3.7) dienen zur Bewertung existierender Lösungsansätze sowie dem in dieser Arbeit vorgeschlagenen Modell.

Die Kriterienliste wird in Kapitel 4 zur Eignungsprüfung existierender Lösungsansätze verwendet und dient der abschließenden Bewertung des vorgeschlagenen Modells in Kapitel 11. Auf Grund der Unterschiedlichkeit der einzelnen Kriterien konnte keine einheitliche Bepunktungsskala verwendet werden. Vielmehr wird jedem Kriterium eine Ergebnismenge mit jeweils 3-4 charakterisierenden Attributen zugeordnet. Die Bewertung des Kriteriums 5 / Problemdimension 1 wird sowohl in Kapitel 4 als auch in Kapitel 11 auf die betrachteten Fallstudien bezogen.

#	Kriterium	Ergebnismenge
Komplexität der Infrastrukturarchitektur (Problemdimension 1)		
1	Auswirkung des Modells auf den Grad der Komplexität der Infrastrukturarchitektur	{senkend, ohne, steigernd}
2	Auswirkung des Modells auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betrifft	{senkend, ohne, steigernd}
3	Auswirkung des Modells auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten	{senkend, ohne, steigernd}
4	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur im Allgemeinen	{kein, wenig, ausreichend, groß}
5	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur in einem spezifischen Unternehmen	{kein, wenig, ausreichend, groß}
Wissenskomplexität und Wissensvolumen (Problemdimension 2)		
1	Unterstützung des Modells bei der Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	{kein, wenig, ausreichend, groß}
2	Beitrag des Modells, sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgemein zugänglich zu dokumentieren	{kein, wenig, ausreichend, groß}
3	Beitrag des Modells, die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass in mehr als 60% aller Fälle nachvollzogen werden kann weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	{kein, wenig, ausreichend, groß}
4	Beitrag des Modells zur Verwendung einer standardisierten Notation zur Dokumentation der Infrastrukturarchitektur	{kein, wenig, ausreichend, groß}
5	Beitrag des Modells zur Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	{kein, wenig, ausreichend, groß}

Finanzdienstleistungsinformatik (Problemdimension 3)		
1	Beitrag des Modells zur Anpassung an die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in einem Unternehmen auswirkt	{kein, wenig, ausreichend, groß}
2	Beitrag des Modells zur Anpassung an den Kostendruck, der auf ein Unternehmen ausgeübt wird	{kein, wenig, ausreichend, groß}

Tabelle *3.7*: Kriterienliste zur Bewertung von Lösungen der zentralen Problemstellung.

4 Betrachtung und Bewertung existierender Modellansätze

Ziel dieses Kapitels ist es, problemrelevante Verfahren zu erfassen und bezüglich ihrer Eignung zur Lösung der zentralen Problemstellung dieser Arbeit zu bewerten.[179]

4.1 Betrachtung der Modellansätze

4.1.1 ITIL

4.1.1.1 Allgemeiner Überblick

Grundlagen

Die sog. *IT Infrastructure Library* (ITIL)[180] ist ein weltweit akzeptierter De-facto-Standard für die Gestaltung, die Implementierung und das Management wesentlicher Steuerungsprozesse in der IT. ITIL ist eine Verfahrensbibliothek, die Praxiserprobte Lösungen für unterschiedliche Problemstellungen dokumentiert. Hauptaufgabe von ITIL ist es, eine vorwiegend Technologiezentrierte IT-Organisation Prozess-, Service- und Kundenorientiert auszurichten. Die von ITIL verwalteten Lösungen sind generisch formuliert und unabhängig von spezifischen Technologien bzw. einzelnen Anbietern. ITIL umfasst ca. 40 englischsprachige, öffentlich zugängliche Publikationen, die eine fachliche Dokumentation zur Planung, Erbringung und Unterstützung von IT-Dienstleistungen bereitstellen. Bis zum Jahr 2000 handelte es sich bei ITIL um eine lose Dokumentensammlung ohne ein gemeinsames Modell, in dem sich die einzelnen Dokumente hätten einordnen lassen. ITIL beschreibt die Architektur zur Etablierung und zum Betrieb von IT Service Management. Der Anspruch von ITIL ist es, dass die dokumentierten Empfehlungen auf alle möglichen IT-Dienstleistungen und Unternehmensstrukturen (von mittelständisch bis hin zu großen Unternehmen) anwendbar sind. In Form einer Prozessorientierten Sicht werden Aktivitäten be-

[179] Anmerkung: Die Auswahl der Modelle beruht auf den praktischen Erfahrungen des Autors.
[180] Siehe http://www.itil.co.uk

schrieben, die von Organisationseinheiten und Rollen mit Hilfe von Ressourcen ausgeführt werden. Nach Ansicht von KEMPER ET AL. existieren verschiedene Modelle und Methoden, die sich mit dem sog. *IT-Servicemanagement*[181] auseinander-setzen. Eine flächendeckende Etablierung hat in diesem Zusammenhang jedoch lediglich ITIL erreichen können.[182]

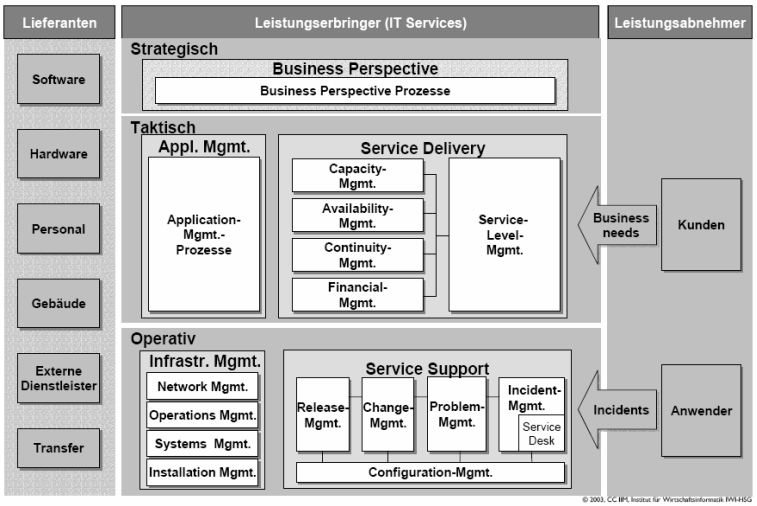


Abbildung 4.1*: ITIL-Framework (Quelle: Hochstein/Hunziker (2003)*[183]*).*

Ursprung

Der Ursprung von ITIL liegt in der zweiten Hälfte der 80er Jahre begründet, als die britischen Behörden die Effizienz des IT-Einsatzes grundsätzlich anzweifelten. Zu diesem Zeitpunkt wurde in Mangel an standardisierten Verfahren zur zweckmäßigen Etablierung und Abwicklung von IT-Dienstleistungen festgestellt. Die britische Regierung beauftragte daraufhin die *Central Computer and Telecommunications Agency* (CCTA) bewährte Verfahren zur Steuerung von Qualität, Sicherheit und Wirtschaftlichkeit der IT-Dienstleistungen zu sammeln, zu dokumentieren und zu vereinheitlichen. Im Jahr 2001 wurde die CCTA in das *Office of Government Commerce* (OGC) integriert, dessen Aufgabe die Unterstützung britischer Regierungsbehörden bei der Modernisierung der Einkaufsaktivitäten und IT-Dienstleistungen ist. Die Pflege

[181] Anmerkung: Der Begriff *IT-Servicemanagement* ist in Anlehnung an Kemper et al. (2004) eine Sammelbezeichnung für Konzepte zur Professionalisierung der IT-Bereiche. Die Aufgaben dieser Disziplin konzentrieren sich auf die Planung, Entwicklung, Überwachung und Steuerung des Lieferprozesses der Geschäftsprozess-unterstützenden Funktionen.
[182] Vgl. Hegering et al. (1999) sowie Kemper et al. (2004)
[183] Nach Hafner et al. (2004)

und Weiterentwicklung des ITIL-Standards wird von der OGC übernommen.

Philosophie und Einflüsse

ITIL basiert auf der Erkenntnis, dass die Anforderungen an die Akzeptanz, die Qualität, die Sicherheit und die Wirtschaftlichkeit von IT-Dienstleistungen nur dann erfüllbar sind, wenn diese IT-Dienstleistungen strukturiert organisiert und gelenkt werden. Prämisse dieser Feststellung ist, dass diejenigen Managementsysteme, die sich mit der Qualität und der Sicherheit der IT-Dienstleistungen auseinandersetzen, kombiniert werden müssen. Ihre Ausrichtung muss sich am Kunden orientieren. Der in ITIL enthaltene Begriff Infrastruktur umschreibt dabei die Gesamtheit der Anwendungen, IT-Systeme, Netzwerkkomponenten, Rechenzentren sowie die Haustechnik.

4.1.1.2 Module

ITIL ist auf Grund seiner generischen Struktur ein Framework. Die in ITIL dokumentierten Lösungsansätze sind daher als Richtungsweiser zu verstehen. Die konkrete Umsetzung der von ITIL vorgegebenen Prozesse variiert daher von Unternehmen zu Unternehmen.[184] Nach HAFNER ET AL. sind die wichtigsten Komponenten des ITIL-Frameworks *Service Delivery*, *Application Management*, *Service Support* bzw. *ICT Infrastructure Management*.[185]

Die oberste Strukturierungsebene des Frameworks bilden die sog. *Layer*. Insgesamt zählt das Framework drei Layer, denen unterschiedliche Module untergeordnet sind (Abbildung 4.1):[186]

- Der *strategische Layer* des ITIL-Frameworks ist weniger detailliert beschrieben als die anderen beiden. Er behandelt das Management der IT-Dienstleistungen. Die Module, die diesem Layer zugeordnet sind, thematisieren die Planung, das Controlling und das Marketing der IT-Dienstleistungen. Aufgaben der einzelnen Komponenten sind die Planung der IT-Infrastrukturarchitekturen, die Unterstützung der Software- und Hardwarelebenszyklen, das Qualitätsmanagement der

[184] Vgl. Kemper et al. (2004)
[185] Vgl. Hafner et al. (2004)
[186] Vgl. Kneer (2003), S. 79ff

IT-Dienstleistungen sowie die Aufrechterhaltung der Kunden- und Zuliefererbeziehungen.

- Der Fokus des *taktischen Layers* des ITIL-Frameworks liegt auf der Planung und dem Controlling der Prozesse. Die Durchführung dieser Aktivitäten soll die Voraussetzung für eine Kundenorientierte Dienstleistung schaffen.
- Die Module des *operativen Layers* unterstützen die Erbringung der IT-Dienstleistungen.

Auf diese drei Layer teilen sich die nachfolgend beschriebenen Module auf. Die publizierten Modulteile sind in Abbildung 4.2 dargestellt.

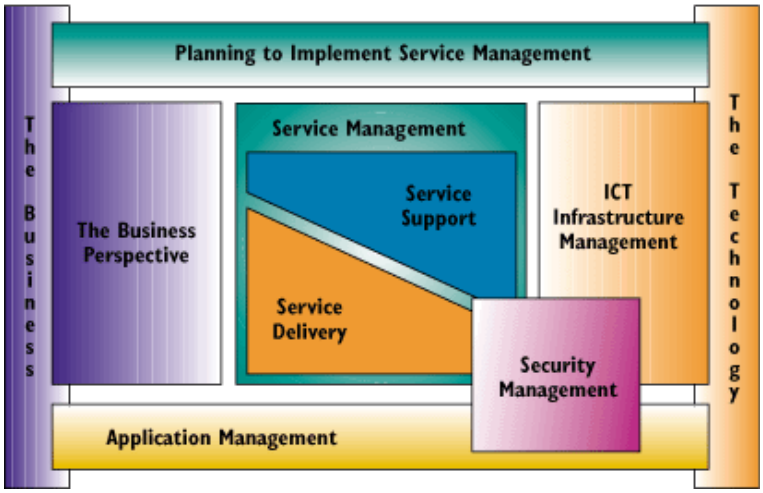


Abbildung 4.2: ITIL Publication Framework (Quelle: OGC (2002a)).

4.1.1.2.1 The Business Perspective

Das in OGC (2003) dokumentierte Modul *The Business Perspective* betrachtet die IT Prozesse aus Sicht der Unternehmensleitung: Das Modul umfasst die Kapitel *Business Continuity Management*, *Partnerships and Outsourcing*, *Surviving Change and Transformation of business practice through radical change* sowie *Understanding and Improving*. Die von der Unternehmensleitung geprägte Sicht auf die strategischen Aktivitäten eines IT-Dienstleisters dient insgesamt der Sicherstellung qualitativ hochwertiger IT-Dienstleistungen.

4.1.1.2.2 Planning to Implement Service Management

Das Modul *Planning to Implement Service Management* ist in OGC (2002c) beschrieben: Der in diesem Modul präsentierte Prozess *Conti-*

nuous Process Improvement befasst sich mit der Einführung und ständigen Verbesserung sämtlicher ITIL-Prozesse der jeweiligen Organisation. Welche Vorteile ein Unternehmen durch den Einsatz von ITIL hat, wird ebenfalls in diesem Modul beschrieben.

4.1.1.2.3 Service Support

Der in OGC (2000) dokumentierte *Service Support* fasst Prozesse zusammen, die den Kunden der IT-Dienstleistung unterstützen und informieren: Das sog. *Service Desk* bildet eine zentrale Anlaufstelle für alle Kunden. Hat der Kunde Probleme mit der IT-Dienstleistung oder identifiziert er Störungen in derselbigen, so kontaktiert er das Service Desk, um seine Probleme möglichst schnell lösen zu können. Um eine rasche Problembehebung kümmert sich das sog. *Incident Management*. Um nach der Behebung des Problems die Ursachen für selbiges identifizieren zu können, wird zusätzlich ein sog. *Problem Management* benötigt. Die Aktivitäten des Problem Managements können Auswirkungen auf die betroffenen IT-Systeme des Kunden haben. Beispielsweise kann eine neue Softwareversion eingespielt werden. Jegliche Änderungen an dem betroffenen System wird durch das sog. *Change Management* überwacht und koordiniert. Die letztendliche Freigabe dieser Änderungen übernimmt das sog. *Release Management*.

4.1.1.2.4 Service Delivery

Das Modul *Service Delivery* ist in OGC (2001) enthalten: Unter dem Begriff Service Delivery werden diejenigen Prozesse zusammengefasst, die zur Sicherstellung der angebotenen IT-Dienstleistung in Umfang, Qualität und Verfügbarkeit dienen. Wesentliche Aufgabe des sog. *Service Level Managements* ist es, das Angebot der IT-Dienstleistung exakt in Form der sog. *Service Level Agreements* (SLAs) zu definieren. Die betriebswirtschaftliche Realisierbarkeit der SLAs wird unter Mitarbeit des sog. *Financial Managements* überprüft. Die Verfügbarkeit[187] der Dienstleistung ist hierbei derart entscheidend, dass ein eigener

[187] Anmerkung: Die Verfügbarkeit beschreibt die Zeitspanne, in der eine Applikation für Nutzer theoretisch erreichbar ist. Internetapplikation werden häufig mit einer Verfügbarkeit von „24x7“ definiert, wobei der IT-Dienstleister gewährleisten muss, dass die Applikation die gesamte Woche (7 Tage pro Woche) durchgängig (24 Stunden je Tag) erreichbar sein muss.

Prozess namens *Availability Management* die Aufgabe hat, diese Verfügbarkeit zu erhöhen und die mit der Verfügbarkeit verbunden Ausfallzeiten zu senken. Die Sicherstellung der hierfür benötigten Ressourcen übernimmt das sog. *Capacity Management*. Damit im sog. *Katastrophenfall*[188] (K-Fall) ein Weiterbetrieb der Geschäftstätigkeit durch ein sog. *Desaster Recovery* garantiert ist, muss das sog. *Continuity Management* entsprechende Vorbereitungen treffen.

4.1.1.2.5 Applications Management

Das rein auf das Management von Softwareapplikationen ausgerichtete *Applications Management* wird in OGC (2002b) detailliert erläutert. Das *Applications Management* umfasst die beiden Teilprozesse *Software Lifecycle Support* und *Testing of IT Services*. Ersterer beider Prozesse beschäftigt sich mit der Planung der Entwicklung, des Testens und der Implementierung von Softwareapplikationen über den gesamten Systemlebenszyklus hinweg. Dem Testen der Applikationen widmet sich der eigene Prozess *Testing of IT Software*.

4.1.1.2.6 Security Management

Bereits 1999 wurde das Modul *Security Management* fertig gestellt:[189] *Security Management* besteht aus einem einzigen gleichnamigen Prozess, welcher das Einführen und Aufrechterhalten der in den SLAs vereinbarten Sicherheitsanforderungen steuert. Eine Aufgabe ist das Aufstellen eines Sicherheitsplans, in dem alle Maßnahmen aufgelistet sind, die die Datensicherheit einer Unternehmung garantieren. Um dieses Ziel zu erreichen, muss das *Security Management* regelmäßig die Sicherheit überprüfen und Schulungen im Sicherheitsbereich durchführen. Weiter müssen etwaige gesetzliche Rahmenbedingungen der Geschäftsprozesse bzgl. ihrer Einhaltung überprüft werden.

4.1.1.2.7 ICT Infrastructure Management

[188] Anmerkung: Der K-Fall beschreibt ein Worst-Case Szenario, bei dem ein unternehmenskritisches System ausfällt.
[189] Vgl. OGC (1999)

In OGC (2002a) wird das sog. *ICT*[190] *Infrastructure Management* (ICTIM) beschrieben. Diese Disziplin befasst sich mit den wesentlichen Aspekten, die mit der Infrastruktur von Informationssystemen zusammenhängen. Der Prozess überwacht die IT-Infrastruktur und beschäftigt sich mit der Einführung neuer Verfahren in diesem Umfeld. Weitere Kernthemen sind die Planung, der Betrieb und der Unterhalt der Rechenzentren sowie Netzwerke. In Anlehnung an KEMPER ET AL. konzentriert sich das ICTIM eher auf die operativ ausgerichteten Prozesse der Planung, Steuerung und des Betriebs technischer Infrastrukturkomponenten.[191] ICTIM gliedert sich in 4 Unterprozesse auf, die wie folgt definiert sind:

Definition: Design and Planning (D&P) Process

"The development and maintainance of ICT Design and Planning strategies and processes for the deployment and appropriate ICT infrastructure solutions throughout the organisation."[192]

Definition: Deployment Process

"The Deployment process is concerned with the implementation and rolling out of the business, and/or ICT, solution as designed and planned, with minimum disruption to the operations of the business processes."[193]

Definition: Operations Process

"The Operation process compromises all activities and measures necessary to enable and/or maintain in intended use of ICT services and infrastructure in order to meet Service Level Agreements and business targets."[194]

Definition: Technical Support Process

[190] ICT = Information and Communications Technology
[191] Vgl. Kemper et al. (2004)
[192] OGC (2002a), S. 21
[193] OGC (2002a), S. 75
[194] OGC (2002a), S. 115

> "The development of knowledge for the evaluation, support and proofing of all current and future ICT infrastructure solutions."[195]

Überblick

Motiviert wird das Modul dadurch, dass Wissen und Informationen als die zwei wichtigsten strategischen Ressourcen eines jeden Unternehmens identifiziert worden sind. Damit diese Ressourcen adäquat unterstützt werden können, bedarf es einer entsprechenden Investition in die Architektur, die Planung und das Management der diese Ressourcen unterstützenden Informations- und Kommunikationstechnologie. Charakteristisch für ICT ist, dass die unterstützten Systeme komplexer werden, und dass sich die Einführungszeiten neuer Systeme sukzessiv verkürzen. Der enorme Anspruch an das ICTIM erwächst zum einen aus der Fülle optionaler informationstechnischer Möglichkeiten und zum anderen aus dem Fehlen erprobter ICTIM-Standards. Der Umstand, dass sich die ICT-Komponenten über das Gesamte Unternehmen verteilen, erschwert das ICTIM. Hieraus ergeben sich folgende Aufgabengebiete des ICTIM:

- Ganzheitliche Betrachtung des Unternehmens, seiner Prozesse und seiner Tools[196].
- Sicherstellung einer stabilen Informations- und Kommunikationsinfrastruktur.
- Bildung der Ausgangslage für die ITIL-Module *Service Support* und *Service Delivery*.

Business Context

Das ITCIM hat ein effektives und gleichzeitig proaktives Management der Informations- und Kommunikationstechnologien zum Ziel: „[…] to reduce the overall Total Cost of Ownership (TCO) of ICT, while maintaining the overall quality of the ICT services provided."[197] Motivierende Faktoren hierfür sind:[198]

- *Abhängigkeit*: Organisationen werden zunehmend abhängiger von ICT.

---

[195] OGC (2002a), S. 165
[196] Anmerkung: Eine detaillierte Beschreibung, was das ICTIM unter dem Begriff „Tools" versteht, ist in OGC (2002a), Kapitel 2.7.2 beschrieben.
[197] OGC (2002a), S. 3
[198] Vgl. OGC (2002a), S. 2

- *Verlässlichkeit*: ICT hat sich zu einem Vertriebskanal für Produkte und Dienstleistungen einer Organisation entwickelt.
- *Komplexität*: Die Infrastrukturen im Bereich ICT werden zunehmend größer, verteilter und komplexer.
- *Flexibilität*: Wechselnde Kundenanforderungen erfordern neue Dienstleistungen. Das Dienstangebot muss jedoch mit ein- und derselben Infrastruktur offeriert werden.
- *Kundenzufriedenheit*: Kunden werden stets intoleranter. Fehlerhafte Dienstleistungen können einen gravierenden Einfluss auf das Geschäft haben.
- *Investitionen*: ICT hat bereits einen erheblichen Anteil an den Investitionen, die in einer Organisation getätigt werden.
- *Time-to-market*: Unternehmen sind Im Umfeld eines globalen Wettbewerbs mit kürzeren Lebenszyklen konfrontiert. Umso wichtiger ist es daher für ein Unternehmen Dienstleistungen und Produkte schneller als bisher auf den Markt zu bringen.

Um ICTIM erfolgreich in einem Unternehmen etablieren zu können bedarf es nach OGC (2002a) einer guten Beziehung zwischen dem CEO- und dem CIO-Bereich, durch die das sog. *Alignment* zwischen Geschäfts- und IT-Strategie sichergestellt werden kann. Die Ausrichtung der IT-Systeme nach den Bedürfnissen des Geschäfts ist formal eine gestalterische Aufgabe, die in Anlehnung an HAFNER ET AL. dem Architekturmanagement zugeschrieben wird.[199] Das Architekturmanagement wird von DERN auf mehrere Einzeldisziplinen aufgeteilt:[200] Die Gestaltung der sog. *IT-Basisinfrastruktur*, die die Menge aller Hardware- und aller systemnahen Softwarekomponenten der Informationssysteme bildet, ist Aufgabe des IT-Architekten. Die IT-Basisinfrastruktur erfasst demzufolge die vom ICTIM erfassten Hardware- und Softwarekomponenten. Zu diesen gehören u. a.:[201] zentrale Server[202], Lokationsser-

[199] Vgl. Hafner et al. (2004)
[200] Vgl. Dern (2003), S. 27
[201] Vgl. OGC (2002a), S. 48
[202] Anmerkung: Als zentrale Server werden die in einem Rechenzentrum aufgestellten Server bezeichnet.

ver[203], Mainframes, LANs, WANs, Datenbank Systeme sowie Storage Systeme.

Um beurteilen zu können, inwieweit ITIL eine Lösung für die zentrale Problemstellung dieser Arbeit darstellen kann, muss im Folgenden derjenige Teil des ITIL-Frameworks genauer betrachtet werden, der sich per Definition explizit mit der Infrastrukturarchitektur auseinandersetzt.

4.1.1.3 Infrastrukturarchitektur innerhalb des Prozesses *Design and Planning*

Das ICTIM befasst sich innerhalb von ITIL mit allen Aspekten, die mit der Infrastruktur von Informationssystemen zusammenhängen. ICTIM setzt selbst sich aus den vier zuvor in Abschnitt 4.1.1.2.7 definierten Prozessen *Design and Planning*, *Deployment*, *Operations* und *Technical Support* zusammen. In Anlehnung an die Definitionen dieser vier Prozesse sowie in Anlehnung an HAFNER ET AL. ist der Prozess *Design and Planning* (D&P) für die Ausgestaltung des Architekturmanagements im Bereich der Infrastruktur zuständig.[204] Als Synonym für den innerhalb des ICTIM verwendeten Begriffs *ICT Infrastructure* wird im Folgenden der Begriff *Infrastruktur* verwendet.

4.1.1.3.1 Überblick

Die Hauptaufgabe des D&P Prozesses ist die Gestaltung von Richtlinien für die Entwicklung und die Installation der ICT Infrastruktur, welche den aktuellen und zukünftigen Bedürfnissen der Geschäftsbereiche[205] entspricht. Bei der Ausführung dieses Prozesses sollten folgende Punkte sichergestellt werden:[206]

- Die Infrastruktur orientiert sich an den Bedürfnissen der Geschäftsbereiche.

---

[203] Anmerkung: Lokationsserver sind Server die in Lokationen außerhalb des Rechenzentrums (z. B. in Filialen) aufgestellt sind.
[204] Vgl. Hafner et al. (2004) bzw. Abschnitt 4.1.1.2.7
[205] Anmerkung: ITIL verwendet den Begriff des *Business*. Im Rahmen dieser Arbeit wird stattdessen der Begriff *Geschäftsbereich* verwendet.
[206] Vgl. OGC (2002a), S. 21

- Die Einflussfaktoren Innovation, Risiko und Kosten werden ausgewogen.
- Die Infrastrukturen des Kunden und der Partner sind aufeinander abgestimmt.
- Die Infrastrukturen orientieren sich an bekannten Frameworks und Standards.
- Zwischen den Planern der Infrastruktur und den Planern der Geschäftsbereiche existiert eine koordinierte Schnittstelle.
- Alle Prozesse, die sich mit dem Themengebiet ICT befassen, werden als strategisch betrachtet.

Hierbei werden folgende Ziele verfolgt:[207]

- Die Anforderungen der Geschäftsbereiche an die Infrastruktur werden erfüllt.
- Infrastrukturlösungen und Lösungen für die Geschäftsbereiche werden schneller eingeführt.
- Infrastrukturlösungen werden möglichst einfach und unter wirtschaftlichen Gesichtspunkten entworfen.
- Strategien werden durch ihre tägliche praktische Umsetzung „gelebt“.
- Alle Ressourcen der Infrastruktur werden optimal ausgenutzt.
- Die Qualität der Infrastruktur wird unter wirtschaftlichen Gesichtspunkten kontinuierlich verbessert.
- Langfristig auftretende Kosten müssen reduziert, minimiert oder limitiert werden.

Inputdaten für diesen Prozess sind beispielsweise existierende Strategien, Richtlinien oder auch Pläne. Insbesondere Strategien bzgl. der Ausgestaltung der existierenden Informationssysteme und der Infrastruktur müssen berücksichtigt werden.

4.1.1.3.2 Rollen

[207] Vgl. OGC (2002a), S. 23

Innerhalb des D&P Prozesses definiert ITIL zwei Schlüsselrollen:[208] den ICT Planer/Strategen (im Folgenden kurz Planer) und den ICT Designer/Architekten (im Folgenden kurz Architekt). Hauptaufgabe des Planers sind die Erstellung und die Koordination von Plänen und Strategien im ICT-Umfeld. Sein Ziel ist die Ausrichtung der Infrastruktur an den Bedürfnissen der Geschäftsbereiche. Der hierzu notwendige architektonische Entwurf der Infrastruktur wird im Rahmen des D&P Prozesses von dem Architekten übernommen. Konkret definiert das ICTIM 21 Aufgaben eines Architekten, die jedoch sehr unspezifisch formuliert sind. Drei Beispiele hierfür sind:[209]

- recommends proactive innovative ICT solutions for the improvement of ICT design and operation whenever and wherever possible
- documents all work using required standards, methods and tools
- works with ICT colleagues where appropriate, producing or updating ICT and corporate documentation and models

4.1.1.3.3 ICT Architekturen

Eine sog. *ICT Architektur* ist ein Entwurf der Infrastruktur auf lange Sicht, der sich an den Bedürfnissen der Geschäftsbereiche orientiert und der dabei hilft, Architekturentscheidungen kurzfristig fällen zu können. ITIL schreibt vor, dass Architekturen im ICT-Umfeld in den zwei Gebieten Technologie und Technologiemanagement entworfen werden müssen:[210] Eine exakte Definition *technologischer Architekturen* fehlt in ITIL. Vielmehr werden Beispiele genannt, in welchen ICT-Bereichen derartige Architekturen entworfen werden können. Beispielbereiche sind: Gesamtinfrastruktur, Server- und Unixwelten, Storage Systeme oder auch Netzwerke. Die Natur dieser Bereiche erfordert eine Hierarchisierung der einzelnen Architekturen. Im Gegensatz zu den technologischen Architekturen sind die sog. *Management Architekturen* des Gebiets Technologiemanagement weniger technologisch als eher organisatorisch. Die Aufgabe dieser Form der Architekturen ist die Schaffung einer Grundlage zur Zusammenarbeit der Planer und Architekten mit

[208] Vgl. OGC (2002a), S. 28
[209] Vgl. OGC (2002a), S. 66ff
[210] Vgl. OGC (2002a), S. 48ff

den Geschäftsbereichen. Somit ist eine Management Architektur des ICTIM eine Prozessarchitektur, die sich aus folgenden Komponenten zusammensetzt:[211]

- *Geschäftsbereiche*: Die Anforderungen, Bedürfnisse, Prozesse und Ziele der Geschäftsbereiche.
- *Menschen*: Die Aktivitäten der Manager und Angestellten, die an der Leistungserstellung beteiligt sind.
- *Prozesse*: Die Prozesse und (organisatorischen) Prozeduren, die benötigt werden, um ICT Dienstleistungen für Kunden und Geschäftsbereiche erbringen zu können.
- *Tools*: Alle Management-Werkzeuge, die für das ICTIM benötigt werden.
- *Technologie*: Die Infrastrukturtechnologie, die benötigt wird, um eine ICT Dienstleistung der richtigen Person am richtigen Platz zur richtigen Zeit zur Verfügung zu stellen.

4.1.2 CMMI

4.1.2.1 Allgemeiner Überblick

Definition

Das sog. *Capability Maturity Model Integration* (CMMI) ist ein Modell, das die Theorien des Prozessorientierten Software-Qualitätsmanagements (PSQM) praktisch umsetzt. Nach GLINZ ist das CMMI „[…] das älteste und wohl bekannteste Verfahren, das zur Verbesserung von Software-Prozessen eingesetzt wird.“[212] STELZER bezeichnet das CMM als einen Leitfaden, der die Grundgedanken des prozessorientierten Software-Qualitätsmanagements in idealtypischer Weise beschreibt.[213] In Anlehnung an DYMOND ist das CMMI „[…] ein lebendes Dokument, das sich, wie der Software-Prozess, den es behandelt, mit der Zeit fortentwickelt.“[214] Ferner ist das CMMI „[…] ein Modell, das beschreibt, wie sich Praktiken des Software-Engineerings

[211] Vgl. OGC (2002a), S. 49f
[212] Glinz (1999)
[213] Vgl. Stelzer (1998), S. 1f
[214] Dymond (2002), S. 6

in Organisationen unter bestimmten Bedingungen entwickeln."[215] Diese Bedingungen umfassen zum einen eine Organisation und die Betrachtung ihrer Arbeitsschritte als Prozess und zum anderen eine systematische Leitung der Prozess-Entwicklung.

Historie

Im Jahr 1986 beauftragte das amerikanische *Department of Defense* (DoD) das *Software Engineering Institute* (SEI) damit, eine neue Möglichkeit der Beurteilung (neben der niedrigsten Angebotspreis-Beurteilung) von Software-Lieferanten zu entwickeln. Der Grund für diesen Auftrag war die Tatsache, dass das DoD mehrere Millionen Dollar fehlinvestiert hatte, weil Software-Lieferanten ihre Produkte teilweise zu teuer, verspätet, fehlerhaft oder überhaupt nicht ausgeliefert hatten.[216] Das DoD hatte erkannt, dass es eine Methode benötigte, mit deren Hilfe es möglich war, die Fähigkeiten eines Software-Lieferanten einfach ermitteln zu können. Das Resultat der ersten Bemühungen seitens des SEI war das im September 1987 erstmals von Watt S. Humphrey vorstellte Dokument *A Method for Assessing the Software Engineering Capability of Contractors*. Hierbei handelt es sich um ein ca. 60 Seiten langes Dokument, das eine Liste von Fragen beinhaltet. Jede dieser Fragen gehört zu einem von fünf Reifegraden.

Nachdem 1991 das erste CMM in der Version 1.0 veröffentlicht wurde, und in den zwei Folgejahren diverse Erfahrungen bei dessen Einsatz gesammelt wurden, verabschiedete das SEI 1993 das CMM in der Version 1.1. Vier Jahre später, 1997, wurde an der Version 2.0 Draft C des CMM gearbeitet. In die Entwürfe der Version 2.0 sind wiederum Erfahrungen und Wünsche der CMM-Nutzer eingeflossen. Die Arbeit an der Version 2.0 wurde jedoch 1998 auf Drängen des DoD eingestellt. Grund hierfür war, dass in den Jahren zuvor viele andere Modelle unter dem Dach des CMM entwickelt worden sind. Hierzu zählten zum Beispiel das CMM für Systementwicklung (SE-CMM) oder das CMM für Personal (People CMM). Hinzu kam, dass eine Vielzahl von Unternehmen mehrere CMM parallel eingesetzt hatte und feststellen musste, dass dies zu enormen Schwierigkeiten in der Praxis geführt hatte.[217]

---

[215] Dymond (2002), S. 7
[216] Vgl. Kneuper (2003), S. 7, sowie May (1998), S. 9f
[217] Vgl. Kneuper (2003), S. 7f

Das DoD hat die Problematik des CMM auf Grund von Erfahrungsberichten erkannt und 1997 ein neues Projekt am SEI in Auftrag gegeben. Hauptziel dieses Projekts war die Schaffung eines neuen Modells (oder besser gesagt eines Frameworks), mit dessen Hilfe ein Unternehmen mehrere Sub-Modelle gleichzeitig und ohne die bis dato aufgetretenen Probleme einsetzen konnte. Aufgabe des SEI-Teams war also die Schaffung eines neuen Modells, dessen Intention die Integration mehrerer Modelle war. Als Quelldokumente dienten das CMM Version 2 (Entwurf C), der EIA/Interim Standard 731 (Entwurf 1.0), der das SE-CMM und das Systems Engineering Capability Model (SECM) in sich vereint, sowie das IPD-CMM Version 0.98, das sich mit der integrierten Produktentwicklung befasst. Nach der Veröffentlichung eines Pilotmodells mit der Versionsnummer 1.0 im Herbst 2000, wurde im August 2001 das CMMI für Software- und Systementwicklung sowie Integrierte Produktentwicklung mit dem Kürzel CMMI-SW/SE/IPPD und der Versionsnummer 1.1 veröffentlicht. Der Vorteil dieses Modells ist laut KNEUPER die auf die einzelnen Sub-Modelle (SW, SE und IPPD) abgestimmte Modellstruktur. In diesem Modell lassen sich – im Gegensatz zu den separaten Vorgängermodellen – übereinstimmende Aufgaben, Definitionen, Beschreibungen und Inhalte für die verschiedenen Disziplinen SW, SE und IPPD finden.[218]

Ziel

Durch die Unterstützung des Leitgedankens des PSQM, ist das Einsatzziel des CMMI die Erhöhung der Qualität von Software-Produkten bei gleichzeitiger Reduktion der Entwicklungskosten. Neben dem CMMI existiert eine Reihe von anderen Qualitätsmanagement-Modellen, die sich in Anlehnung an STELZER lediglich durch den Detaillierungsgrad und die Form der Darstellung von diesem unterscheiden. Die Grundgedanken, Ziele und Gestaltungsempfehlungen stimmen jedoch mit denen des CMMI überein.[219]

Eine allgemeine Abgrenzung des CMMI von ISO 900x, Bootstrap, ISO 15504 (SPICE) und anderen Modellen lässt sich bei KNEUPER finden.[220]

[218] Vgl. Kneuper (2003), S. 11f
[219] Vgl. Stelzer (1998), S. 2
[220] Vgl. Kneuper (2003), S. 3ff

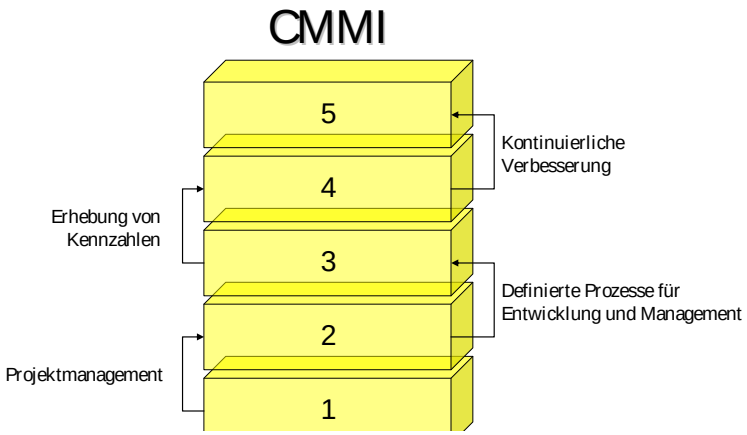


Abbildung 4.3: Die 5 Reifegrade des CMMI. (Quelle: Eigene Darstellung)

DYMOND differenziert zwischen CMMI und ISO 9001:[221] DYMOND führt den Unterschied beider Ansätze auf die Natur der beiden Modelle zurück. Er gibt an, dass ISO 9001 ein statischer Standard ist, dessen Wirkung sich dadurch entfaltet, dass sie einer Organisation ein Minimum an praktischen Richtlinien vorgibt, die diese zu befolgen hat. Das CMMI ist im Gegensatz dazu abstrakt und basiert auf Erfahrungswerten. Im CMMI wurden viele Praktiken der Softwareentwicklung gesammelt, um genau diejenigen Prozesse permanent zu verbessern, die für die Wartung und Weiterentwicklung von Software verwendet werden. Hierdurch erhält das CMMI im Gegensatz zu ISO 9001 einen dynamischen, progressiven Charakter. Dieser soll ein Unternehmen in die Lage versetzen, die momentan eingesetzten Software-Praktiken zyklisch und permanent zu optimieren.

Zentrale Bestandteile des CMMI sind die Begriffe *Reife*, *Fähigkeit* und *Prozess*. Diese werden im Folgenden genauer spezifiziert.

4.1.2.2 Reife

In der Bezeichnung des CMMI sind die beiden Wörter *Capability* (deutsch *Fähigkeit*) und *Maturity* (deutsch *Reife*) enthalten. Bezüglich des Reife-Begriffs wird zwischen reifen und unreifen Unternehmen unterschieden:[222] Ein unreifes Unternehmen zeichnet sich dadurch aus, dass seine Software-Prozesse stets auf improvisierten Praktiken der beteiligten Entwickler und des Managements beruhen. Selbst wenn ein

[221] Vgl. Dymond (2002), S. 7ff
[222] Vgl. Paulk et al. (1993), S. 2ff

Software-Prozess definiert ist, wird er in unreifen Unternehmen nicht verfolgt. Budgets werden partout überschritten, Zeitpläne werden nicht eingehalten. Dies basiert auf der Tatsache, dass Vorstellungen bezüglich des Budgets und der Zeit nicht auf realistischen Schätzungen beruhen. Als Folge dieser Fehleinschätzungen müssen Abstriche bei der Funktionalität und Qualität der Produkte gemacht werden, um Termine bzw. Budgets einhalten zu können. Im Gegensatz dazu zeichnet sich ein reifes Unternehmen durch ein organisationsweites Management der Softwareentwicklung aus. Weiterhin müssen existierende Prozesse definiert, eingehalten und permanent optimiert werden. Der Software-Prozess muss sowohl den alteingesessenen als auch den neu hinzugekommen Mitarbeitern exakt vermittelt werden. Neuerungen bezüglich der Software-Prozesse müssen durch Pilotversuche und Kosten/Nutzen-Analysen vor ihrem praktischen Einsatz überprüft werden. Darüber hinaus zeichnet sich ein reifes Unternehmen dadurch aus, dass die Qualität der Software-Produkte und die Zufriedenheit der Kunden analysiert werden. Hierzu muss es sowohl eine objektive als auch quantitative Basis für die Messbarkeit dieser beiden Faktoren geben. Erfahrungen mit Fehlschlägen sowie Aufwands- und Zeitschätzungen werden gesammelt, um diese Daten bei neuen Projekten zur Hilfe zu nehmen. Der *Reifegrad* (eng. *Maturity Level*) eines Unternehmens gibt demzufolge an, in welchem Ausmaß Software-Prozesse definiert, gesteuert, kontrolliert, optimiert und gemanagt werden.

Im CMMI wird der Reifegrad einer Organisation als eine Möglichkeit angesehen, die zukünftige Performance dieser Organisation bzgl. einer (oder mehrerer) gegebenen Disziplin(en) vorherzusagen.[223] Weiterhin heißt es, dass eine Organisation mit jedem Reifegrad ihre Prozesse um einen erheblichen Teil stabilisiert. Das CMMI kategorisiert Unternehmen mit Hilfe von 5 Reifegraden. Die Verfasser des CMMI gehen davon aus, dass die Produktqualität mit zunehmendem Reifegrad steigt, und dass die Entwicklungsdauer und die mit der Entwicklung verbundenen Kosten sinken.[224]

[223] Vgl. SEI (2002), S. 10ff
[224] Vgl. Stelzer (1998), S. 202; vgl. auch Paulk et al. (1993), S. 19ff

Das CMMI verbindet jeden der 5 Reifegrade (bis auf den ersten) mit einer Anzahl von Tätigkeiten, die zur Erreichung von festgelegten Zielen führen. Der erste Reifegrad ist ein Standard-Reifegrad, den jedes Unternehmen automatisch besitzt. Erst wenn die vorgeschriebenen Ziele des zweiten Reifegrades erfolgreich erreicht wurden, hat das Unternehmen den zweiten Reifegrad erreicht. Wie in Abbildung 4.3 dargestellt, werden die 5 Reifegrade des CMMI stufenförmig dargestellt. Ein Unternehmen muss diese imaginäre Treppe (bis zur fünften Stufe) erklimmen, um den höchsten Grad der prozessorientierten Organisationen zu erreichen. Mit jedem Reifegrad-Wechsel ist auch eine thematische Fokussierung der zu erreichenden Ziele verbunden.[225] Wie aus Abbildung 4.3 hervorgeht, liegt der Fokus beim Wechsel von dem als „chaotisch“ oder auch als „ad hoc“ charakterisierten Reifegrad 1 hin zu Reifegrad 2 auf dem Projektmanagement. Nach KNEUPER ist das Ziel des 2. Reifegrads, „[…] die wesentlichen Managementprozesse zu etablieren, um Kosten, Zeitpläne und Funktionalitäten von Projekten zu planen und zu steuern.“[226] Dies deutet darauf hin, dass sich der Horizont des 2. Reifegrads primär auf der Projektebene bewegt. Ein wesentliches Ziel des CMMI ist allerdings die Institutionalisierung von Prozessen.[227] Daher erweitert sich der Horizont des Prozessmanagements beim 3. Reifegrad auf die Organisation als Ganzes. DYMOND charakterisiert das Erreichen des 3. Reifegrades des CMMI damit, dass „[…] die besten Praktiken der Projekte bereits verallgemeinert [wurden], damit sie in der gesamten Organisation angewendet werden können.“[228] Um den 4. Reifegrad des CMMI zu erreichen, müssen Metriken und Kennzahlen intensiv genutzt werden. KNEUPER begründet dies damit, dass die hiermit verbundenen Daten als Entscheidungshilfen für Prozessverbesserungen genutzt werden können.[229] Nach dem Erreichen des 5. und letzten Reifegrades konzentriert sich das CMMI eine kontinuierliche Verbesserung der Prozesse. DYMOND formuliert die Optimierungsaufgaben der Mitarbeiter bei Erreichen des 5. Reifegrads wie folgt: „Sie

[225] Vgl. Kneuper (2003), S. 15ff
[226] Kneuper (2003), S. 15
[227] Vgl. CMMI (2002), S. 36
[228] Dymond (2002), S. 97
[229] Vgl. Kneuper (2003), S. 16

konzentrieren sich darauf, den Software-Prozess entsprechend der Geschäftsstrategien zu verändern: für Wettbewerbsvorteile durch höhere Qualität und Produktivität sowie ein schnelleres Time-to-Market.“[230]

Reifegrad	Prozessgebiet
2	Anforderungsmanagement (Requirements Management) Projektplanung (Project Planning) Projektverfolgung und –kontrolle (Project Monitoring and Control) Management von Lieferantenvereinbarungen (Supplier Agreement Management) Messung und Analyse (Measurement and Analysis) Qualitätssicherung von Prozessen (Process and Product Qualilty Assurance) Konfigurationsmanagement (Configuration Management)
3	Anforderungsentwicklung (Requirements Development) Technische Unterstützung (Technical Solution) Produktintegration (Product Integration) Validation (Validation) Verifikation (Verification) Organisationsweiter Prozessfokus (Organizational Process Focus) Organisationsweite Prozessdefinition (Organizational Process Definition) Organisationsweites Training (Organizational Training) Integriertes Projektmanagement (Integrated Project Management) Risikomanagement (Risk Management) Entscheidungsanalyse und –findung (Decision Analysis and Resolution)
4	Quantitatives Projektmanagement (Quantitative Project Management) Performanz organisationsweiter Prozesse (Organizational Process Performance)
5	Ursachenanalyse und Problemlösung (Causal Analysis and Resolution) Organisationsweite Innovation und Verbreitung (Organizational Innovation and Deployment)

Tabelle 4.1: Die einzelnen Prozessgebiete des CMMI 1.1 im Überblick.

4.1.2.3 Fähigkeit

Im CMM ist der Begriff *Fähigkeit* mit an den Software-Prozess gebunden:[231] Demzufolge beschreibt die Fähigkeit eines Software-Prozesses, innerhalb welchen Intervalls sich die zu erwartenden Ergebnisse befinden. Das CMMI hingegen definiert explizit den Begriff des Fähigkeitsgrads (engl. *Capability Level*). Einem Fähigkeitsgrad sind spezifische

[230] Dymond (2002), S. 180
[231] Vgl. Paulk et al. (1993), S. 3

und generische Praktiken zugeordnet:[232] Sofern diese Praktiken dazu führen, dass das mit ihnen verbundene Ziel erreicht wird, gilt auch der zugehörige Fähigkeitsgrad als erreicht. Das CMMI definiert insgesamt 6 Fähigkeitsgrade.

4.1.2.4 Prozess

Analog zum PSQM definiert das CMMI einen Prozess als eine Abfolge von Schritten, deren Durchführung notwendig ist, um ein festgelegtes Ziel zu erreichen.[233] Explizit definieren die Verfasser des CMMI einen Prozess als ein „[…] Zusammenspiel von Aktivitäten, Methoden und Veränderungen, die Menschen durchführen, um Software und die damit verbundenen Produkte zu entwickeln und zu warten.“[234] Beispiele für die vom CMMI als Produkte bezeichneten Artefakte sind Projektpläne oder auch Programmcodes.

Jedes Entwicklungsvorhaben benötigt die drei untereinander zusammenhängen Komponenten Personal, Prozess und Technologie, die von PHILLIPS als die Hauptdeterminanten der Entwicklungskosten, der Zeitpläne sowie der Produktqualität bezeichnet werden.[235] Nach Ansicht von DYMOND sind zwar alle drei Komponenten an einem Softwareentwicklungsprozess beteiligt, die Probleme aber, die für gewöhnlich bei Software-Produkten auftauchen (z. B. eine zu späte Auslieferung, auftretende Fehler etc.), sind aber meist prozessbedingt. Nahe liegend ist es daher, die Ursachen dieser Probleme in der Komponente *Prozess* zu beseitigen.[236] PHILLIPS führt diesen Gedanken fort und stellt fest, dass es nicht ausreicht über sehr gute und ausreichend motivierte Mitarbeiter zu verfügen, solange diese Mitarbeiter auf Grund eines unverstandenen oder schlecht ausgearbeiteten Prozesses ihr Bestes geben können.[237]

Die nachfolgend abgedruckte Tabelle 4.1 listet die einzelnen Prozessgebiete des CMMI in der Version 1.1 auf.

---

[232] Vgl. Kneuper (2003), S. 21ff
[233] Vgl. Dymond (2002), S. 7ff
[234] Dymond (2002), S. 9
[235] Vgl. Phillips (2002)
[236] Vgl. Dymond (2002), S. 10
[237] Vgl. Phillips (2002)

4.1.3 ISO 15504

Die Norm ISO 15504, die unter dem Namen *Software Process Improvement and Capability dEtermination* (SPICE) bekannt ist, gehört wie das CMMI zu den Qualitätsmanagementmodellen. Nach KNEUPER soll es einen einheitlichen Rahmen für verschiedene Modelle (wie zum Beispiel das CMMI) geben. SPICE ist kompatibel mit dem CMMI. Kernpunkte von SPICE sind zum einen die Bestimmung des Prozessreifegrads (engl. *Capability Determination*) und zum anderen die Verbesserung der Prozesse an sich (engl. *Process Improvement*). Vergleichbar mit dem CMMI können mit Hilfe sog. *Assessments* Unternehmen bzgl. ihrer Prozessreife bewertet werden.[238]

SPICE besteht aus insgesamt fünf Komponenten:[239] *Vocabulary and Concepts* (ISO 15504-1), *Performing an Assessment* (ISO 15504-2), *Guidance on Performing an Assessment* (ISO 15504-3), *Guiding on Using Assessment Results* (ISO 15504-4), *An Exemplar Process Assessment Model* (ISO 15504-5). Diese einzelnen Komponenten führen in die Grundbegriffe ein, die für ein Verständnis von SPICE relevant sind. Zielführende Faktoren des Modells sind die Prozessverbesserung und die Erhöhung der Fähigkeit. Zur Durchführung eines Assessments im Sinne der Norm werden zunächst die Aktivitäten eines derartigen Vorhabens definiert. Hierzu zählen neben der Planung, die Datensammlung, die Prüfung der Daten, die Prozessmessung sowie die Berichterstellung über die Messergebnisse. Weiterhin definieren diese Komponenten die minimalen Anforderungen an ein Assessment. Wie im Fall des CMMI legt SPICE fest, was erreicht werden muss, um eine spezifische Prozessreife zu erreichen. SPICE beschreibt jedoch nicht, wie eine Prozessreife erreicht werden kann.

[238] Vgl. Kneuper (2003), S. 4
[239] Vgl. SPICE (2004)

4.1.4 Management der Informations- und Kommunikationstechnik (IKT)

Die Aufgabe des IKT-Managements ist die Steuerung und Kontrolle der Planung sowie effiziente und effektive Implementierung, Nutzung und Weiterentwicklung der IKT als Infrastruktur.[240] Im Gegensatz zu HEINRICH wird IKT-Management bei KRCMAR nicht nur als Teil des strategischen Managements aufgefasst, sondern auch auf operativer Ebene behandelt.[241]

In Anlehnung an KRCMAR können folgende Aufgaben des IKT-Managements identifiziert werden:[242]

- *Wartung und Betrieb der IKT*: Wartung und Betrieb der IKT werden als operatives IKT-Management bezeichnet. Teilaufgaben sind die Wartung und Reparatur von Hard- und Software, Helpdesk-Aktivitäten, Netzwerkdienste sowie die Durchführung von Schulungen. Zudem gehören Disziplinen wie etwa das Lizenzmanagement oder die Bestandsfürhung zu diesem Aufgabengebiet.
- *Strategisches Management der IKT*: Zentrales Aufgabengebiet des strategischen IKT-Managements ist die Erfassung zukünftiger, für das Unternehmen relevanter Techniktrends. Methoden zur Bewältigung dieser Aufgabe sind die Bestimmung des optimalen Einsatzzeitpunktes einer Anwendung, die Standardauswahl sowie das Technology-Roadmapping.
- *Management der Aneignung von IKT*: Unter Aneignung der IKT versteht KRCMAR das rudimentäre Verstehen der Technlogie. Nur wenn die Technologie verstanden wird können Rückschlüsse auf ihre Eignung in Bezug auf den Einsatz im Unternehmen gezogen werden.

4.1.5 Zachman Framework

Das Framework für unternehmensweite IT-Architekturen wurde 1987 erstmals veröffentlicht:[243] ZACHMAN kreiert mit diesem Framework eine

[240] Vgl. Krcmar (2005), S. 210ff
[241] Vgl. Krcmar (2005), S. 216, bzw. Heinrich (2002)
[242] Vgl. Krcmar (2005), S. 217ff

Analogie zwischen dem Bau von Häusern und dem von Informationssystemen. Es besteht aus einer Menge von Bausteinen die zusammengestzt die Grundlage zum Bau einer sog. *Enterprise Architektur* ergeben. Jeder Baustein ist in einer 6x6 Felder großen Matrix angeordnet, die die Zugehörigkeit des Bausteins zum Verstehen der Gesamtarchitektur in X- und Y-Richtung der Matrix definiert.

Jedem Quadranten sind Artefakte zugeordnet. Ein Artefakt ist ein Element (z. B. Anforderungsdefinition, Handbücher, Programmcodes etc.), das zum Verstehen der Gesamtarchitektur benötigt wird. In horizontaler Richtung ist der sog. *Fokus* eines Artefakts ausgerichtet. Er besteht aus den Fragen nach dem *What?*, *How?*, *Where?*, *Who?*, *When?* und *Why?*. In vertikaler Richtung sind die Perspektiven auf die Unternehmensarchitektur angeordnet.

Im Detail definiert das Zachman Framework in horizontaler Richtung den folgenden Fokus:

- *Data/What?*: Wichtiges für das unternehmerische Handeln wie bspw. alle Hauptwörter, die selbiges beschreiben (z. B. „Konto“)
- *Function/How?*: Alle unternehmerischen Aktivitäten wie bspw. alle Verben, die die Aktivitäten beschreiben (z. B. „verbuchen“)
- *Network/Where?*: Alle Orte an denen unternehmerisches Handeln praktiziert wird.
- *People/Who?*: Alle Personen, die zum unternehmerischen Handeln benötigt werden.
- *Time/When?*: Ein Zeitplan, aus dem hervorgeht, wann unternehmerische Aktivitäten ausgeführt werden.
- *Motivation/Why?*: Alle Gründe, weshalb Aktivitäten ausgeführt werden.
- Detailperspektiven sind:
- *Planner/Scope*: Die in dieser Zeile ausgeführten Handlungen sind primär strategischer Natur.

---

[243] Vgl. Zachman (1992)

- *Owner/Enterprise*: Alle Aktivitäten, die relevant für das Unternehmen sind. Primär handelt es sich um Geschäftsprozessmodellierung.
- *Designer/System*: Auf konzeptioneller Ebene wird eine Applikationsarchitektur erzeugt.
- *Builder/Technology*: Die konzeptionelle Ebene wird in dieser Zeile durch physische Architekturbeschreibungen konkretisiert.
- *Subcontractor/Components*: Sämtliche Spezifikationen, die zum Codebau benötigt werden.
- *Functioning System*: Das pyhsisch vorhandene System.

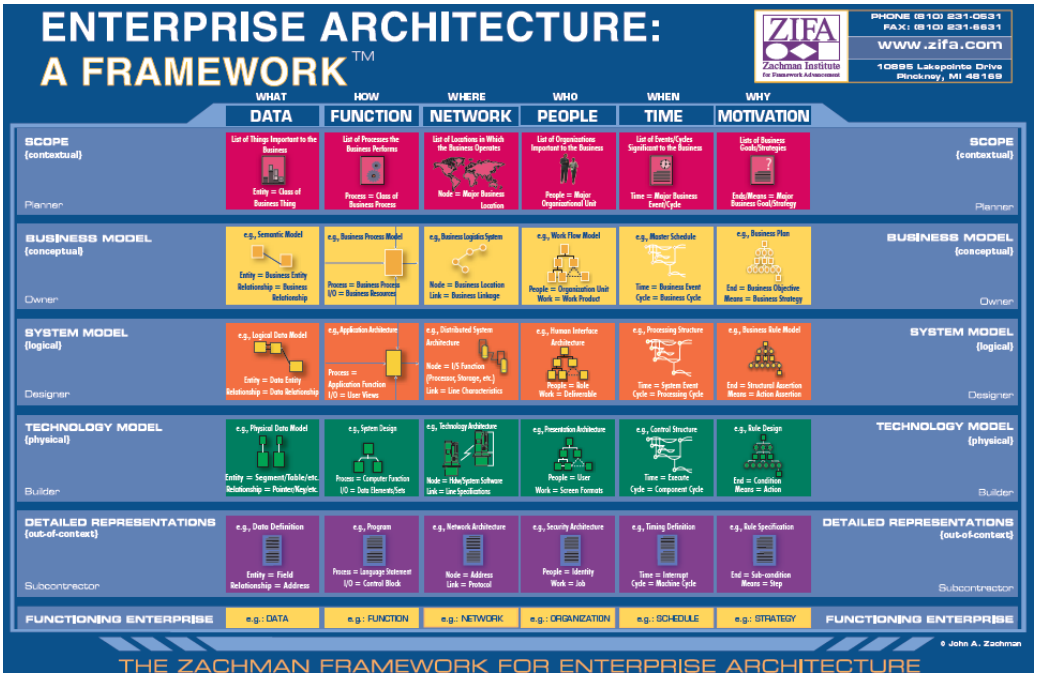


Abbildung 4.4: Zachman Framework for Enterprise Architecture (Quelle: www.zifa.com, Abruf am 01.03.2007).

4.2 Bewertung der Modellansätze

Auf Basis der in den letzten Abschnitten gewonnenen Erkenntnisse werden im Folgenden die vorgestellten Modellansätze bzgl. ihrer Eignung zur Lösung der zentralen Problemstellung dieser Arbeit analysiert und bewertet. Die Bewertung ist subjektiv.

4.2.1 ITIL

Im Allgemeinen

Anhand der Erkenntnisse aus Abschnitt 4.1 lassen sich folgende allgemeine Charakteristika von ITIL feststellen:

- ITIL ist ein Standard für die Gestaltung, die Implementierung und das Management von Steuerungsprozessen in der IT.
- Die wesentliche Aufgabe von ITIL ist es, eine vorwiegend technologiezentrierte IT-Organisation Prozess-, Service- und Kundenorientiert auszurichten.
- Die von ITIL verwalteten Lösungen sind generisch formuliert und unabhängig von spezifischen Technologien bzw. einzelnen Anbietern.
- ITIL beschreibt die Architektur zur Etablierung und zum Betrieb von IT Service Management.
- Der Anspruch von ITIL ist es, dass die dokumentierten Empfehlungen auf alle möglichen IT-Dienstleistungen und Unternehmensstrukturen anwendbar sind.
- ITIL ist eine Disziplin des IT-Servicemanagements.

Speziell für den Bereich der Infrastruktur beinhaltet ITIL das Modul ICTIM. ICTIM befasst sich mit den wesentlichen Aspekten, die mit der Infrastruktur von Informationssystemen zusammenhängen: Der Prozess konzentriert sich auf die operativ ausgerichteten Prozesse der Planung, Steuerung und des Betriebs technischer Infrastrukturkomponenten.[244] Planungsinstrument ist die ICT Architektur, die einen Entwurf der In-

[244] Vgl. Kemper et al. (2004)

frastruktur auf lange Sicht darstellt, und sich an den Bedürfnissen der Geschäftsbereiche orientiert. Die ICT Architektur soll helfen, Architekturentscheidungen kurzfristig fällen zu können. Eine exakte Definition *technologischer Architekturen* fehlt in ITIL. Es werden lediglich Beispiele genannt, in welchen ICT-Bereichen derartige Architekturen entworfen werden können: Gesamtinfrastruktur, Server- und Unixwelten, Storage Systeme oder auch Netzwerke.

Im Speziellen

In Bezug auf die drei Problemdimensionen lässt sich folgende Eignung von ITIL zur Lösung der zentralen Problemstellung dieser Arbeit feststellen:

- Die *Komplexität der Infrastrukturarchitektur* wird lediglich als existent erwähnt. Der hierfür zuständige Prozess D&P übernimmt jedoch die Aufgabe der Gestaltung von Richtlinien für die Entwicklung und die Installation der ICT Infrastruktur. Der Prozess ist im Wesentlichen auf die organisatorische Verwaltung der Handlungen und Handelnden im Umfeld einer Infrastruktur ausgerichtet. Grundlegend werden Rollen, Voraussetzungen und Ziele definiert („Was?“). Eine detaillierte Vorgabe konkreter Handlungsschritte („Wie?“) ist jedoch nicht Teil des Moduls.
- Eine explizite Behandlung der Problemdimension *Wissenskomplexität und Wissensvolumen* ist kein Bestandteil von ITIL.
- Auf Grund des Anspruchs von ITIL, dass die Inhalte auf alle möglichen IT-Dienstleistungen und Unternehmensstrukturen anwendbar sind, fehlt eine explizite Berücksichtigung der Problemdimension *Historische Entwicklung der Finanzdienstleistungsinformatik*.

Zusammenfassend lässt sich feststellen, dass der Fokus von ITIL auf der Prozessausgestaltung einer IT-Organisation liegt. ITIL (einschließlich aller Module) ist als Framework ausgelegt. Somit gibt ITIL einen organisatorischen Rahmen vor, der im Groben Aktivitäten und Rollen generisch definiert. ITIL beschreibt die Input- und Outputdaten eines Prozesses sowie die Akteure, die für die Erstellung und den Empfang dieser Daten verantwortlich sind. Eine konkrete Auseinandersetzung mit der zentralen Problemstellung dieser Arbeit, die nach einer spezifi-

schen Lösung der Problemdimensionen verlangt, kann ITIL jedoch in Gänze nicht entnommen werden.

Die Tabelle 4.2 durchgeführte Bewertung erfolgt auf Grund der reinen Analyse des in OGC (2002a) abgedruckten Inhalts. Die individuell mögliche Umsetzung des Frameworks in einem Unternehmen wird nicht in die Bewertung mit aufgenommen. Das überwiegend negative Bewertungsergebnis in Bezug auf die spezifische Problemsituation kann auf die von KEMPER ET AL. durchgeführte Charakterisierung zurückgeführt werden: „ITIL versteht sich als Framework. Somit ist es nicht das Ziel von ITIL, in Form einer exakten Rezeptur sämtliche Details [..] vorzugeben.“[245]

4.2.2 CMMI & ISO 15504

CMMI und ISO 15504 (SPICE) sind beides Modelle, die die Theorien des PSQM praktisch umsetzen. Beide Modelle streben die Verbesserung von Softwareprozessen an. KNEUPER nennt als Motivation für das CMMI: „Bei der Entwicklung von Software […] hat fast jede Organisation Schwierigkeiten, in der vorgesehenen Zeit, im Budget und mit der zugesagten Qualität fertig zu werden. […] Für die Auftraggeber führt das dazu, dass sie die Software nicht wie bestellt bekommen und dadurch typischerweise Zusatzkosten haben […]“[246]

Ferner heißt es bei KNEUPER: „Derartige Probleme bei der Vergabe hochkomplexer Software im Rahmen des SDI-Projektes […] haben das amerikanische Verteidigungsministerium […] um 1986 veranlasst, einen neuen Lösungsansatz dafür zu suchen, um bei der Vergabe einigermaßen sicher zu sein, dass der Auftragnehmer wie versprochen liefern kann. Aus dieser Arbeit entstand 1991 das […] CMM.“[247]

Diese Aussagen zeigen, dass es die Motivation des CMMI ist, die Reife und Fähigkeit eines Unternehmens zu verbessern. Die technische Infrastruktur, die Softwaresysteme benötigen, wird nicht explizit thematisiert. Auch SPICE thematisiert die technische Infrastruktur von Soft-

[245] Kemper et al. (2004)
[246] Kneuper (2003), S. 1
[247] Kneuper (2003), S. 1

waresystemen nicht.[248] Somit fehlt auch in diesen beiden Modellen die konkrete Auseinandersetzung mit der zentralen Problemstellung dieser Arbeit. Die Bewertung ist in *Tabelle* 4.3 abgedruckt.

4.2.3 Management der Informations- und Kommunikationstechnik (IKT)

Zentrale Aufgabe des IKT-Managements nach KRCMAR ist die Steuerung und Kontrolle der Planung sowie effiziente und effektive Implementierung, Nutzung und Weiterentwicklung der IKT als Infrastruktur.[249] Teilaufgaben sind Wartung und Betrieb der IKT, Strategisches Management der IKT sowie Management der Aneignung von IKT. Es handelt sich hierbei nicht um ein konkretes Modell, mit dem die in dieser Arbeit adressierte Problemstellung gelöst werden kann. Ein Grund hierfür ist, dass KRCMAR mit jeder Teilaufgabe thematische Richtungen vorgibt und lediglich einzelne Teilaspekte der Vielfältigkeit der Gesamtproblematik anspricht. Jeder der angesprochenen Teilaspekte wird mit einem Verweis auf entsprechende wissenschaftliche Arbeiten versehen. Hinzu kommt, dass die Komplexität der Infrasturkturarchitektur nicht explizit behandelt wird. Vielmehr wird die Komplexität in Software-Standards implizit erwähnt.

Eine Vernachlässigung der Infrastrukturarchitektur kann auch bei HEINRICH beobachtet werden:[250] Der physische Teil eines Informationssystems wird als *Technologie-Architektur* bezeichnet, die die Umsetzung eines logischen Architekturmodells mittels IKT ist. Eine Unterteilung findet lediglich grob in Hardware-, Software- und Netzwerkarchitektur statt.

Ein konkretes Modell, das dabei hilft, die komplexitätsbedingten Entscheidungsprobleme der Infrastrukturarchitektur eines Finanzdienstleisters bewältigen zu können, ist in beiden Ansätzen nicht auszumachen. Dennoch liefern die beiden Ansätze konkrete Leitlinien, die für den

---

[248] Anmerkung: SPICE und CMMI verwenden den Begriff Infrastruktur lediglich in Zusammenhang mit der organisatorischen Ausgestaltung einer sog. *Prozessinfrastruktur*.
[249] Vgl. Krcmar (2005), S. 210ff
[250] Vgl. Heinrich (2000)

Entwurf eines eigenen Models richtungsgebend sind. Die Bewertung ist in *Tabelle* 4.4 abgedruckt.

4.2.4 Zachman Framework

Das Zachman Framwork ist ein Rahmengerüst, das richtungsweisend die zur Schaffung einer unternehmensweiten IT-Architektur zu berücksichtigenden Bausteine definiert. Das Framework ist vorwiegend generisch gehalten. Vorgaben, welche konkreten Ausprägungen ein einzelner Baustein der 6x6 Matrix haben soll (d. h. welche Notationsform beispielsweise die Beschreibung der Infrastrukturarchitektur eines Informationssystems hat), wird nicht vorgeschrieben. Die Konkretisierung des Frameworks wird im Sinne eines Rahmengerüsts anderen Modellen überlassen.
Zudem wird die Komplexität der Infrastrukturarchitektur im Speziellen nicht erwähnt bzw. behandelt. Infrastrukturarchitektur ist ein abstrakter Baustein. Daher stellt das Zachman Framework ebenfalls keinen konkreten Lösungsansatz dar, mit dem komplexitätsbedingte Entscheidungsprobleme der Infrastrukturarchitektur bewältigt werden können. Die Bewertung ist in *Tabelle* 4.5 abgedruckt.

#	Kriterium	Ergebnis-menge
Komplexität der Infrastrukturarchitektur (Problemdimension 1)		
1	Auswirkung des Modells auf den Grad der Komplexität der Infrastrukturarchitektur	Ohne
2	Auswirkung des Modells auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betrifft	Ohne
3	Auswirkung des Modells auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten	Ohne
4	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur im Allgemeinen	wenig
5	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur in einem spezifischen Unternehmen	wenig
Wissenskomplexität und Wissensvolumen (Problemdimension 2)		
1	Unterstützung des Modells bei der Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	wenig
2	Beitrag des Modells, sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgeimein zugänglich zu dokumentieren	wenig
3	Beitrag des Modells, die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass in mehr als 60% aller Fälle nachvollzogen werden kann weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	wenig
4	Beitrag des Modells zur Verwendung einer standardisierten Notation zur Dokumentation der Infrastrukturarchitektur	wenig
5	Beitrag des Modells zur Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	wenig
Historische Entwicklung der Finanzdienstleistungsinformatik (Problemdimension 3)		
1	Beitrag des Modells zur Anpassung an die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in einem Unternehmen auswirkt	wenig
2	Beitrag des Modells zur Anpassung an den Kostendruck, der auf ein Unternehmen ausgeübt wird	wenig

Tabelle 4.2: Eignung von ITIL zur Lösung der zentralen Problemstellung.

#	Kriterium	Ergebnis-menge
Komplexität der Infrastrukturarchitektur (Problemdimension 1)		
1	Auswirkung des Modells auf den Grad der Komplexität der Infrastrukturarchitektur	ohne
2	Auswirkung des Modells auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betrifft	ohne
3	Auswirkung des Modells auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten	ohne
4	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur im Allgemeinen	kein
5	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur in einem spezifischen Unternehmen	kein
Wissenskomplexität und Wissensvolumen (Problemdimension 2)		
1	Unterstützung des Modells bei der Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	wenig
2	Beitrag des Modells, sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgeimein zugänglich zu dokumentieren	wenig
3	Beitrag des Modells, die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass in mehr als 60% aller Fälle nachvollzogen werden kann weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	iein
4	Beitrag des Modells zur Verwendung einer standardisierten Notation zur Dokumentation der Infrastrukturarchitektur	kein
5	Beitrag des Modells zur Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	kein
Historische Entwicklung der Finanzdienstleistungsinformatik (Problemdimension 3)		
1	Beitrag des Modells zur Anpassung an die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in einem Unternehmen auswirkt	kein
2	Beitrag des Modells zur Anpassung an den Kostendruck, der auf ein Unternehmen ausgeübt wird	kein

Tabelle *4.3*: Eignung von CMMI und SPICE zur Lösung der zentralen Problemstellung.

#	Kriterium	Ergebnis-menge
Komplexität der Infrastrukturarchitektur (Problemdimension 1)		
1	Auswirkung des Modells auf den Grad der Komplexität der Infrastrukturarchitektur	ohne
2	Auswirkung des Modells auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betrifft	ohne
3	Auswirkung des Modells auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten	ohne
4	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur im Allgemeinen	wenig
5	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur in einem spezifischen Unternehmen	wenig
Wissenskomplexität und Wissensvolumen (Problemdimension 2)		
1	Unterstützung des Modells bei der Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	kein
2	Beitrag des Modells, sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgeimein zugänglich zu dokumentieren	kein
3	Beitrag des Modells, die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass in mehr als 60% aller Fälle nachvollzogen werden kann weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	kein
4	Beitrag des Modells zur Verwendung einer standardisierten Notation zur Dokumentation der Infrastrukturarchitektur	kein
5	Beitrag des Modells zur Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	kein
Historische Entwicklung der Finanzdienstleistungsinformatik (Problemdimension 3)		
1	Beitrag des Modells zur Anpassung an die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in einem Unternehmen auswirkt	kein
2	Beitrag des Modells zur Anpassung an den Kostendruck, der auf ein Unternehmen ausgeübt wird	ausreichend

Tabelle *4.4*: Eignung des Managements der IKT zur Lösung der zentralen Problemstellung.

#	Kriterium	Ergebnis-menge
Komplexität der Infrastrukturarchitektur (Problemdimension 1)		
1	Auswirkung des Modells auf den Grad der Komplexität der Infrastrukturarchitektur	ohne
2	Auswirkung des Modells auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betrifft	ohne
3	Auswirkung des Modells auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten	ohne
4	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur im Allgemeinen	wenig
5	Unterstützung des Modells bei der Beherrschung der Komplexität der Infrastrukturarchitektur in einem spezifischen Unternehmen	wenig
Wissenskomplexität und Wissensvolumen (Problemdimension 2)		
1	Unterstützung des Modells bei der Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	wenig
2	Beitrag des Modells, sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgeimein zugänglich zu dokumentieren	wenig
3	Beitrag des Modells, die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass in mehr als 60% aller Fälle nachvollzogen werden kann weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	wenig
4	Beitrag des Modells zur Verwendung einer standardisierten Notation zur Dokumentation der Infrastrukturarchitektur	kein
5	Beitrag des Modells zur Beherrschung der Wissenskomplexität bzw. des Wissensvolumens der Infrastrukturarchitektur	wenig
Historische Entwicklung der Finanzdienstleistungsinformatik (Problemdimension 3)		
1	Beitrag des Modells zur Anpassung an die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in einem Unternehmen auswirkt	kein
2	Beitrag des Modells zur Anpassung an den Kostendruck, der auf ein Unternehmen ausgeübt wird	kein

Tabelle *4.5*: Eignung des Zachman Frameworks zur Lösung der zentralen Problemstellung.

5 Entdeckungszusammenhang – Leittheorien zum Modellentwurf

Neben der Beobachtung im Rahmen der beiden Fallstudien sind die Systemtheorie[251] und die wissenschaftlichen Diskurse im Umfeld der Finanzdienstleistungsinformatik prägend für den Entdeckungszusammenhang dieser Arbeit. Zur Entwicklung eines Modells zur Lösung komplexitätsbedingter Entscheidungsprobleme der Infrastrukturarchitektur eines Finanzdienstleisters dienen die nachfolgenden Leittheorien dem Modellentwurf.

5.1 Ansätze zur Industrialisierung

5.1.1 Industrialisierung im Bereich des Automobilbaus

5.1.1.1 Historische Entwicklung

Manufakturen

Gegen Ende des 19. Jahrhunderts herrschte im Bereich des Automobilbaus das Prinzip einzelner Manufakturen vor. Die damaligen Automobile wurden von wenigen gut ausgebildeten Handwerkern nach den Wünschen der Kunden maßgeschneidert zusammengebaut. Die Folge einer solchen Fertigung nach Kundenwunsch war, dass die produzierten Automobile zum einen sehr teuer waren und nur in begrenzter Stückzahl hergestellt werden konnten.

Fließbandproduktion

Diesem Prinzip der Handwerksproduktion folgte Anfang des 20. Jahrhunderts das Prinzip der Massenproduktion: Henry Ford gründete 1903 die Ford Motor Company. 1913 führte er in seinem Unternehmen das Fließband ein. Damit hielt die Massenproduktion Einzug in die Industrie. In den ersten Jahren erzwang die Massenfertigung Kompromisse in Bezug auf die Modellvielfalt. Die Ford-Werke produzierten exakt ein einziges Modell in einer einzigen Farbe. Den Ford T gab es anfangs nur in schwarz. Ein Umstand, den Ford selber mit dem Satz kommentierte: "Give the customers any colour they want, so long it is black." Mit Hilfe

[251] Vgl. Seiffert (1992)

dieses Modells konnte ein bisher nie da gewesener Absatz erzielt werden. Der Ford T erreichte in 19 Fertigungsjahren die Stückzahl von 15 Millionen.[252]

Im Vergleich zum vorherrschenden Manufakturprinzip setzte Ford auf ein Konzept, das es erlaubte, ein hochkomplexes System wie ein Automobil kostengünstig in Masse herzustellen. Kern dieses Konzeptes war die vollständige und passgenaue Austauschbarkeit einzelner Bauteile, deren Zusammenbau stark vereinfacht wurde. Die Montage der Bauteile erfolgte in Reihe und prägte den Begriff der *Fließbandproduktion*. Ford produzierte das Modell T später mit 9 verschiedenen Karosserieformen. Der Käufer hatte die Auswahl zwischen 4 Farben.[253]

Das Konzept der Fließbandproduktion wurde später von General Motors zur Produktion unterschiedlicher Modellreihen (Pontiac, Cadillac, Buick, Oldsmobile und Chevrolet) adaptiert und um die Standardisierung von Bauteilen erweitert.[254]

Lean Production

Ein Vorteil der Massenproduktion ist, dass in kürzester Zeit ein größtmöglicher Absatz zu niedrigsten Kosten erzielt werden kann. Ein Nachteil dieser Produktionsform ist jedoch, dass individuelle Wünsche eines Kunden nur schwer zu berücksichtigen sind.[255] WOMACK ET AL. bezeichnen daher die Phase der Automobilproduktion nach der Phase der Massenproduktion als „Die zweite Revolution der Autoindustrie“[256]. Grund für den Ausdruck „Revolution“ ist das von Enji Toyoda und Taiichi Ohno entwickelte Produktionsprinzip, das den Namen *Lean Production* trägt. Im Kern geht es bei der Lean Production darum, alle Prozesse auf die Wertschöpfung zu fokussieren und „Verschwendung“ zu vermeiden.[257]

Im Produktionsbereich selbst wird den Mitarbeitern ein Maximum an Aufgaben und Verantwortlichkeiten übergeben. Nach ZINK werden hierbei die Arbeitsteilung und die Fließbandproduktion weitgehend beibe-

---

[252] Vgl. Sorenson/Williamson (1956), S. 1ff bzw. Chandler (1964), S. 1ff (nach Rothe (2004), S. 54)
[253] Vgl. Womack et al. (1992)
[254] Vgl. Chandler (1964), S. 145ff
[255] Anmerkungen: Weitere Nachteile der Massenproduktion werden in Womack et al. (1992), S. 53ff aufgelistet.
[256] entsprechend dem Titel ihres Werks Womack et al. (1992)
[257] Vgl. Womack et al. (1992)

halten.[258] Auf der Grundlage von hochflexiblen und zunehmend automatisierten Maschinen werden die Vorteile von handwerksartiger Produktion und Massenproduktion verbunden.

Den Namen einer schlanken Produktion hat nach WOMAK ET AL. der Forscher John Krafcik diesem Prinzip verliehen, da es nach seinen Beobachtungen von allem weniger einsetzt als die Massenproduktion: die Hälfte des Personals in der Fabrik, die Hälfte der Produktionsfläche, die Hälfte der Investitionen in Werkzeuge und die Hälfte der Zeit zur Entwicklung eines neuen Produktes.[259] Das Prinzip der Lean Production wurde im Laufe der Jahre von allen Automobilherstellern adaptiert.

Komplexitätszuwachs im Automobilbau

Kernproblem der industriellen Entwicklung von Gütern ist die Komplexität des zu entwickelnden Produkts. GÖPFERT und STEINBRECHER identifizieren folgende Merkmale der industriellen Entwicklung, die als allgemeingültige Komplexitätstreiber identifiziert werden können:[260]

- Durch eine Vielzahl einzelner Komponenten, die zur Fertigung des industriellen Guts herangezogen werden, entstehen nur schwer überschaubare Produkte.
- Zwischen den einzelnen Komponenten entstehen Interdependenzen.
- Eine Vielzahl von Aufgabenträgern und Projektbeteiligten produzieren unklare Zuständigkeiten.
- Zwischen internen und externen Aufgabenträgern und Entwicklern entsteht ein hoher Koordinationsbedarf.
- Es existieren unterschiedliche Anforderungen an die zu fertigen Produkte. Dies impliziert die Existenz unterschiedlicher Anforderungen an die Komponenten.

Auch die Entwicklung von Fahrzeugen hat im Laufe der Jahre deutlich an Komplexität zugenommen: Der Käufer eines Automobils erwartet frei konfigurierbare Ausstattungsmerkmale wie Motorisierung, Farben, Materialien oder Komfort- und Elektronikumfänge. Die Folge ist eine enorme Produktkomplexität, getrieben durch die Variantenvielfalt der wählbaren Bestellumfänge: im Werk Sindelfingen der des Automobilbauers

[258] Vgl. Zink (1992), S. 13.
[259] Vgl. Womack et al. (1992), S. 16
[260] Vgl. Göpfert/Steinbrecher (2000), S. 20ff

DaimlerChrysler waren im Jahr 1998 statistisch gesehen nur 2,2 Fahrzeuge der ca. 420.000 ausgelieferten Fahrzeuge identisch.[261]

Mit den Anforderungen der Individualisierung steigen die Erwartungen an innovative Produkte, Lieferzeit, Liefertermintreue, Qualität und Preis. In diesem Zusammenhang haben sich wichtige Zielgrößen wie *Time-to-Market* und *Time-to-Customer* in der Branche etabliert, an deren Optimierung die Hersteller kontinuierlich arbeiten.[262] Time-to-Market optimiert den Produktentstehungsprozess bis zur Serienreife und bewertet die Fähigkeit zur Verkürzung der Produktlebenszyklen. Time-to-Customer (auch Order-to-Delivery genannt) optimiert den Kundenauftragsprozess und die Logistikkette (die sog. *Supply Chain*) von der Fahrzeugbestellung bis zur Auslieferung an den Kunden.[263]

Ein weiterer Komplexitätstreiber ist die Modellvielfalt der Automobilbauer: Aus strategischen Gründen werden neue Fahrzeugsegmente besetzt und die Produktlebenszyklen verkürzt. Beispiele für solche Nischenprodukte sind SUVs (Sport Utility Vehicles), Roadster, Compact Vans oder Shuttlefahrzeuge.[264]

5.1.1.2 Entwicklungen zur Reduktion Komplexitätszuwachses im Automobilbau am Beispiel Volkswagen

Plattformstrategie

Die Entwicklung einer zunehmenden Individualisierung des Automobils und des Markt-/Kundenanspruchs zu mehr Flexibilität hat bis heute zu zwei wesentlichen Implikationen geführt: Zum einen zwingt die hohe Produkt- und Variantenkomplexität die Automobilhersteller zur Verlagerung der Materialbereitstellungskomplexität. Zum anderen müssen die Kundenanforderungen möglichst schnell und flexibel umgesetzt werden.

Um dieser Komplexitätssteigerung zu begegnen haben die Automobilhersteller Modularisierungs- und Outsourcingstrategien entwickelt. Im Zuge zunehmender weltweiter Konkurrenz haben sich unterschiedlichste Produktionsprinzipien basierend auf dem Kernprinzip der Lean Production bei den einzelnen Automobilherstellern etabliert. MEFFERT nennt

[261] Vgl. Schick/Binder (1998)
[262] Vgl. Stoßberg/Hellingrath (2002)
[263] Vgl. Hellingrath et al. (2000)
[264] Vgl. Dudenhöfer (2001)

als Beispiel die 1993 von Ferdinand Piëch beim Autobauer Volkswagen eingeführte *Plattformstrategie*. Ziel dieser Strategie ist es, so MEFFERT, eine große Produktpalette und Modellvielfalt bei reduzierten Kosten durch eine größtmögliche Synergienutzung zu ermöglichen:[265] Das Prinzip der Plattformstrategie ist, dass verschiedene Plattformen entwickelt werden, die mit einer vorgegebenen Kombination aus Vorderachse, Lenkung und Motor, Längsträger, Boden Hinterachse und Tank zusammengebaut werden. Jedes neu zu entwickelnde Automobil muss auf der Basis einer dieser vordefinierten Plattformen entwickelt werden. Karosserien werden letztendlich wie „Hüte“ auf die Plattformen aufgesetzt, ohne dass die Plattformen Einfluss auf das Erscheinungsbild des Automobils nehmen. Für die innerhalb des VW-Konzerns (incl. Audi, Skoda etc.) zu entwickelnden Automobile standen nach der Einführung des Prinzips 17 Plattformen zur Verfügung. Nach mehreren Optimierungs- und Analyseschritten konnte deren Zahl auf 4 reduziert werden. Jede der Plattformen verfügt über bestimmte qualitative Eigenschaften (Länge, Breite, Motorleistung etc.), die die Ingenieure bei der Entwicklung neuer Fahrzeuge berücksichtigen müssen.

Die Plattformen sind eine Entwicklung von Ingenieuren für Ingenieure. Sie ermöglichen Entwicklern neuer Fahrzeuge auf technisch erprobte Bauelemente zurückzugreifen. Diese Möglichkeit ist gleichzeitig auch eine Verpflichtung. Nach MEFFERT werden 60% aller Kosten für die Entwicklung und Fertigung eines Autos über die Plattform bestimmt.

Dass die Modellvielfalt bei der Anwendung des Plattformprinzips keinen Abbruch erleidet, zeigt die Tatsache, dass u. a. die Modelle VW Golf IV, VW New Beatle, Audi A3, Audi TT, und Seat Leon auf ein- und derselben Plattform gebaut wurden. Nach Meffert findet eine Angleichung der unterschiedlichen Modelle „[b]ei alledem, was Kunden nicht sehen“[266] statt.

Meffert erkennt folgende Vorteile der Anwendung der Plattformstrategie:[267]

---

[265] Vgl. Meffert (2000), S. 1331ff
[266] Meffert (2000), S 1331
[267] Vgl. Meffert (2000), S. 1332

- Der Entwicklungsaufwand beschränkt sich auf eine Plattform für alle Modellreihen eines Segments und teilt sich somit durch die Zahl der Produktionsvolumina.
- Die Entwicklungszeiten für neue Fahrzeuge sinken drastisch, da eine weitreichend erprobte Basis für alle Konzernunternehmen zur Verfügung steht. Der Autobauer kann schneller als vorher auf Veränderungen am Markt reagieren.
- Die Anlaufkosten einer Produktion sinken und die Qualität erreicht ein höheres Niveau.
- Beschaffung, Produktion und Teileversorgung für den Service werden vereinfacht.

Der Nachteil der Anwendung der Plattformstrategie von Volkswagen ist, dass die enge technische Anlehnung aller Modelle des gesamten Konzerns die Gefahr einer „Eigenkannibalisierung" insich birgt. Die produzierten Modelle der Konzerntöchter unterscheiden sich nur noch durch ihre Optik. MEFFERT bezeichnet dies als „markentypische Hüte".[268]

Modulstrategie

Im Jahr 2001 hat Volkswagen das Prinzip der Plattformstrategie in Form einer sog. *Modulstrategie* erweitert:[269] Im Rahmen der Modulstrategie wird die Verwendung gleicher Teile für elf ausgewählte Module auf mehrere Fahrzeugklassen angewendet werden. Als Folge können konzernweit beispielsweise gleiche Bremsen im Golf- und Passat verwenden werden.

Module sind nach THALER komplettierte, funktionsfähige Baugruppen und Komponenten, die einbaufertig angeliefert werden.[270] Die Lieferanten, die die Baugruppen produzieren, werden als *Modullieferanten* bezeichnet. Lieferanten, die Einzelteile liefern, werden *Teilefertiger* genannt. Nach BULLINGER und THALER besteht ein Unterschied zwischen der Modullieferung und der Teilefertigung. Während ein Teilefertiger Einzelteile gemäß einer vorgegeben Spezifikation fertigt, muss ein Mo-

[268] Vgl. Meffert (2000), S. 1332
[269] Vgl. VW (2004), S. 16ff
[270] Vgl. Thaler (1999), S. 87

dullieferant eine hohe Ingenieursleistung in Form der Spezifikation und Berechnung erbringen.[271]

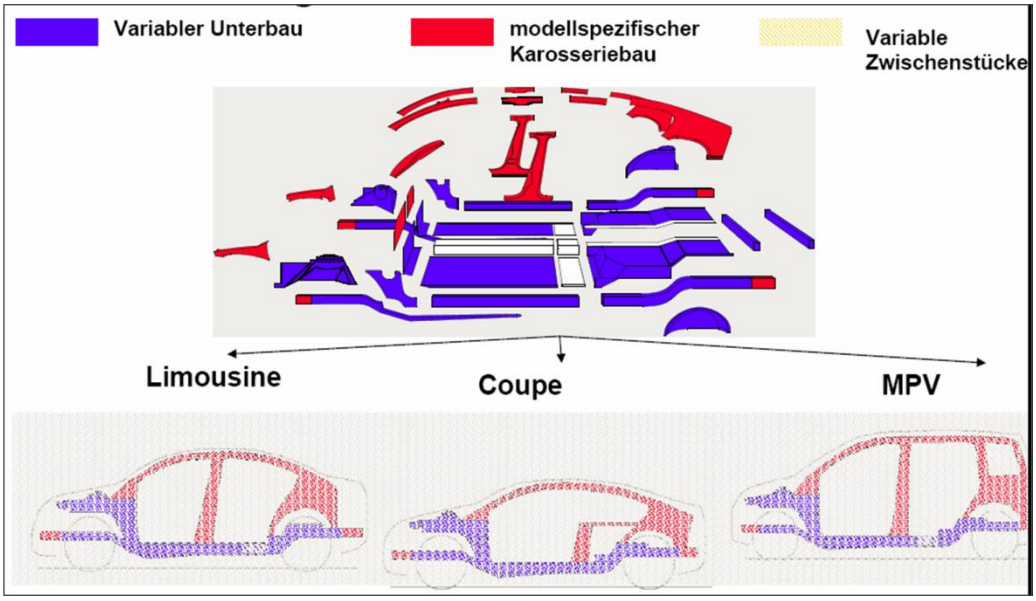


Abbildung 5.1: Die Module der Modelle des Volkswagen Konzerns (Quelle: Unbekannt).

Wie der in Abbildung 2.3 dargestellte Aufbau eines Fahrzeug-„Rohlings“ im Volkswagenkonzern zeigt, bauen die einzelnen Modelle des Konzerns auf einem variablen Unterbau auf, der für die verschiedenen Modelle der gleiche ist. Die Individualität der einzelnen Modelle ergibt sich durch den von MEFFERT als „Hut“ bezeichneten modellspezifischen Karosseriebau. Im Rahmen der Plattformstrategie wurden innerhalb einer Fahrzeugklasse[272] die gleichen Teile verwendet. Im Rahmen der Modulstrategie werden die definierten Module in vier Kategorien (I bis IV) gemäß ihren qualitativen Eigenschaften unterteilt. Mit Hilfe dieses Konzept sind die Volkswageningenieure in der Lage, ein Modul (z. B. eine Bremsanlage) in unterschiedlichen Fahrzeugmodellen (= Fahrzeugklassen) einzubauen.

Nach PIECH ergeben sich folgende Vorteile dieses Konzeptes:[273]

> Mehrfachverwendung von Komponenten.

---

[271] Vgl. Bullinger/Thaler (1992)

[272] Anmerkung: Die Fahrzeugklassen der Automobilhersteller besitzen firmeninterne Codes. A00 ist beispielsweise das Kürzel für ein Modell der Klasse VW Lupo. In der Klasse B sind u. a. die Modelle VW Passat und Skoda Octavia angesiedelt.

[273] Vgl. Piech (2004), S. 198ff; Anmerkung: Nachteile dieses Verfahrens werden nicht genannt. Es lässt sich aber vermuten, dass diese den Nachteilen der Plattformstrategie entsprechen.

- Erhöhung der Anlaufqualität durch Nutzung und Erfahrung.
- Geringere Kaufteilpreise und Investitionen.
- Geringerer Entwicklungsaufwand.
- Weltweiter Teileaustausch.

Das hierbei angewendete Prinzip wird auch als *Baukastensystem* bezeichnet. Unter einem Baukastenprinzip oder auch Baukastensystem wird per Definition eine Methode zum Aufbau von komplizierten technischen Systemen mithilfe einfacher Bausteine verstanden. Die Herstellung, Überprüfung und der Austausch dieser Bausteine soll sich wirtschaftlich rentieren. Durch Kombinationsmöglichkeiten und eine gleichzeitige Funktionsflexibilität soll eine wirtschaftlich optimale Anpassung des Gesamtsystems an die Aufgabenstellung ermöglicht werden.[274]

5.1.2 Industrialisierung im Bereich des Software-Engineerings

Balzert, Bauer

Der Begriff *Software-Engineering* wurde erstmals vor 40 Jahren eingeführt. Die Notwendigkeit der Einführung sah BAUER[275] in der Tatsache begründet, dass die bis dahin praktizierte *Softwareentwicklung* eine systematische und ingenierswissenschaftliche Disziplin ist.[276] BALZERT spricht in diesem Zusammenhang von einer Software-Technik, deren Teilgebiet die Softwareentwicklung ist, und die die "[…] zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen[.]"[277] betreibt.

Taubner

TAUBNER[278] erkennt in der technischen, manageriellen und wirtschaftlichen Professionalisierung des Software-Engeneerings den Grund dafür, dass heutzutage von einer *Software-Industrialsierung* gesprochen wird. Er sieht eine produktive, laufende

[274] Vgl. Brockhaus (1997), Band I, S. 470
[275] Bauer (1993)
[276] Vgl. Taubner (2005)
[277] Balzert (2001), S. 36
[278] Vgl. Taubner (2005)

Software, die das Ergebnis eines Softwareentwicklungsprozesses ist, als Produzent automatisierter Geschäftsprozesse bzw. Steuerungsinstrument von Geräten. Damit wird diese Form der Software auf die gleiche Ebene gesetzt wie die Maschinen einer Fabrik, die Werkzeuge und Güter produzieren. Software ist demzufolge ein Hilfsmittel zur Industrialisierung der diese Software einsetzenden Anwendungsdomäne (z. B. Finanzdienstleistung). TAUBNER hält jedoch den Prozess der Softwareentwicklung für industrialisierenswert und definiert:[279] Unter dem Begriff der *Software-Industrialisierung* wird die Entstehung und Ausbreitung der Softwareentwicklung mit industriellen Mitteln[280] verstanden.

Klassische Industrie	**Softwarebranche**
Massenproduktion	Softwareprodukte
Arbeitsteilung, Fließband	Zerlegung der Arbeit, Vorgehen
Spezialisierung	Spezialisierung
Rationalisierung	Rationalisierung
Automatisierung	Werkzeuge, MDA[281]
Kontinuierliche Verbesserung	ISO900x, CMMI Level 5
Standardisierung	Standardisierung (IP, J2EE etc.)
Plattformstrategie (z. B. Autoindustrie)	z. B. Common IT-Plattform[282]
Modulkomponenten (z. B. Autoindustrie)	Komponentenbasierte Architektur
Verringerung der Fertigungstiefe	Outsourcing
Nutzung globaler Märkte	Nutzung globaler Märkte
Globale Nutzung von Lohngefälle	Offshoring

Tabelle 5.1: Typische Mittel klassischer Industrien werden auf die Softwarebranche übertragen (Quelle: Taubner (2005)).

Die Analogie zwischen der klassischen Industrie und der Softwarebranche wird von TAUBNER durch eine Gegenüberstellung der sog. *typischen industriellen Mittel* hergeleitet (Tabelle 5.1). Tenor dieser Gegenüberstellung ist, dass die Mittel, die die klassische Industrie zu einem indust-

[279] Vgl. Taubner (2005)
[280] Siehe Tabelle 5.1
[281] Anmerkung: Bei der Model Driven Architecture (MDA) stehen Modelle im Zentrum des Software-Entwicklungsprozesses. Nach Kempa/Mann (2005) ist das Ziel der MDA die möglichst automatisierte Ableitung plattformspezifischer Modelle aus plattformunabhängigen Modellen.
[282] Anmerkung: Die Common IT-Plattform ist nach Ganswindt (2004) ein Projekt der Mitglieder der sog. STAR Alliance (einer Kooperation internationaler Fluggesellschaften), dessen Ziel es ist, die zentralen Kernfunktionen der beteiligten Fluggesellschaften (z. B. Reservierung, Ticketmanagement und Flugabfertigung) über eine gemeinsame Soft- und Hardware-Plattform zu betreiben.

rialisierten Gewerbe machen, artverwandt mit denjenigen Mitteln sind, die in der Softwarebranche zum Zweck der Adaption eines industrialisierten Produktionsprozesses eingesetzt werden.

Herzwurm et al.

Das Anstreben einer industrialisierten Entwicklung von Software wurde im Rahmen der sog. *Software Factory* verfolgt. Nach dem Vorbild der industriellen Produktion „[…] werden mit Hilfe standardisierter, tergestützter Werkzeuge auf Basis eines formalisierten, mittels technischer und ökonomischer Kennzahlen kontrollierten Prozesses Softwareprodukte […] erstellt."[283]

Denert

DENERT sieht hingegen keine Analogie zwischen der klassisch-industriellen Produktion von Gütern im Stil Henry Fords und der „Produktion" von Software. Schließlich sei die Vorstellung einer Softwarefabrik mit streng nach Reih und Glied ausgerichteten „Softwerkern" kein erstrebenswerter Arbeitsplatz. Ebenso sei der Vergleich mit einem Arbeiter am Fließband falsch, da weniger ein minderqualifizierter Arbeiter denn ein hochqualifizierter Ingenieur in der Softwareentwicklung benötigt werde.[284]

Janßen

JANßEN charakterisiert das Stadium der Softwareentwicklung nach heutigem Stand als „Manufaktur und Kunsthandwerk"[285]. Diese Charakterisierung basiert auf dem Vergleich zwischen der Einstellung eines Software-Entwicklers zu seiner Arbeit und deren Ergebnissen von den Teilnehmern an anderen industriellen Entwicklungsaufgaben. Diese Einstellungen unterscheiden sich auf Basis JANßENS Annahmen wie folgt:

> Ein Architekt, der Gebäude baut, befasst sich nach der Fertigstellung des Gebäudes nicht mit dem Gebäudemanagement. Die emotionale Bindung zu dem Gebäude reduziert sich auf ein Minimum, so dass er sich einem neuen Projekt widmen möchte. Im Fall eines Software-Entwicklers lässt sich nach der Fertigstellung der Applikation durchaus eine enge emotionale Bindung zu dem hergestellten Produkt erkennen. JANßEN folgert dies aus dem Wunsch von Software-

[283] Herzwurm et al. (1994)
[284] Vgl. Denert (1994)
[285] Vgl. Janßen (2005)

Entwicklern „ihre“ Anwendung über den Lebenszyklus begleiten zu können.

> Die Anwendungsentwicklung ist geprägt vom sog. *Not-Invented-Here-Syndrom*. SPIEWAK kommentiert das auch in anderen Bereichen bekannte Phänomen wie folgt: „Innovationen sind lästig für jeden, der sich an das Bestehende gewöhnt hat, denn er muss sich umstellen. Krempeln Innovationen eine Technik völlig um, können sie sogar brandgefährlich sein, für alle, die an der alten Technik verdienen.“[286] Die Einführung von Innovationen birgt auch einen hohen Risikoanteil in sich, da zur Zeit der Einführung unbekannt ist, wie sich die Neuerung auf Dauer verhält. SPIEWAK erläutert im Zusammenhang mit der Ablehnung einer neuen wegweisenden Technologie in der Automobilindustrie das Syndrom als eine „[…] institutionalisierte Abwehr der Hauskonstrukteure gegen alles, was von außen kommt“[287]. JANßEN beschreibt, dass die Verwendung von vorgegebenen Bausteinen, Entwicklungsumgebungen etc. in der Softwareentwicklung auf einen emotionalen Widerstand trifft. Diese Blockadehaltung sorgt dafür, dass Projekte oftmals scheitern.

> Die emotionale Beziehung, die ein Software-Entwickler zu *seiner* Entwicklungsumgebung – seinem „Werkzeug“ – aufbaut, ähnelt weniger der eines Dachdeckers zu seinem Hammer als eher der eines Geigers zu *seiner* Stradivari.

Auf Basis dieser Annahmen schließt JANßEN, dass die derart geprägte Softwareentwicklung in der heutigen Zeit und unter den gegeben Umständen im Umfeld der Finanzdienstleister nicht mehr zeitgemäß ist. Übertragen auf die Automobilindustrie gleiche sie der Entwicklung eines Rolls Royce oder Bentley aber nicht der eines Automobils für die Masse vom Typ Volkswagen. Die Geschwindigkeit, mit der Änderungen in die bestehenden betrieblichen Anwendungssysteme aufgenommen werden müssten, habe sich im Laufe der Jahre derart erhöht, dass unter dem

[286] Spiewak (1999)
[287] Spiewak (1999)

Fokus des Kostenorientierten Time-to-Market Prinzips auf die Detailverliebtheit eines Donald Knuth[288] verzichtet werden müsse.

Berensmann

BERENSMANN testiert der IT (also einschließlich der von JANßEN sierten Softwareentwicklung) im Vergleich zu anderen produzierenden Industrien einen frühen Reifegrad:[289] Es können vier Stufen von Produktionsverfahren identifiziert werden, wobei jede Stufe einen Volumengrad der jeweils produzierten Menge impliziert. Mit zunehmendem Volumengrad sinken die Stückkosten. Getreu BERENSMANN „[…] befindet sich die IT-Industrie und mit ihr die IT-Funktionen in Unternehmen bestenfalls an der Grenze zwischen Handwerker und Fabrikation“[290].
Diese Einschätzung entspricht in etwa der von JANßEN. BERENSMANN lässt jedoch offen, wie er die „Stückkosten“ der heutigen IT berechnet. Grundsätzlich können die von der Softwareentwicklung „produzierten“ Applikationen auf die beiden Kategorien Standard- und Individual-Software aufgeteilt werden. Während Individual-Software im Sinne eines Unikats entworfen wird (d. h. sie wird „unmittelbar und speziell für die konkrete Anwendung entwickelt“[291]), wird eine Standard-Software für den mehrfachen Einsatz entwickelt und vom Software-Hersteller bezogen.

Jochum

Im Gegensatz zu JANßEN und BERENSMANN glaubt JOCHUM eindeutige Anzeichen einer industrialisierten Softwareentwicklung erkennen zu können: „Gerade im Bereich der Softwareerstellung finden sich inzwischen industrieähnliche Produktionsabläufe.“[292] Als Faktoren einer zunehmenden Industrialisierung im Umfeld der IT nennt JOCHUM *Standardisierung* und *Kommoditisierung*[293]. Diese beiden Faktoren gelten sowohl für Hardware als auch für Software. Beispiele hierfür sind Server, Betriebssysteme, sog. *Laufzeitumgebungen*[294] oder *Software-*

[288] Anmerkung: Donald Knuth hat in mühevoller Kleinstarbeit TeX entwickelt.
[289] Vgl. Berensmann (2005)
[290] Berensmann (2005)
[291] Stahlknecht/Hasenkamp (1997), S. 250
[292] Jochum (2005)
[293] Anmerkung: Der Begriff *Kommoditisierung* stammt ursprünglich aus dem Englischen („commodisation“). Er bezeichnet eine Entwicklung weg vom Handel mit langfristigen Verträgen hin zu einem kurzfristigen Markt.
[294] Anmerkung: Die Laufzeitumgebung ist ein einem Computerprogramm während seiner Laufzeit zur Verfügung stehender universeller Satz von Funktionen und Variablen. In manchen Fällen ist die Laufzeitumgebung eine virtuelle Maschine, die platt-

Plattformen[295]. Bei diesen Produkten handelt es sich um eine spezialisierte Ware, die von Dienstleistern mit standardisierten Leistungsmerkmalen angeboten wird. So bieten beispielsweise die Hersteller IBM, BEA, Sun Microsystems oder auch Oracle jeweils einen sog. *J2EE Application Server* an, der verschiedene herstellerunabhängige, administrative Grundfunktionen wie die Datenbankverwaltung oder das Management konkurrierender Zugriffe im Rahmen eines von der Firma Sun Microsystems standardisierten Komponentenmodells anbietet. Jede Applikation, die auf Basis einer solchen Software-Plattform entwickelt wird, verwendet allgemein verwendbare Funktionen dieser Plattform. Die eigentliche Implementierung der Anforderungen, die diesen Applikationen zugrunde liegen, findet nach wie vor statt. Die Entwicklung kann hierdurch stärker auf das Wesentliche reduziert werden. Das bedeutet, dass eine Individualisierung jeder Applikation auch unter Zuhilfenahme von standardisierten Software-Plattformen stattfinden kann. Dieses Verfahren ähnelt weitestgehend dem Individualisierungskonzept, das im Rahmen der Plattform- und Modulstrategie bei Volkswagen eingesetzt wird.[296]

Das Indiz nach JOCHUM für eine industrialisierte Softwareentwicklung ist die gezielte Zerlegung der Prozesskette des Herstellungsprozesses von Software. Die innerhalb dieser Prozesskette identifizierten Teilprozesse wie z. B. die Programmierung oder das Testen werden zunehmend von externen Partnern übernommen. Auch der Einsatz standardisierter Software-Plattformen erinnert stark an ein industrialisiertes Vorgehen.

Westphal

Zu einem ähnlichen Schluss kommt auch WESTPHAL:[297] Durch eine sukzessive Spezialisierung einzelner Software-Entwickler ergibt sich ein hochgradig arbeitsteiliger Entwicklungsprozess, der den Prozessen im Bereich der industriellen Produktion sehr nahe kommt. Der Grund für eine derartige Spezialisierung ist, dass die Technologien, die heutzutage im Bereich der Softwareentwicklung angeboten werden, zum einen

---

formunabhängige Befehle in Befehle des jeweils verwendeten Betriebssystems übersetzt. Beispiele hierfür sind die plattformunabhängige Programmiersprache Java und Microsofts .NET Plattform.

[295] Anmerkung: Eine Software-Plattform ist eine Laufzeitumgebung, die über umfangreiche meist administrative Applikationsteile verfügt. Beispiel hierfür ist die Java 2 Plattform, Enterprise Edition (J2EE).

[296] Siehe Abschnitt 5.1.1.2

[297] Vgl. Westphal (2005)

in einer hohen Vielfalt präsent sind, und zum anderen eine hohe Veränderungsfrequenz besitzen. Die Folge dieser Entwicklung ist, dass die Bewältigung dieser „Technologieflut“, wie WESTPHAL die Situation beschreibt, stetig mehr Zeit in Anspruch nimmt. Daher kann ein Software-Entwickler nur noch bei wenigen Technologien eine angemessene Wissenstiefe zu deren optimalem Einsatz in Entwicklungsprojekten sammeln. Ein Entwickler, der ein breit gefächertes und sich in einer kurzen Frequenz änderndes Einsatzgebiet vorweist (z. B. Frontend-Entwicklung, Geschäftslogik-implementation oder Datenbankzugriffsoptimierung), kann infolge dessen nur oberflächlicheres Wissen in den jeweiligen Einsatzgebieten aufbauen und vorweisen, als ein Entwickler, der nur in einem einzigen Einsatzgebiet tätig ist. WESTPHAL stellt fest:[298]

- Der heutigen technologischen Differenziertheit der Softwareentwicklung steht keine entsprechende Differenziertheit der Tätigkeitsfelder eines Software-Entwicklers gegenüber.
- Nur durch eine wesentlich stärkere Spezialisierung als bisher wird es einem Software-Entwickler gelingen, mit der Neuerung der von ihm beherrschten Technologie Schritt zu halten.

JANßEN schlägt in diesem Zusammenhang eine Aufteilung der Gesamtarbeit des Software-Engineerings auf vier Rollen vor, die von den jeweiligen Anwendungsszenarien und Technologien zu entkoppeln sind, und die unterschiedliche, persönliche Kompetenzen erfordern:[299]

- *Architekt*: Die Aufgabe des Architekten ist die Aufnahme des Anwendungsproblems, dessen Strukturierung sowie die Skizzierung eines Lösungswegs.
- *Ingenieur*: Der Ingenieur analysiert den vom Architekten skizzierten Lösungsweg und entwickelt auf dieser Basis technische Realisierungsvorschläge.
- *Projektmanager*: Der Projektmanager plant, organisiert und überwacht die Arbeit.

[298] Vgl. Westphal (2005)
[299] Vgl. Janßen (2005)

- *Entwickler*: Der Entwickler implementiert letztendlich die Applikation auf Basis der Vorgaben des Architekten und des Ingenieurs.

Indikatoren für ein industrialisiertes Software-Engineering

Die bisherigen Ausführungen haben gezeigt, dass die alleinige Praktizierung des von BAUER definierten Software-Engineerings nicht ausreicht, um von einem industrialisierten Software-Engineering sprechen zu können. Auch die von BALZERT definierte Software-Technik mit ihrem ingenieurmäßigen Vorgehen, welches von OTT mit den Prinzipien *systematisches Vorgehen*, *Denken in Baugruppen*, *Wiederverwendung*, *Prozessstrukturierung* und *Qualitätsbewusstsein* beschrieben wird,[300] umfasst nicht alle Aspekte einer industrialisierten Vorgehensweise.

Auf Basis der in diesem Abschnitt verwendeten Literatur wird im Folgenden von einem *industrialisierten Software-Engineering* gesprochen, sofern die in Tabelle 5.2 aufgelisteten Indikatoren auf das jeweils praktizierte Vorgehen im Umfeld der Softwareentwicklung zutreffen.

HERZWURM ET AL. stellen bereits 1994 fest, dass „[…] der Begriff Software Factory seine Popularität verloren hat, die Idee ist lebendig und ist weiterhin wirksam.“[301] Die analoge Betrachtungsweise einer Fabrikähnlichen Herstellung von Software mit dem permanenten Fokus auf alleinige Automatisierung des Herstellungsprozesses hat mit der Software Factory ein Ende gefunden. Die neue industrialisierte Betrachtungsweise versucht den Herstellungsprozess ähnlich wie in der Automobilindustrie zu zerlegen. Mehrere Autoren[302] versuchen sich zum Zweck der Vermittlung ihres Bildes eines industrialisierten Software-Engineerings in dem Vergleich der Entwicklung eines Automobils mit der Entwicklung eines Software-Produktes.

#	Indikatoren	Autor(en)
A	Eine arbeitsteilige und ingenieurmäßige Entwicklung wird praktiziert.	Balzert, Bauer
B	Eine technische, managerielle und wirtschaftliche Professionalisierung findet statt.	Taubner
C	Auf Basis eines formalisierten, mittels technischer und ökonomischer Kennzahlen kontrollierten Prozesses werden Softwareprodukte erstellt.	Herzwurm et al., Taubner
D	Die sprichwörtliche „Liebe zum Detail“ bzw. die emo-	Janßen

[300] Vgl. Ott (1994), S. 32ff
[301] Herzwurm et al. (1994)
[302] Zum Beispiel JANßEN, JOCHUM, TAUBNER oder MAIDL

	tionale Bindung zum hergestellten Produkt wird abgebaut.	
E	Im Rahmen der Softwareentwicklung wird auf fertige und vorgegebene Bausteine, Werkzeuge etc. zurückgegriffen.	Janßen
F	Der Entwicklungsprozess ist Kostenorientiert und reaktiv.	Janßen, Berensmann
G	Standardisierung und Kommoditisierung sind bereits etabliert und nehmen permanent zu.	Jochum
H	Die Prozesskette des Herstellungsprozesses von Software wird gezielt zerlegt.	Jochum
I	Teilprozesse des Herstellungsprozesses von Software werden von Subunternehmen ausgeführt, wodurch eine Verringerung der Fertigungstiefe angestrebt wird.	Taubner, Jochum

Tabelle 5.2: Indikatoren für das Praktizieren eines industriellen Software-Engineerings.

Solange eine Analogie zwischen der Entwicklung von Software und der Entwicklung von Automobilen aufrechterhalten werden kann, scheint eine Adaption des Verhaltens der Fahrzeuge produzierenden Branche innerhalb der Software produzierenden Branche legitim.

In Bezug auf die von DENERT kritisierte Adaption der streng nach Reih und Glied ausgerichteten „Softwerker“, die ähnlich einem Fließbandmitarbeiter nur minderqualifiziert sein müssen, scheint die Adaption der ingenieurmäßigen Tätigkeit eines Fahrzeugtechnikers, der Automobile *entwickelt*, haltbarer zu sein.

Die in Abschnitt 5.1.1.2 vorgestellte Praktizierung der Plattform- bzw. Modulstrategie bei Volkswagen ist eine weitere Steigerung ingenieurmäßiger Tätigkeit im Umfeld der Automobilindustrie, die nachweißlich die Komplexität der angebotenen Modelle reduziert, die Herstellungskosten senkt, die Qualität dieser Fahrzeuge erhöht und dennoch die Individualisierung einzelner Modelle erlaubt. Um derartige Plattformen und Module definieren zu können, werden Ingenieure benötigt. Diese Ingenieure müssen Modellübergreifend tätig sein und Synergien zwischen den einzelnen Fahrzeugtypen Konzernübergreifend erkennen können. Sie müssen die Eigenschaften der von ihnen definierten Plattformen und Module in Form von qualitativen Attributen so aufbereiten, dass sie von denjenigen Ingenieuren für deren modellindividuelle Entwicklung eines Fahrzeugs (z. B. eines Golf V) verwendet werden können.

5.1.3 Abgrenzungen

Die Adaption des industriellen Verhaltens der Automobilindustrie auf den Bereich Software Engineering erfordert eine differenzierte Betrachtung der beiden Ingenieurdisziplinen. Obwohl der eigentliche Fokus dieser Arbeit das Architekturmanagement im Bereich Infrastruktur eines Informationssystems ist, so ist die Infrastrukturarchitektur Teil der Softwarearchitektur. Die Akzeptanz eines industrialisierten Vorgehens im Bereich der Softwarearchitektur senkt die Hemmschwelle für ein industrialisiertes Vorgehen im Bereich der Infrastrukturarchitektur. Da die Literaturbasis dieser Arbeit industrielles Verhalten rein im Software Engineering Bereich beschreibt, wird davon ausgegangen, dass ein industrialisiertes Software Engineering auch ein industrialisiertes Vorgehen im Infrastrukturarchitekturbereich ermöglicht.

Marktumfeld eines hauseigenen IT-Dienstleisters

Das Marktumfeld eines hauseigenen IT-Dienstleisters ist beschränkt. Die IT-Dienstleister der Finanzdienstleister wurden vereinzelt als eigenständige Gesellschaften ausgegliedert. Beispiele hierfür sind die Postbank Systems, die Sparkassen Informatik oder die HVB Systems. Der Kunde dieser eigenständigen Gesellschaften ist – wie im Fall einer großen IT-Abteilung bzw. eines IT-Ressorts – der Mutterkonzern bzw. die zu ihm gehörenden Tochtergesellschaften. Hierdurch hat ein interner IT-Dienstleister weniger Freiräume als ein am Markt agierendes IT-Unternehmen. Als Beispiel hierfür nennt KÜTZ[303] den Umstand, dass ein interner IT-Dienstleister seine Preisvorstellungen mit der Informationswirtschaft und mit der obersten Leitung der Organisation abstimmen muss.

DIETRICH und SCHIRRA umschreiben die Aufgabe eines internen IT-Dienstleisters wie folgt: „Der oder die internen IT-Provider haben die Aufgabe, sich von der Mutter Aufträge zu sichern, diese in wettbewerbsfähiger Art und Weise zu erfüllen und sich so aufzustellen und auszurichten, dass sie dies wirtschaftlich tun können.“[304] Hieraus ist zu folgern, dass ein interner IT-Dienstleister nur auf individuelle Wünsche seines Kunden reagiert. Ein Informationssystem, das von ihm in Auftrag

[303] Vgl. Kütz (2005), S. 127
[304] Dietrich/Schirra (2004), S. 4

gestellt wird, ist ein an den Kunden individuell angepasstes System. Der Kunde ist aufgeteilt in verschiedene Fachabteilungen, die unterschiedliche Bedürfnisse haben. Demnach ist jedes System bzgl. seiner Anforderungen unterschiedlich. Bezogen auf den Mikromarkt eines einzelnen Finanzdienstleisters ist jedes System für sich gesehen ein Unikat. Es kann jedoch vermutet werden, dass, bezogen auf den Makromarkt aller Finanzdienstleister, Informationssysteme mit gleichen Anforderungen redundant vorkommen. Der Bereich der sog. *Gesamtbanksteuerung* ist beispielsweise bei allen Banken funktional gleich ausgeprägt.[305]

Der in Abschnitt 2.1.2 identifizierte Kostendruck, unter dem die hauseigenen IT-Dienstleister operieren, zwingt sie dazu, ihrem Kunden eine mindestens qualitativ gleichwertige Dienstleistung zu maximal gleichen Kosten anzubieten. Der Auftraggeber im Einzelnen ist jedoch nicht der Mutterkonzern sondern eine Fachabteilung desselbigen. Daher muss eine Kostenreduktion als Makrosicht über alle Fachabteilungen des Auftraggebers hinweg von einer Kostenreduktion als Mikrosicht für einzelne Fachabteilungen unterschieden werden.

Hierzu ein Beispiel: Angenommen 5 Fachabteilungen beauftragen innerhalb dreier Jahre in zeitlich versetzten Abständen je 3 neue Informationssysteme. Ziel des IT-Dienstleisters sollte es gemäß den o. g. Ausführungen sein, dem Kunden eine möglichst kostengünstige (aber qualitativ hochwertige) Lösung vorzuschlagen. Jeder der 5 Kunden spricht jedoch die Mikrosicht der Kostenreduktion an, da jede Fachabteilung in der Regel unabhängig von einer anderen Fachabteilung agiert und entsprechende Angebote beim IT-Dienstleister einholt. Wird nur die Mikrosicht der Kostenreduktion je Auftrag betrachtet, könnte dies in diesem Beispiel dazu führen, dass jedes Informationssystem ein anderes Datenbankmanagementsystem von unterschiedlichen Herstellern bezieht, da jeder Hersteller sein Produkt mit unterschiedlichen qualitativen Eigenschaften anbietet. Jedes der beauftragten Informationssysteme hat entsprechende unterschiedliche qualitative Anforderungen. Wie in Abbildung 5.2 dargestellt, stehen den 5 qualitativ unterschiedlichen Anfor-

[305] Vgl. Berensmann (2004), S. 61ff; Lederer (2004), S. 87ff; Neumann (2004), S. 98ff

derungen der Kunden 5 qualitativ den Anforderungen entsprechende Produkte gegenüber. Angenommen jedes der 5 Produkte verursache gemäß seines qualitativen Abdeckungsgrades Kosten (z. B. in Form von TCO) von 1 GE, dann ergäben sich eine Summe von 1*3+2*3+3*3+4*3+5*3 = 45 GE. Angenommen die Hersteller der Produkte 3 und 5 gewährten einen Mengenrabatt (z. B. in Form einer sog. *Konzernlizenz*), so dass ab einer Anzahl von 3 Lizenzen ein Fixum von 19 GE bzw. 21 GE fällig wäre, wäre dies in Summe günstiger als der einzelne Produkteinsatz. Die Kunden 1, 2 und 3 könnten dann das Produkt 3 nutzen, das in den Fällen 1 und 2 „überqualifiziert“ ist. Die Kunden 4 und 5 könnten das Produkt 5 nutzen, das im Fall des Kunden 4 „überqualifiziert“ ist.

Aus Makrosicht ist dies die günstigere Lösung. Aus Mikrosicht der einzelnen Kunden führt dies natürlich dazu, dass es bei einer anteilsmäßigen Kostenberechnung zu höheren Kosten kommen kann, als dies im Fall der Einzellösungen der Fall wäre. Konkret bedeutet dies für die Kunden 1, 2 und 4 einen erheblichen Mehraufwand. Statt 3 GE TCO würde Kunde 1 beispielsweise über 6 GE TCO bezahlen müssen.

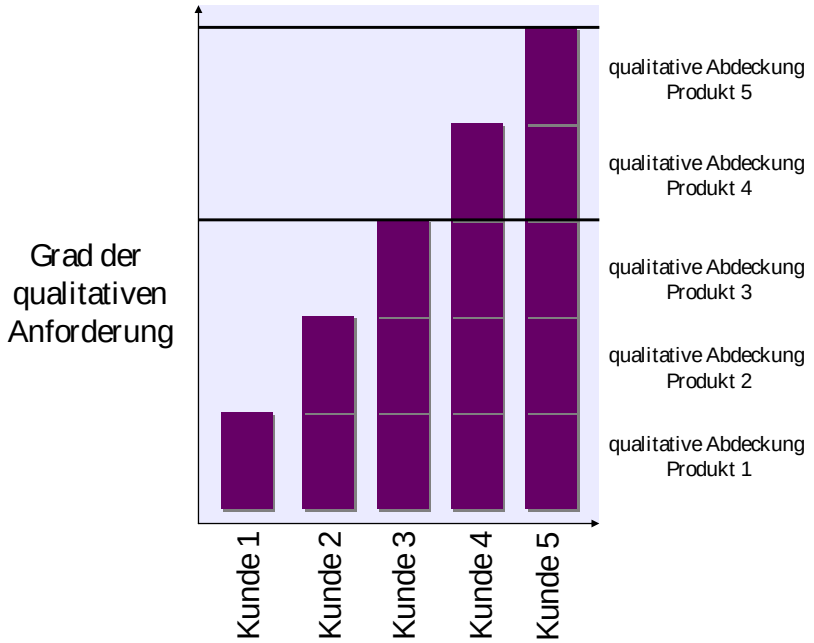


Abbildung 5.2: Gegenüberstellung 5 qualitativ unterschiedlicher Kundenanforderungen und 5 qualitativ er Produktanforderungen. (Quelle: Eigene Darstellung)

Abgrenzung zum Automobilhersteller

Der Analogieschluss zwischen der Industrialisierung des Automobilbaus und der Industrialisierung des Software Engineerings (mit all seinen

Teildisziplinen) erfordert im Kontext einer Fokussierung der Finanzdienstleistungsinformatik eine Abgrenzung. Während ein Automobilhersteller wie Volkswagen einen Massenmarkt mit großen Stückzahlen anspricht, ist der Absatzmarkt in der hausinternen Finanzdienstleistungsinformatik auf eher geringe Stückzahlen begrenzt. Es kann vermutet werden, dass pro Jahr bestimmte Komponenten wie beispielsweise Datenbankserver im 1- bis niedrigen 2-stelligen Bereich in neuen Architekturentwürfen vorgesehen werden. Somit handelt es sich bei der hausinternen Finanzdienstleistungsinformatik eher um die Herstellung von Individualsystemen.

Da die Entscheidungsprobleme, welche Komponenten in einem Infrastrukturarchitekturentwurf untereinander komponiert werden, komplex sind, wird im nachfolgenden Abschnitt untersucht, welche Ansätze im Umfeld des Komplexitätsmanagements im Allgemeinen und im Speziellen existieren.

5.2 Ansätze des Komplexitätsmanagements

Nach MALIK ist unter grundsätzlich vergleichbaren Unternehmen dasjenige besonders leistungsfähig, dem es am besten gelingt, mittels der Anwendung strategischer Konzentration sowie einem Schnittstellen- und Komplexitätsmanagement Einfachheit zu verwirklichen.[306]

Komplexität ist jedoch nicht zwangsweise negativ. In Anlehnung an BLISS sehen BÖHMANN und KRCMAR eine Organisation stets mit einem Trade-Off konfrontiert:[307] Je größer die Alternativenmenge zur Kompensierung von Umweltveränderungen ist, desto größer ist die Möglichkeit sich an zukünftige Umweltzustände anpassen zu können, desto größer ist aber auch die Komplexität des Systems. Ändert sich die Umwelt eines Systems, müssen Elemente der Alternativenmenge gezielt selektiert werden. Dieser Selektionsprozess kostet Zeit. Reicht die Zeittoleranz für die Reaktion auf die Umweltveränderungen nicht aus, ist die Überlebensfähigkeit des Systems gefährdet. In einer dynamischen

[306] Vgl. Malik (1996), S. 184 (nach Hofer (2001), S. 17)
[307] Vgl. Böhmann/Krcmar (2005a); vgl. Bliss (2000), S. 133ff; vgl. auch Rommel et al. (1993), S. 23

Umwelt wird daher vermutet, dass eine geringe Systemkomplexität eine schnellere Reaktion auf Umweltveränderungen impliziert.

5.2.1 Variantenmanagement in nicht-informationstechnischen Bereichen

SCHUH und SCHWENK stellen fest, dass Unternehmen durch die stetig neue Erschließung von Nischenmärkten dem vorherrschenden Preiskampf am Markt zu entweichen versuchen. Alternativ wären sie in der Lage, sich auf ihre Kernkompetenzen zu besinnen und zu versuchen in einer geringen Zahl von Marktsegmenten ihre jeweils angebotene Dienstleistung zu optimieren. Die Folge dieses Handelns ist, dass das Verhältnis zwischen betrieblichem Aufwand und tatsächlich wahrgenommenem Kundennutzen aus der Waage gerät. Das Ungleichgewicht dieser beiden Faktoren produziert stetig ansteigende Komplexitätskosten.[308]

Nach EHRLENSPIEL ET AL. sind diese Komplexitätskosten Opportunitätskosten, da die Variantenerstellung und Pflege Kapazitäten einer Organisation bindet.[309] Das Überangebot an Produktvarianten führt zwangsläufig zu überkomplexen Unternehmen, die die angebotenen Produkte nicht gewinnbringend mit den vom Kunden geforderten Eigenschaften Qualität und Preis produzieren können.[310]

Variantenmanagement & Variantenvielfalt

Die Höhe der internen Komplexität wird durch die Höhe der u. a. durch Kundenbedürfnisse definierten Ausprägung der externen Komplexität beeinflusst. Die von PILLER und WARINGER genannten externen Komplexitätstreiber[311] nehmen direkten Einfluss auf die Variantenvielfalt des Leistungsangebots. Die Ausgestaltung der Höhe der internen Komplexität in Abhängigkeit von der Höhe der externen Komplexität kann durch das sog. *Variantenmanagement* beeinflusst werden.

Das Variantenmanagement umfasst nach FIRCHAU ET AL. alle Steuerungsvorgänge zur Optimierung der sog. *Variantenvielfalt* und zur Be-

[308] Vgl. Schuh/Schwenk (2001)
[309] Vgl. Ehrlenspiel et al. (1998), S. 266 (nach Hofer (2001), S. 16)
[310] Vgl. Schuh/Schwenk (2001)
[311] Vgl. Piller/Waringer (2001), S. 7

herrschung der Auswirkungen variantenreicher Produktspektren.[312] Hierbei beschreibt die Variantenvielfalt die Anzahl und die Verschiedenheit der Varianten eines Bauteils, einer Baugruppe oder eines Produktes.[313]

Nach HOFER ist Variantenmanagement eine Teildisziplin des Komplexitätsmanagements, die als Bindeglied zwischen der durch Kundenbedürfnisse definierten externen Komplexität und der internen Komplexität fungiert. Der Grad der internen Komplexität wird von der Organisation selbst beeinflusst.[314]

SCHUH formuliert das Aufgabengebiet des Variantenmanagements als die Bewältigung der vom Produkt ausgehenden Komplexität (Teilezahl, Varianz etc.) sowie der auf das Produkt einwirkenden Komplexität (Marktdiversifikation, Produktionsabläufe etc.) mittels geeigneter Instrumente.[315]

Demzufolge ist die Ausprägung der internen Komplexität in Abhängigkeit von der Ausprägung der externen Komplexität das Ergebnis eines praktizierten Variantenmanagements, dessen Ziel „[…] die Harmonisierung von Wiederholhäufigkeit und Kundennutzen“[316] ist. Die Aktivitäten eines Variantenmanagements umfassen dabei die Entwicklung, die Gestaltung und Strukturierung von Produkten, Dienstleistungen und Produktsortimenten. Ein Nebenprodukt dieser Aktivitäten ist die frühzeitige Förderung eines Bewusstseins über die jeweils existierende Variantenvielfalt.[317]

In Anlehnung an das Gesetz von Ashby, nach dem das Verhältnis zwischen externer und interner Komplexität idealerweise ausgewogen sein sollte, bedeutet ein Überangebot von Produktvarianten eine zu hohe interne Komplexität in Abhängigkeit von der Markinduzierten externen Komplexität. SCHUH und SCHWENK setzen daher eine optimale Variantenvielfalt des Produktsortiments mit dem Erreichen eines optimalen

[312] Vgl. Firchau et al. (2002), S. 12
[313] Vgl. Firschau et al. (2002), S. 12
[314] Vgl. Hofer (2001), S. 18
[315] Vgl. Schuh/Schwenk (2001), S. 5
[316] Hofer (2001), S. 18
[317] Vgl. Firchau et al. (2002), S. 13

Verhältnisses zwischen der externen und internen Komplexität gleich.[318]

Die Anzahl der von einer Organisation angebotenen Produktvarianten am Markt beruht auf strategischen Entscheidungen des Managements im Kontext der Analyse der externen Komplexitätstreiber. Um das eigene Angebot von dem der Konkurrenz auf dem Markt zu differenzieren, versuchen Unternehmen durch das Angebot zusätzlicher Produktvarianten, sog. *Nischenprodukte*, neue Kunden zu gewinnen bzw. den Nutzen für existierende Kunden weiter zu erhöhen.

Interne und Externe Variantenvielfalt

In Anlehnung an WILDEMANN zählen die Produktbezogene Variantenvielfalt und das Produktdesign zu den internen Komplexitätstreibern produzierender Betriebe.[319] Die Variantenvielfalt kann durch eine sog. *externe Vielfalt* und eine sog. *interne Vielfalt* unterschieden werden.

Nach FIRSCHAU ET AL. repräsentiert die externe Vielfalt die vom Kunden nutzbare Anzahl von Produktvarianten:[320] Damit dieser aus Differenzierungsgründen die Vielfältigkeit des Produktangebots wahrnehmen kann, muss die externe Variantenvielfalt für ihn erkennbar sein. Die angebotenen Produktvarianten setzten sich jedoch in der Regel aus einzelnen Unterelementen (Bauteile, Baugruppe, Prozesse und Produkte) zusammen. Je höher die Unterschiedlichkeit und die gegenseitigen Abhängigkeiten dieser Unterelemente sind, desto höher ist die Ausprägung der internen Vielfalt und desto höher ist der Herstellungsaufwand (und damit implizit der Grad der internen Komplexität).

SCHLOTE ET AL. nennen ein Beispiel für die negativen Auswirkungen einer hohen internen Komplexität:[321] Für den geschäftlichen Misserfolg der Porsche AG im Jahr 1992 wird u. a. die Vorgehensweise verantwortlich gemacht, die verschiedenen Modelle, die die externe Variantenvielfalt repräsentieren, mit jeweils eigenen, individuellen Bauteilen zusammenzubauen. Dadurch dass praktisch keine Modellübergreifende Wiederverwendung von Bauteilen stattgefunden hat, ist die interne Variantenvielfalt exponentiell gestiegen. Die Folge dieses exponentiellen Anstiegs, den SCHLOTE ET AL. als „Luxus“ bezeichnen, war, dass das

[318] Vgl. Schuh/Schwenk (2001)
[319] Vgl. Wildemann (1998), S. 48
[320] Vgl. Firschau et al. (2002), S. 13
[321] Vgl. Schlote et al. (1998), S. 167f

gesamte Entwicklungspotential für die interne Variantenvielfalt aufgebracht werden musste, und dass nur wenige Kapazitäten für neue Entwicklungen zur Verfügung standen.

Variantenreduktion der internen Vielfalt

Die von MENGE als Steuerungsvorgänge zur Optimierung der Variantenvielfalt bzw. zur Beherrschung der Auswirkungen variantenreicher Produktspektren bezeichneten Aktivitäten im Rahmen eines Variantenmanagements müssen nach FIRSCHAU ET AL. generell darauf abzielen, die interne Variantenvielfalt bei gleichzeitiger Bereitstellung der Markt-induzierten externen Variantenvielfalt zu minimieren.[322] Wie in Abbildung 5.3 bzw. Abbildung 5.4 dargestellt, zielen die Verfahren des Variantenmanagements (wie z. B. die Modularisierung) darauf ab, möglichst viele Produktvarianten mit möglichst wenigen, gleichen Bauteilen zusammenzubauen. Der erhoffte Vorteil dieser Verfahrensweise ist eine geringere Komplexität indem die einzelnen Bauteile der internen Vielfalt häufiger als zuvor zur Produktion der externen Vielfalt herangezogen werden. Hierdurch erhöht sich automatisch deren Losgröße, was ein Garant für geringere Einkaufspreise und Herstellungskosten ist.

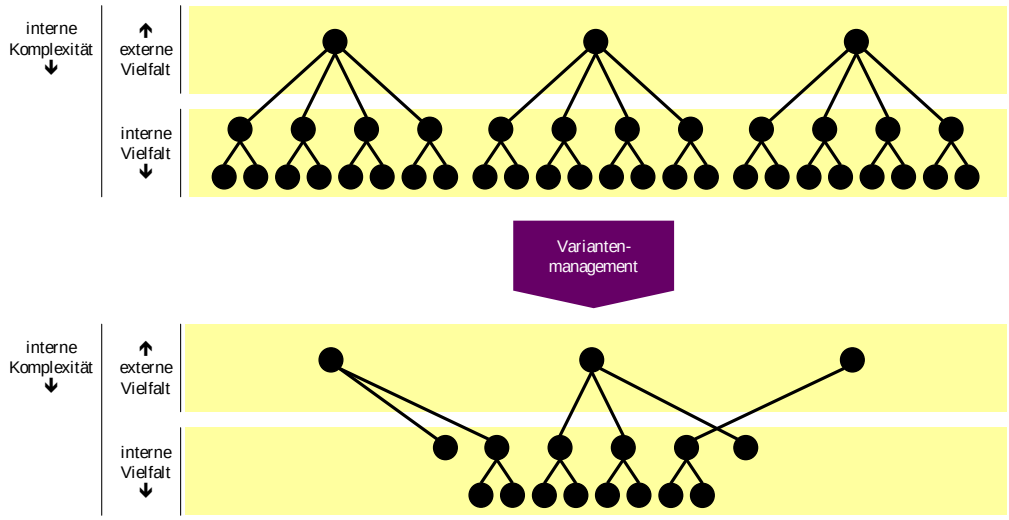


Abbildung 5.3: Reduktion der Variantenvielfalt. (Quelle: In Anlehnung an Firschau et al. (2002))

5.2.2 Modularisierung im Variantenmanagement

Modularisierung bedeutet, dass ein Gesamtsystem entweder durch eine top down-Entwicklung in Teilsysteme zerlegt wird oder durch einen bot-

[322] Vgl. Menge (2001); vgl. Firschau et al. (2002), S. 13

tom up-Ansatz aus Teilsystemen zusammengesetzt wird.[323] Das Prinzip der Modularisierung hat seinen Ursprung in der Produktgestaltung.[324] Auf Produktebene bezeichnen Module Baugruppen bzw. austauschbare Teile, deren Vormontageumfang wesentlich größer ist als ihr Einbauumfang in die nächst übergeordnete Baugruppe.

Module, Schnittstellen, Plattformen

HOFER definiert Modularisierung als „[…] die Gestaltung von Produkten und Komponenten durch die Kombination […] von funktional wie physisch abgegrenzten Bausteinen (Modulen), die durch Schnittstellen miteinander verbunden sind."[325] Wie in Abbildung 5.4 dargestellt, wird ein komplexes System (= Produkt) nach dem top down-Ansatz in Teilsysteme (= Module) zerlegt. Damit die Teilsysteme nach dem bottom up-Prinzip wieder zu einem komplexen System zusammengebaut (d. h. komponiert) werden können, werden Schnittstellen benötigt. Durch exakt definierte Schnittstellen ist es möglich, unterschiedliche Module zusammenzubauen und mit Hilfe der Unterschiedlichkeit Produktvarianz zu erzeugen. Nach RAPP bietet diese Vorgehensweise eine Möglichkeit die Vielfalt der Produkte zu erhöhen und die interne Komplexität auf dem ursprünglichen Niveau zu halten.[326] Zur Reduktion der internen Komplexität bietet sich die Reduktion der möglichen Kombinationen einzelner Module an.[327] Diese Reduktion kann zum einen durch ein Kombinationsverbot erreicht werden oder durch eine fixe Vorgabe von Kombinationen. Werden fixe Kombinationen von Modulen vorgegeben, die variabel mit anderen Modulen kombiniert werden können, so werden diese Vorgaben als eine sog. *Plattform* bezeichnet. PILLER und WARINGER verstehen unter einer Produktplattform „[…] ein Set aus zusammengehörenden Komponenten oder Teilen[…], die eine gemeinsame Struktur bilden, auf deren Basis eine Anzahl unterschiedlicher Produkte entwickelt und produziert werden können."[328] Modularisierung setzt eine Dekomposition voraus, um anschließend die dekomponierten Elemente zu höheraggregierten Modulen komponieren zu können.

---

[323] Vgl. Stahlknecht/Hasenkamp (1997), S. 292
[324] Vgl. Bund (1998), S. 566 und Baldwin/Clark (1998), S. 39ff (nach Piller/Waringer (2001), S. 37)
[325] Hofer (2001), S. 28
[326] Vgl. Rapp (1999), S. 59 (nach Hofer (2001), S. 28)
[327] Anmerkung: Welche Komplexitätsprobleme sich theoretisch ergeben können wird u. a. in Franke (1998) beschrieben.
[328] Piller/Waringer (2001), S. 64

Das Prinzip der Definition von Plattformen ist grundsätzlich von dem der Modularisierung zu unterscheiden: Während die Modularisierung eine Dekomposition eines komplexen Systems anstrebt, um hierdurch die Abhängigkeiten einzelner Teilsysteme reduzieren zu können, wird bei der Plattformdefinition eine Hierarchisierung der Teilsysteme fokussiert. Die durch die Fixierung von Modulen zu einer Plattform reduzierte interne Variantenvielfalt des Gesamtsystems wird durch die Flexibilität und Effizienz des „Ansteckens“ variabler Module an die Plattform wieder wettgemacht.[329] Antrieb für eine Plattformdefinition ist die Feststellung, dass sowohl in der Variantenmenge eines Produkttyps als auch der Menge aller Produkttypen gemeinsame Basismodule identifiziert werden können.[330]

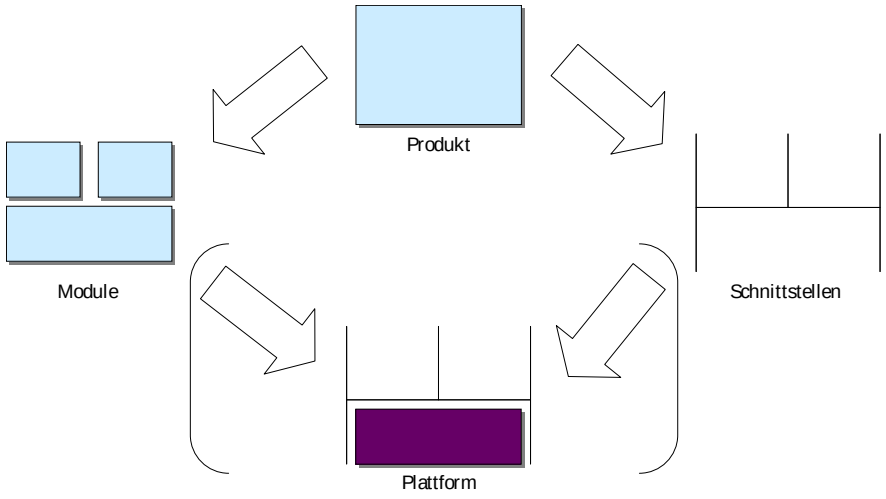


Abbildung 5.4: Die Aufteilung eines Produkts in Module und Schnittstellen sowie deren evtl. Fixierung in Form einer Plattform (in Anlehnung an Hofer (2001), S. 28).

Volkswagen

Die wohl bekannteste Anwendung des Plattformkonzepts findet sich bei der Firma Volkswagen AG:[331] Getrieben durch die externen, Marktinduzierten Komplexitätstreiber wuchs die interne Komplexität derart, dass innerhalb von 8 Jahren die Anzahl der Modelle von 28 auf 65 angestiegen ist. Um diese interne Komplexität wirtschaftlich beherrschen zu können, wurde Anfang der 90ger Jahre die Plattformstrategie eingeführt. PIECH zählt zu einer Plattform all diejenigen Einzelteile eines Automobils, die nicht Formgebend sind. Ein Vorteil der von PIECH als

[329] Vgl. Hofer (2001), S. 37
[330] Vgl. Waringer/Piller (2001), S. 64
[331] Vgl. Piech (2004), S. 198ff

„Gleichteile“ bezeichneten Module ist, dass die Module bereits mehrfach qualitätsgesichert wurden und somit ein erneuter Prüfaufwand entfällt.

Der positive Effekt der Wiederverwendung macht sich insbesondere im Bereich der Losgröße bemerkbar. Ohne Modularisierung bestellte Volkswagen 330.000 gleiche Hinterachsen für den Golf. Durch die Modularisierung und der damit impliziten Wiederverwendung konnte die Losgröße für einen Typ Hinterachsen auf über 2 Millionen vergrößert werden. Der sich hieraus ergebende Einspareffekt wird von PIECH auf über 40% geschätzt.

Aus Kundensicht betrachtet, hat die Wiederverwendung von Modulen Vorteile. Die Wiederverwendung der Gleichteile betrifft nicht die Formgebenden Elemente eines Autos. Durch die gesteigerten Losgrößen können höherwertigere Teile zu Preisen eingekauft werden, die unter dem Preisniveau geringwertigerer Teile mit kleineren Losgrößen liegen. Diese höherwertigeren Teile werden für Modellvarianten benötigt, die diese Höherwertigkeit versprechen. Dadurch dass diejenigen Gleichteile, die für höherwertigere Modellsegmente benötigt werden, in niederwertigeren Modellsegmenten wirtschaftlich eingebaut werden können, wird gleichzeitig die Qualität und Wertigkeit niederer Modellsegmente im Sinne des Kunden angehoben. Kundenorientierung ist eine Voraussetzung auf dem Weg zur Kundenzufriedenheit. Um das eigene Angebot von dem der Konkurrenz auf dem Markt zu differenzieren, versuchen Unternehmen durch das Angebot zusätzlicher Produktvarianten, sog. *Nischenprodukte*, neue Kunden zu gewinnen bzw. den Nutzen für existierende Kunden weiter zu erhöhen. Aus wirtschaftlicher Sicht betrachtet loht die Einführung dieser Produktvarianten aber nur dann, wenn sie kostendeckend produziert werden können.

Nach BLICHER[332] stellt eine nicht kostendeckende Produktion von Varianten eine Quersubventionierung dar, die die Gewinne zu lasten profitabler Varianten sinken lässt. Je höher die interne Komplexität ist, desto größer ist die Gefahr, dass nicht kostendeckend produzierte Varianten auf Grund der von ihnen verursachten Komplexitätskosten die Wirt-

[332] Vgl. Blicher/Quast (2005)

schaftlichkeit der Gesamtproduktion in Frage stellen. Die Aussage PIECHS, dass erst durch die Einführung der Plattformstrategie eine kostendeckende Produktion im Kleinwagensegment möglich wurde,[333] deutet darauf hin, dass vor der Fixierung von Modulen in Rahmen von Plattformen eine Quersubventionierung zur reinen Gewinnung von Marktanteilen stattgefunden haben muss.

Varianten-management im Sondermaschinen-bau

Die Produktion von Varianten gehört im Maschinenbau bereits zum Alltag.[334] Gemäß DECKER wandelt sich der Markt jedoch derart, dass der Kunde immer häufiger den Inhalt des Produktes bestimmt:[335] Da sich die Ansprüche und Bedürfnisse der Kunden in der Regel mit jedem Auftrag ändern, werden stetig mehr individuelle Produktvarianten entwickelt und produziert, die die Variantenvielfalt im Allgemeinen intern sowie extern vergrößern. Die Lebensdauer der Produkte von durchschnittlich 15-30 Jahren sorgt dafür, dass zusätzlich ein logistisches Problem bzgl. der Vorhaltung von Ersatzteilen auftritt.

Kunden des Maschinenbaus sind nach BLICHER ebenfalls produzierende Unternehmen, die selbst wiederum die Ausgewogenheit der internen und externen Komplexität zu bewältigen haben. Durch den weltweit gestiegenen Konkurrenzdruck hat sich der Anspruch der Kunden erhöht, die geforderte Leistung schneller als zuvor liefern zu können. Entscheidende Nebenbedingung dieser Forderung ist, dass die Leistungserstellung zu maximal den gleichen Kosten und zu minimal der gleichen Qualität erfolgen muss.[336]

Auf Grund der im Sondermaschinenbau kundenindividuell gefertigten Unikate besteht die Notwendigkeit, bestehende Produktstrukturen bzgl. ihrer internen Komplexität zu optimieren. Als wesentliches Werkzeug zur Reduktion von Herstellungskosten und zur Lieferzeitverkürzung nennt DECKER die aus der Automobilindustrie adaptierte Plattformstrategie. Die letztlich verkauften Unikate sind individuell konfigurierte und mit variablen Modulen auf Plattformen aufbauende Lösungen, die in einem Verkaufskatalog aufgelistet sind.[337]

---

[333] Vgl. Piech (2004), S. 200
[334] Vgl. Blicher/Quast (2005)
[335] Vgl. Decker (2002), S. 124ff
[336] Vgl. Blicher/Quast (2005)
[337] Vgl. Decker (2002), S. 147

Aufgabe des Variantenmanagements muss es folglich sein, eine externe Variantenvielfalt zu finden, die individuell zu konfigurierende Fertiglösungen anbietet, die zum einen die Kundenzufriedenheit sicherstellt und zum anderen die Reaktionsfähigkeit und Flexibilität der produzierenden Organisation erhöht. Erfolgreich ist die Lösung dieses Optimierungsproblems nur dann, wenn die Wirtschaftlichkeit der durch die externe Variantenvielfalt implizierten, internen Variantenvielfalt gewährleistet ist.
Nach FIRCHAU ET AL. existiert keine Branchenübergreifende Lösung für dieses Optimierungsproblem. Vielmehr müssen Lösungen Situationsbedingt gefunden werden.[338]

5.3 Methoden zur Beherrschung der Wissenskomplexität und des Wissensvolumens

5.3.1 Konfigurationswissen

Das Software-Engineering ist eine schwierige und komplexe Disziplin. Die Ursache ist nach KAHLBRANDT die Schaffung eines Problemverständnisses seitens der Software-Ingenieure. Zu jedem Problem in dieser speziellen Disziplin existieren meist mehrere Lösungen. Die Auswahl einer problemadäquaten Lösung erfordert eine detaillierte Analyse sowie die Abwägung von Vor- und Nachteilen.[339]
Die Organisation des Software-Engineering Prozesses trägt wesentlich zur Komplexitätsreduktion bei. Ein Erfolgsfaktor dieser Organisation ist die Koordination des Informationsaustausches über mehrere Ebenen, Interessengruppen und Akteure hinweg. EBERHARD kommt zu dem Schluss, dass Softwaresysteme auf dem regen Gedankenaustausch von Fachabteilungen und den Entwicklern der Systeme basieren.[340]

[338] Vgl. Firchau et al. (2002), S. 16
[339] Vgl. Kahlbrandt (1998), S. 6ff
[340] Vgl. Eberhard (1996), S. 19

Wird in einem Software-Engineering Prozess die Wiederverwendung bereits definierter Plattformen bzw. Modulen angestrebt, muss den diese Teile wiederverwendenden Architekten die Möglichkeit geboten werden, mit einfachen Mitteln die für die konkrete Problemsituation geeigneten Module auszuwählen. Die im Zusammenhang mit der Plattformstrategie bzw. der Modularisierung aufgeführte Konfiguration erfordert das Wissen bzgl. der Kombinationsmöglichkeiten und –verbote der einzelnen Elemente. Die Beherrschung des Konfigurationswissens ist nicht trivial.

In Kapitel 2 wurde festgestellt, dass die Infrastruktur eines Finanzdienstleisters komplex ist. Dementsprechend bedarf der architektonische Entwurf eines Infrastruktursystems eines entsprechend komplexen Wissens bezüglich der Regeln und Einschränkungen, die sich aus der Komplexität der Infrastruktur ergeben.

Charakteristikum des architektonischen Entwurfs einer Infrastruktur in der Finanzdienstleistungsinformatik ist die Verwendung von Hardware- und systemnahen Softwarekomponenten, die über Zulieferer beschafft werden. Somit werden innerhalb des Architekturentwurfs im Bereich Infrastruktur fertige Systemelemente zu einem Gesamtsystem funktionstechnisch komponiert. Eine derartige Verfahrensweise wird als *Konfiguration* bezeichnet. MITTAL und FRAYMANN definieren eine Konfiguration als „[…] a set of components an a description of the connections between the components and a description of the connections between the components in the set"[341]. BONGULIELMI zählt folgende Schritte eines Konfigurationsprozesses auf:[342]

- Selektion von einzelnen Komponenten.
- Definition abstrakter und spezifischer Schnittstellen zwischen den Komponenten sowie das Testen der Abhängigkeiten.
- Prüfung der Übereinstimmung zwischen den Kundenwünschen und der erzeugten Konfiguration.

Nach MÄNNISTÖ ET AL. werden in einem Konfigurationsprozess Kundenanforderungen einer technischen Spezifikation gegenübergestellt, die

[341] Mittal/Fraymann (1989), S. 1396 (nach Bongulielmi (2002), S. 45)
[342] Vgl. Bongulielmi (2002), S. 42; ähnlich auch Männistö et al. (2000)

die Bedürfnisse der jeweiligen Kunden in einer während dem Konfigurationsprozess verwendeten Sprache beschreibt.[343]

Hiermit unterscheidet sich die Konfiguration, bei der ausschließlich auf Vorhandenes zurückgegriffen wird, von der *Konstruktion*, bei der anteilsmäßig neue Komponenten entworfen werden, in der Art des Lösungsraums:[344] Konfigurierte Produkte sind nach SCHLINGHEIDER auf den Raum beschränkt, der durch die Kombinationsvielfalt der vordefinierten Strukturen und Komponenten beschrieben wird.[345]

Das Verständnis des Variantenraums, der die verschiedenen Konfigurationsmöglichkeiten eines Systems umfasst, wird durch die beiden Größen Wissenskomplexität und Wissensvolumen bestimmt. Während die Wissenskomplexität durch die Abhängigkeiten der Komponenten determiniert wird, beeinflussen die Regeln wann und wie Komponenten kombiniert werden können das Volumen des benötigten Wissens. Das Resultat einer Vernachlässigung einer wissenszentrischen Sichtweise auf den Variantenraum ist die Konzentration dieses Wissens auf wenige Mitarbeiter innerhalb eines Unternehmens.[346]

Die Beherrschung der Wissenskomplexität und des Wissensvolumens ist nach MÄNNISTÖ ET AL. ein seit Langem bekanntes Problem:[347] Die Ursache hierfür ist oftmals die Komplexität der Produktbeschreibungen. Zudem herrscht kein Konsens über die Detailtiefe, die in einem konkreten Fall für die Konfiguration von Systemelementen benötigt wird.

BARKER und O'CONNOR nennen ein Beispiel für die Wissenskomplexität bzw. das Wissensvolumen, das für ein Verständnis des Variantenraums benötigt wird:[348] Der sog. *XCON-Konfigurator* wurde in den 80ger Jahren verwendet, um Konfigurationsmöglichkeiten von Computern zu verwalten. Mit Beginn der 80ger Jahre beschrieben wenige Angestellte das Regelwerk der Konfiguration. Durch den ständig wachsenden Variantenraum bedurfte es nach einigen Jahren bis zu 12 Monate Einarbeitungszeit für neue Mitarbeiter.

---

[343] Vgl. Männistö et al. (2000)
[344] Vgl. Bongulielmi (2002), S. 48f
[345] Vgl. Schlingheider (1994) (nach Bongulielmi (2002), S. 48)
[346] Vgl. Pulkkinen et al. (1999); vgl. Bongulielmi (2002), S. 48f
[347] Vgl. Männistö et al. (1996); vgl. auch Abschnitt 1.2.2
[348] Vgl. Barker/O'Connor (1989); vgl. Männistö et al. (1996); vgl. Bongulielmi (2002), S. 49

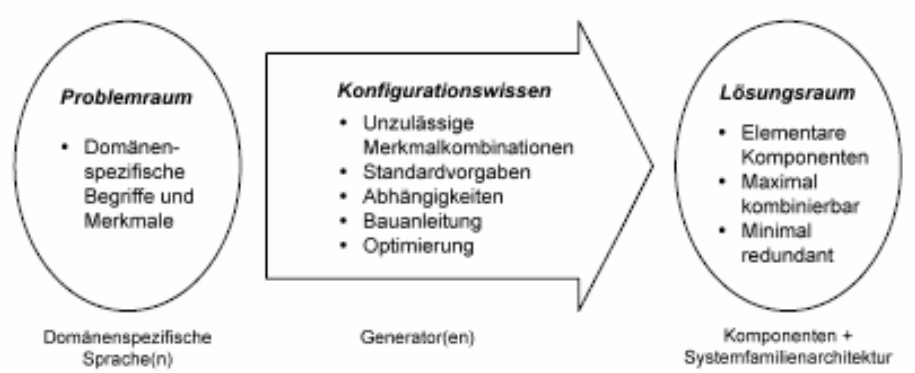


Abbildung 5.5: Generatives Domänenmodell (Quelle: Czarnecki/Eisenecker (2000), S. 6).

Generatives Domänenmodell

Im Gegensatz zur klassischen Konfiguration, bei der die Auswahl geeigneter Komponenten größtenteils manuell erfolgt, setzt das sog. *Generative Domänenmodell* auf eine teil-automatische Montage einzelner Systemkomponenten mit Hilfe eines Generators. Das Generative Domänenmodell bildet den Kern der sog. *Generativen Programmierung*, welche – ähnlich der Objektorientierten oder Strukturierten Programmierung – auf einem eigenen sog. *Paradigma*[349] beruht und nicht an eine spezifische Programmiersprache gebunden ist. CZARNECKI und EISENECKER definieren die Generative Programmierung als ein Paradigma, welches Systemfamilien derart modelliert, dass ausgehend von einer Anforderunkspezifikation mittels Konfigurationswissen aus elementaren und wieder verwendbaren Komponenten ein angepasstes Zwischen- oder Endprodukt nach Bedarf automatisch erzeugt werden kann.[350]

Der Fokus des Paradigmas auf Systemfamilien verschiebt den Betrachtungsraum einer Entwicklung von Einzelsystemen auf eine Menge von Systemen mit ähnlichen Anforderungen. Während bei der Entwicklung von Einzelsystemen ganz gezielt auf die Erfüllung der speziellen Anforderungen eines einzelnen Auftraggebers hingearbeitet wird, steht bei der Entwicklung von Systemfamilien die Entwicklung und Nutzung von elementaren Bausteinen im Vordergrund. Mit Hilfe dieser elementaren Bausteine wird das Ziel verfolgt, einzelne Teilsysteme und ganze Sys-

[349] Aus dem griechischen *pradeigma* = Modell, Muster
[350] Vgl. Czarnecki/Eisenecker (2000), S. 4ff

teme entwerfen zu können. Der Zusammenbau der einzelnen Systemteile erfolgt Generatorunterstützt.[351]

Das in Abbildung 5.5 abgedruckte Generative Domänenmodell besteht aus einem Problemraum, Konfigurationswissen und einem Lösungsraum:

- Der *Problemraum* umfasst Fachbegriffe und spezifische Merkmale, die mit Hilfe einer domänenspezifischen Sprache (engl. *Domain Specific Language*, kurz *DSL*) beschrieben sind.
- Im *Lösungsraum* sind sämtliche elementaren Bausteine enthalten, die zu Teilsystemen oder ganzen Systemen zusammengesetzt werden. Die elementaren Bausteine sind minimal redundant und maximal kombinierbar. Zusätzlich zu diesen Bausteinen enthält der Lösungsraum auch die generischen Architekturen der Systeme.
- Das *Konfigurationswissen* beschreibt unzulässige Merkmalskombinationen, Standardvorgaben, Abhängigkeiten zwischen den Merkmalen und die benötigte Bauanleitung mit Optimierungen. Die Sprache, mit der das Konfigurationswissen beschrieben wird, wird als *Configuration DSL* bezeichnet.[352]

CZARNECKI und EISENECKER trennen anwendungsorientierte Konzepte des Problemraums von Konzepten der Implementierung des Lösungsraums. Das Konfigurationswissen verbindet den Problem- mit dem Lösungsraum indem es über ein umfassendes Regelwerk zur Instanziierung gültiger Varianten verfügt. Der Zusammenhang von Problem und Lösung bezieht sich stets auf eine Domäne. Hierbei wird eine *Domäne* als ein Bereich von Fachwissen bezeichnet, der für spezifische Anforderungen ausgewählt wird, der Konzepte und Begriffe beinhaltet, die durch die Anwender des Fachbereichs verstanden werden und der das Wissen einschließt, wie Softwaresysteme oder Softwaresystemteile für diesen Fachbereich zu bauen sind.[353]

---

[351] Vgl. Czarnecki/Eisenecker (2000)
[352] Vgl. Bayer et al. (2005), S. 19ff
[353] Vgl. Czarnecki/Eisenecker (2000), S. 754

Die Konfiguration von Produkten im klassischen Sinn kann nach Ansicht von FELFERNIG ET AL. einen bedeutenden Einfluss auf die Software Entwicklung haben:[354]

- Die Komplexität der Disziplin erfordert das spezifische Fachwissen von Experten.
- Das Konfigurationswissen muss permanent adaptiert und aktualisiert werden, da sich die Komponenten und deren Zusammenhänge permanent ändern.
- Der Konfigurationsprozess nimmt wesentlich weniger Zeit in Anspruch als der Konstruktionsprozess. Konstruktion und Konfiguration müssen parallel betrieben werden.

Konfigurationswissen

Auf Basis dieser Erkenntnisse wird der Begriff *Konfigurationswissen* wie folgt definiert: *Der Variantenraum, der die verschiedenen Konfigurations-möglichkeiten einer Domäne umfasst und durch die beiden Größen Wissenskomplexität und Wissensvolumen beeinflusst wird, wird im Folgenden als Konfigurationswissen bezeichnet.*

5.3.2 Anforderungen an eine Sprache zur Abbildung von Konfigurationswissen der Infrastrukturarchitektur

Bevor mit der Selektion bzw. Kreation einer sog. *Configuration DSL*, einer Sprache zur Abbildung von Konfigurationswissen, für den Bereich der Infrastrukturarchitektur begonnen werden kann, müssen spezifische Anforderungen an eine solche Sprache gesammelt werden.

Die Charakterisierung des Problembereichs Infrastrukturarchitektur hat gezeigt, dass dessen wesentliches Problem die Komplexität der Infrastruktursysteme ist. SABIN und FREUDER stellen fest, dass Methoden, die sich mit dem Thema *Konfiguration* auseinandersetzen, im Wesentlichen die Anforderungen der Disziplinen *Wissensrepräsentation*, *Effiziente Wissensanwendung* und *zyklische Änderung der Wissendatatenbank* erfüllen müssen.[355] Die Abbildung von Konfigurationswissen erfordert eine formale und exakte Sprache, die die Elemente dieses Konfigurati-

[354] Vgl. Felfernig et al. (2000)
[355] Vgl. Sabin/Freuder (1996)

onswissens repräsentiert. Das mit einer derartigen Sprache abgebildete Konfigurationswissen muss in der Lage sein, folgende Fragen beantworten zu können:[356]

- Existiert zu der Spezifikation eine passende, gültige Konfiguration und wie sieht diese aus?
- Wie kann eine gewählte Konfiguration produziert werden, d. h. welche Prozesse und Teile werden hierzu benötigt?

Erschwerend für die Selektion bzw. Konstruktion einer geeigneten Configuration DSL ist die Änderungsrate der Konfigurationsvarianten. Treten Änderungen auf, so müssen diese erfasst und eingepflegt werden. Daher ist ein umfassender Überblick über den Zustand der jeweils aktuellen Wissensbasis vonnöten, damit eine widerspruchsfreie Datenbasis aufrechterhalten werden kann. Die Schaffung eines solchen Überblicks erfordert eine Strukturierung des Konfigurationswissens sowie eine systematische Herangehensweise.[357]

Allgemeine Anforderungen

Die Beschreibung des Domänenwissens im allgemeinen Sinn erfordert eine spezifische Sprache, deren Komplexität einen besonderen Einfluss auf die Übertragungsgeschwindigkeit und Verständlichkeit von Informationen zwischen Akteuren hat. Die Sprache, die für die Abbildung von Konfigurationswissen verwendet wird, sollte daher eine Vollständigkeit bei gleichzeitiger Einfachheit und Verständlichkeit besitzen. Die Untersuchungen von ELLS und DEWAR stellen fest, dass Zeichen grundsätzlich schneller als Text erkannt werden können.[358] Nach GITTENS können auch grafische Eigenschaften von Formen und Farben die Einfachheit und Verständlichkeit einer Sprache erhöhen.[359] Zudem können Symbole leichter erlernt und visuell schneller erfasst werden. Nach CAROLL ET AL. sind diese Eigenschaften eine Grundvoraussetzung für eine schnelle und erfolgreiche Problemlösung.[360] Besitzt die verwendete Sprache eine geringe Komplexität können Akteure, die nicht kontinuierlich mit dieser Sprache in Kontakt kommen, sich einfache Operationen leichter

[356] Vgl. Sinz (2003), S. 22
[357] Vgl. Sinz (2003), S. 22
[358] Vgl. Ells/Dewar (1979), S. 161ff
[359] Vgl. Gittens (1986)
[360] Vgl. Caroll et al. (1980)

merken[361] Im Gegensatz zu Texten, die artikuliert sein können, kann eine visuelle Darstellung genau eine Funktion verkörpern.[362] Nach ZIMMERMANN sollte eine Configuration DSL, die die Eigenschaften Einfachheit und Verständlichkeit aufweist, auf bekannten Darstellungskonzepten basieren und einfach „einfach“ sein.[363]

Die Sprache, die für die Abbildung des Konfigurationswissens selektiert bzw. konstruiert wird, muss nach SINZ folgenden Anforderungen genügen:[364]

- Leicht erlernbare, möglichst intuitiv verständliche Modellierungssprache.
- Minimierung des Dokumentations- und damit Änderungsaufwands durch Vermeidung redundanter oder nicht adäquat nutzbarer Information. Dies erfordert eine Abwägung zwischen exakter Modellierung und geringerem Dokumentationsaufwand.
- Strukturierte und Modularisierte Datenhaltung, in der das für einen bestimmten Sachverhalt relevante Wissen kompakt dargestellt ist oder einfach generiert werden kann.
- Veränderung können leicht angepasst werden: Kleine Produktänderungen erfordern auch nur kleine Änderungen an der Dokumentation.
- Algorithmisch effizient auswertbar.

Spezifische Anforderungen

Neben diesen allgemeinen Anforderungen muss eine Configuration DSL, mir der das Konfigurationswissen der Infrastrukturarchitektur abgebildet werden kann, folgende Anforderungen erfüllen:

- Infrastrukturarchitektur ist eine Disziplin des Software Engineerings. In diesem Umfeld existiert eine Reihe etablierter Sprachen, mit denen Softwaresysteme dokumentiert werden können. Sofern die Configuration DSL einer bereits bekannten und etablierten Sprache des Software Engineerings entspricht bzw. eine solche Sprache erweitert, lässt sich vermuten, dass diese Sprache eine höhere Akzeptanz besitzt als eine völlig neue Sprache.

---

[361] Vgl. Galitz (1997)
[362] Vgl. Bongulielmi (2002), S. 35ff
[363] Vgl. Zimmermann (1995)
[364] Vgl. Sinz (2003), S. 22f

- Infrastruktursysteme sind Kompositionen von Hardware- und Softwarekomponenten. Eine Configuration DSL speziell für den Bereich der Infrastrukturarchitektur muss zwischen diesen beiden Komponenten differenzieren können.
- Jede Komponente besitzt ein spezifisches Einsatzgebiet. Beispielsweise dient eine Firewall der Unterbindung eines unerlaubten Datenaustauschs zwischen zwei Rechnern. Das Einsatzgebiet entscheidet über die Selektion einer Komponente und muss abgebildet werden.
- Komponenten können unterschiedlich komponiert werden. Jede Kompositionsvariante kann der gleichen Obergruppe (z. B. „Datenbanksystem“) untergeordnet sein, die variationsspezifische Fähigkeit der jeweiligen Komposition kann jedoch unterschiedlich sein (z. B. „Skalierbares Datenbanksystem“ oder „Verfügbares Datenbanksystem“).
- Die Selektion einer Komponente kann die Selektion anderer Komponenten erfordern. Beispielsweise erfordert ein Datenbanksystem zur Lauffähigkeit ein Betriebssystem. Angenommen es existieren die Betriebssysteme BS_1 und BS_2 sowie die Datenbanksysteme $DBMS_1$ und $DBMS_2$. Sofern eine strategische Entscheidung den Betrieb des Datenbanksystems $DBMS_1$ mit dem Betriebssystem BS_2 vorsieht, muss die Configuration DSL eine Muss-Abhängigkeit zwischen $DBMS_1$ und BS_2 abbilden können. Existiert keine derartige Entscheidung müssen Kann-Abhängigkeiten zwischen jedem Datenbanksystem und jedem Betriebssystem abgebildet werden.

5.3.3 Identifikation einer Sprache zur Abbildung von Konfigurationswissen der Infrastrukturarchitektur

Die Konfiguration technischer Systeme ist eins der erfolgreichsten Gebiete der Wissensbasierten Systeme. In einer Erhebung[365] von GÜNTER und KREBS werden vier zentrale Wissenstypen identifiziert, die bei einer Konfiguration relevant sind:[366]

[365] Günter/Krebs (1999)
[366] Vgl. Krebs et al. (2002)

- Es existiere eine Menge von Domänenobjekten samt Parametern. Die Instanziierung eines Objekts impliziert die Erzeugung einer spezifischen Konzeptinstanz, wodurch ein Domänenobjekt seine Instanzen beschreibt.
- Es existiert eine Menge von Beziehungen zwischen Domänenobjekten. Die Beziehungen basieren auf taxonomischen Relationen bzgl. der Anzahl und Art der Alternativen sowie der Relevanz für die Konfiguration.
- Es existiert eine Menge von Anforderungen, die eine Konfiguration erfüllen muss.
- Der Konfigurationsprozess erfordert spezifisches Wissen für die Kontrolle desselben.

Die Konfiguration komplexer Systeme (z. B. Infrastruktursysteme) ist ein zeitintensiver Prozess. Im Umfeld der Künstlichen Intelligenz wurden zahlreiche Konfigurationsmethoden für spezifische Problemräume entwickelt. Die Mehrzahl dieser Methoden ist strukturiert indem sowohl der Konfigurationsprozess als auch die Konfigurationsmöglichkeiten auf dem zu konfigurierenden System beruhen.[367]

Eine Auswahl bestehender Konzepte, die der Abbildung von Konfigurationswissen in unterschiedlichsten Domänen dienen, lässt sich bei BONGULIELMI finden:[368]

- Die *Erzeugnisgliederung* mit einer Unterteilung in Grundstruktur, variantenspezifische Muss- und Kannteile von UNGEHEUER.[369]
- Die *Gitarrenmethode*, bei der Variantenbehaftete Merkmale in Verbindung mit den Komponenten der Produktstruktur bringen. Mit Hilfe von Entscheidungsunterlagen werden Regeln und Formeln zur Auswahl von Komponenten definiert.[370]

[367] Vgl. Kühn (2000)
[368] Vgl. Bongulielmi (2002), S. 58f
[369] Vgl. Ungeheuer (1995)
[370] Vgl. Pelikan (1976) und Müller (1982)

- Die von SCHLINGHEIDER vorgestellten *Variantenbäume*, die die Variantenbildung auf Basis der Produktstruktur mittels UND- und ODER-Verknüpfungen darstellen.[371]
- Die Konfigurationsorientierte *Teilestrukturierung*, die die Komponenten eines Produktes mit Merkmalen und Ausprägungen beschreibt, und die die Ausschlussbeziehungen zwischen den Merkmalen grafisch darstellt.[372]
- Die Konzeption des Konfigurationsmodells, das die Komponenten, die Schnittstellen zwischen den Komponenten, die funktionale Beschreibung sowie das Regelwerk umfasst und in einer Flussdiagramm-ähnlichen Form darstellt.[373]

Einen Überblick über ausgewählte, konkrete Modelle zur Abbildung von Konfigurationswissen liefert SINZ. SINZ unterscheidet hierbei zwischen abzubildenden Produktionsregeln, Ressourcenbasierten Systemen, sog. *Constraint Satisfaction Problemen* (CSPs), Graphen-basierten Systemen und Objektorientierten Modellen:[374]

- *Produktionsregeln* werden mit Hilfe von WENN-DANN-Regeln abgebildet. Diese Regeln beschreiben Komponentenabhängigkeiten oder Aktionen, die immer dann ausgeführt werden müssen, wenn bestimmte Bedingungen erfüllt sind.
- *Ressourcenbasierte Systeme* stellen die Ressource als einen Werkstoff, eine Dienstleistung oder ein abstraktes Gut in den Mittelpunkt. Hierbei muss für jede Komponente angegeben werden, welche Ressourcen diese Komponente produziert bzw. konsumiert. Das Konfigurationsproblem ist das Auffinden einer Menge von Komponenten, die zusammen alle funktionalen Anforderungen erfüllen.
- Bei *CSPs* werden einer gegebenen (endlichen) Menge von Variablen Werte aus fest vorgegebenen (endlichen) Wertemengen zugewiesen, so dass durch Relationen vorgegebene Randbedingungen erfüllt werden. Voraussetzung CSP-basierter Konfigurationsmodelle ist,

[371] Vgl. Schlingheider (1994)
[372] Vgl. Gausemeier et al. (1998)
[373] Vgl. Tiihonen et al. (1999)
[374] Vgl. Sinz (2003), S. 16ff

dass sich jede Konfiguration aus einer Menge atomarer Auswahlmöglichkeiten zusammensetzen lässt.

- *Graphen-basierte Modelle* stellen die Komponenten als Knoten eines Graphen dar. In der einfachsten Form werden UND-ODER-Verknüpfungen abgebildet.
- *Objektorientierte Modelle* können konzeptionell zwischen CSPs und Graphen-basierten Modellen angesiedelt werden. Im Unterschied zu CSPs sind die Wertebereiche der Variablen in Objektorientierten Modellen keine unstrukturierten Mengen, sondern können aus Wertebereichen zusammengesetzt sein. Komponenten werden als Objekte bezeichnet und gehören zu Klassen. Referenzen werden mittels Kompositionen abgebildet.

5.3.3.1 Allgemeine Grundlagen der Modellierung generischer Produktstrukturen

Explizite & Implizite Methoden

Nach MÄNNISTÖ ET AL. existiert eine Vielzahl an Möglichkeiten der Modellierung generischer Produktstrukturen. MÄNNISTÖ ET AL. unterteilen diese grob in *explizite* und *implizite* Methoden zur Produktstrukturbeschreibung, wobei die Adjektive „implizit“ bzw. „explizit“ die Art und Weise hervorheben, auf die eine Produktstruktur die von ihr verwendeten Komponenten identifiziert. WIELINGA und SCHREIBER bezeichnen eine solche Methode als sog. *Problem Solving Method* (PSM). PSMs lassen sich nach MÄNNISTÖ ET AL. wie folgt klassifizieren:[375]

- Eine explizite Methode setzt eine generische Produktstruktur aus den Komponenten, deren Organisation innerhalb einer Teile-Ganzes Hierarchie sowie deren Auswahlmöglichkeiten zusammen. Ein Beispiel hierfür ist nach MÄNNISTÖ ET AL. ein Mountainbike M, das entweder einen Rahmen R_1 oder einen Rahmen R_2 besitzt.
- Eine implizite Methode beschreibt das Wissen über die Kompatibilität von Komponenten, die Konnektivität dieser Komponenten sowie ein Regelwerk. Ein Beispiel für eine implizite Methode ist die Beschreibung eines Bildschirms, der ein hoch auflösendes Videosignal benö-

[375] Vgl. Männistö et al. (1996)

tigt. Eine andere Komponente, z. B. eine Grafikkarte, beschreibt, dass sie ein solches hoch auflösendes Signal produzieren kann. Eine weitere Komponente, ein T-Adapter, beschreibt, dass sie in der Lage ist ein eingehendes, hoch auflösendes Videosignal an zwei Ausgänge gleichzeitig senden kann. Eine explizite Beschreibung eines der Komponenten als Arbeitsgruppe existiert nicht. Vielmehr versetzt die Beschreibung einen Akteur in die Lage, bei Bedarf eine Grafikkarte, einen T-Adapter und zwei Monitore zu einer Arbeitsgruppe zu komponieren.

Eine scharfe Trennung beider Ansätze in einem Modell existiert nicht. Daher ist es möglich, die Kombination einer impliziten und einer expliziten Methode in einem Modell zu verwenden. Ein Beispiel für den Bedarf einer derartigen Kombination ist nach MÄNNISTÖ ET AL. die Beschreibung eines Produkts P, das sich explizit aus den zwei Komponenten A und B zusammensetzt. A und B besitzen jeweils die beiden Komponentenvarianten A’ und A’’ sowie B’ und B’’. Gleichzeitig kann ein implizites Konfigurationswissen beschrieben werden, mit dem bspw. eine Kombination von A’’ und B’ untersagt wird.

In einem rein expliziten Modell kann die Selektion einer variantenreichen Komponente ohne Beachtung anderer Komponenten erfolgen. Der Umstand, dass zwischen Komponenten diverse Abhängigkeiten existieren, kann dazu führen, dass der Konfigurationsprozess einen Kontext benötigt, über den eine Komponentenselektion gesteuert werden kann. MÄNNISTÖ verdeutlicht diesen Bedarf am Beispiel einer Komponente Motor, die in den Varianten M220V und M110V verfügbar ist. Jede dieser beiden Varianten benötigt einen spezifischen Schalter S220V und S110V. Der Kontext für die Selektion kann die Anforderung *Voltzahl* sein, über die entweder die Kombination (M220V, S220V) oder (M110V, S110V) selektiert werden kann. Alternativ könnte eine implizite Methode eine gezielte Selektion der Form: *(WENN Motor = M_{220V} DANN Schalter = S_{220V}) ODER (WENN Motor = M_{110V} DANN Schalter = S_{110V})*. Eine andere mögliche Selektion bestände in der Form: *(WENN Motor = M_{220V} DANN Schalter ≠ S_{110V}) ODER (WENN Motor = M_{110V} DANN Schalter ≠ S_{220V})*. Hieraus folgt, dass explizite Methoden direkter signalisieren, welche Komponenten in einer Konfiguration selektiert

werden müssen, wohingegen implizite Methoden Bedingungen beschreiben, die von Konfigurationen erfüllt werden müssen.[376]

Configuration Design

Die grafische Repräsentation der in einem Konfigurationsprozess selektierbaren Komponenten bzw. deren kombinatorische Anordnung zu höher aggregierten Teilsystemen, wird in der Literatur als *Configuration Design* bezeichnet.[377] WIELINGA und SCHREIBER definieren diese Disziplin wie folgt: „Configuration design is a form of design where a set of pre-defined components is given and an assembly of selected components is sought that satisfies a set of requirements and obeys a set of constraints."[378]

Die Vielzahl kombinatorischer Möglichkeiten der in einem Konfigurationsprozess verwendbaren Komponenten erfordert spezifische PSMs, mit denen das benötigte Konfigurationswissen abgebildet werden kann. Die Mehrzahl der bekannten PSMs[379] stellen große Anforderungen an das vorhandene Konfigurationswissen. PSMs können hierbei wie folgt klassifiziert werden:[380]

- *Case-Based Methods*: Diese Form der PSMs basiert auf der Annahme, dass das Konfigurationswissen explizit repräsentiert werden kann. Das Konfigurationswissen kann sowohl auf bereits umgesetzten Lösungen aufbauen als auch neues Wissen durch Domänenexperten integrieren. Eine derartige Methode, die von WIELINGA und SCHREIBER als *Select & Verify* bezeichnet wird, wird eingesetzt, sofern eine Menge mit Lösungsmöglichkeiten existiert, aus der eine Lösung ausgewählt werden kann, um anschließend ihre Eignung zur Problemlösungen durch einen Vergleich der Anforderungen an die Lösung mit den Eigenschaften der Lösung überprüfen zu können. Ein derartiges Vorgehen erfordert zusätzliches Wissen über die Selektionspfade, die innerhalb eines Konfigurationsprozesses eingeschlagen werden können.

[376] Vgl. Männistö et al. (1996)
[377] Vgl. Mittal/Frayman (1989) und Yu/MacCallum (1995)
[378] Wielinga/Schreiber (1997)
[379] WIELINGA und SCHREIBER zählen hierzu die Methoden Löckenhoff/Messer (1994), Maher (1990) und Puppe (1993)
[380] Vgl. Wielinga/Schreiber (1997)

- *Propose, Critique and Modify Methods*: Bei dieser Methodik wird eine initiale Konfiguration vorgeschlagen. Anschließend wird diese Konfiguration auf die Erfüllung der an die Lösung gestellten Anforderungen geprüft. Wird hierbei eine Verletzung der Gültigkeitsregeln festgestellt, werden Modifikationen zur Behebung dieses Konflikts verwendet.
- *Hierarchical Configuration*: Eine grundsätzliche Ausprägung dieser Klasse von PSMs ist die Konfiguration über einen UND/ODER Graphen. Hierbei wird ein Ziel in alternative Substrukturen dekomponiert, von denen jede Substruktur zur Erreichung des Ziels beiträgt. Sog. *Design-Pläne*, die eine Wissensstruktur abbilden, mit der spezifische Anforderungen erfüllt oder ein Teilprobleme gelöst werden können, unterstützen diese Methodik.

Fazit

Wie die Unterteilungen der Methoden zeigen, kann die Abbildung des Konfigurationswissens einer spezifischen Domäne auf unterschiedlichste Art und Weise erfolgen. WIELINGA und SCHREIBER stellen fest, dass die existierenden Ansätze sich grundsätzlich unterscheiden. Die Vielfältigkeit der Praxisprobleme zwingt daher die Autoren von PSMs oftmals dazu, eigene, auf die Problemdomäne zugeschnittene Verfahren zu entwickeln.

5.3.3.2 UML als domänenspezifische Sprache zur Abbildung von Konfigurationswissen

5.3.3.2.1 Spezifische Eigenschaften der Objektorientierung

Die Idee der Objektorientierung ist über 30 Jahre alt. Während Publikationen über die *Objektorientierte Programmierung* bereits in den 70ger Jahren erschienen sind, erschienen die ersten Bücher über *Objektorientierte Analyse- und Designmethoden* erst mit Beginn der 90ger Jahre. Zu den Methoden gehören nach ÖSTEREICH unter anderem die Ansätze von BOOCH, COAD und YOURDON sowie RUMBAUGH ET AL., deren Methoden auf bestimmte Anwendungsbereiche begrenzt sind. Im Jahr 1997 wurde die sog. *Unified Modeling Language* (UML) von der sog. *Object Management Group* (OMG) verabschiedet, die sich seitdem zu einem Quasi-Standard für Notationen im Umfeld der Objektorientierten Analyse- und Designmethoden entwickelt hat.[381] ROBAK ET AL. stellen fest, dass die UML zur dominantesten Modellierungssprache im Umfeld der Softwareentwicklung avanciert ist.[382] Nach BOOCH repräsentiert die UML einen generellen Vorschlag für eine grafisch notierte Sprache zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Elementen, die im Umfeld von Software-basierten Systemen verwendet werden.[383]

Objektorientierung basiert auf einem Paradigma bei dem der Mensch in den Mittelpunkt gerückt wird. Hierbei wird eine Technologiezentrische durch eine anthropozentrische Sicht auf Systeme ersetzt. Wesentliches Ziel des Objektorientierten Paradigmas ist die Schaffung einer Objektivität. Nach HABERMAS entsteht Objektivität dadurch, dass sie für eine Gemeinschaft sprach- und handlungsfähiger Subjekte als ein- und dieselbe Welt gilt. Eine Bedingung hierfür ist, dass sich kommunikativ handelnde Subjekte miteinander über das verständigen, was in der Welt vorkommt bzw. in ihr bewirkt werden soll.[384]

---

[381] Vgl. Östereich (1998), S. 19ff bzw. Booch (1991), Rumbaugh et al. (1991) und Coad/Yourdon (1991)
[382] Vgl. Robak et al. (2002)
[383] Vgl. Booch et al. (1999)
[384] Habermas (1987), S. 32 (nach Östereich (1998), S. 27)

ÖSTEREICH hebt hervor, dass eine Objektorientierte Vorgehensweise ein Bewusstsein kommunikativer Rationalität schafft und somit eine andere Form der Weltanschauung einleitet. Im Gegensatz zu nicht-Objektorientierten Verfahren, wird mit der Objektorientierung eine konsensuelle Kommunikation forciert, da die Objektorientierung durch ihren Aktivismus die Beschränktheit der menschlichen Kommunikationsmöglichkeiten berücksichtigt.[385]

Nach Ansicht von JECKLE und KERN-BAUSCH hat die zunehmende Komplexität moderner Systeme die Akzeptanz des Objektorientierten Paradigmas gefördert. Die Abstraktionsmittel dieses Paradigmas erlauben es, große Systeme übersichtlich zu gliedern. Mit zunehmender Größe und Komplexität der Systeme wurde die Komplexitätsreduktion auf der Modellebene durch Komplexitätssteigerung auf der Sprachebene erkauft. JECKLE und KERN-BAUSCH folgern hieraus, dass die Möglichkeit der Einbeziehung zukünftiger Anwender des entstehenden Systems in den Entwurfsprozess bei Verwendung Objektorientierter Modellierungs-sprachen auf Grund der Sprachkomplexität erschwert wird. Das Resultat dieser Entwicklung ist eine Qualitätssenkung des entstehenden Objektorientierten Systems, da das vorhandene Fachwissen nicht direkt in den Modellierungsprozess einfließen kann sondern eines zusätzlichen interpretierenden Schrittes bedarf. Darüber hinaus hat speziell die UML mittlerweile ein Komplexitätsniveau erreicht, das die korrekte Abbildung großer Systeme erschwert. Sowohl Entwicklungsprozesse als auch -werkzeuge sind daher gezwungen zunächst den Bruch zwischen Fachleuten und Systementwicklern zu überbrücken und im Weiteren die Umsetzung des erfassten Wissens in adäquate IT-Strukturen zu unterstützen.[386]

5.3.3.2.2 Grundlagen der Wiederverwendung

Ein wesentliches Instrument der Objektorientierung ist die Wiederverwendung bereits definierter Konstrukte. Wiederverwendung ist ein vielfältiges und komplexes Themengebiet: MAYER definiert Wiederverwendung als wiederholte Nutzung von früher erstellten und getesteten

[385] Vgl. Östereich (1998), S. 27
[386] Vgl. Jeckle/Kern-Bausch (2000)

Komponenten.[387] Nach WEBER ist die Wiederverwendbarkeit von Produktkonzepten, von Produkten, aber auch von Verfahrensweisen ein zentrales technologisches Konzept, mit deren Hilfe sowohl eine Kostensenkung als auch eine Qualitätsverbesserung der Produkte angestrebt wird.[388] ROTHE charakterisiert die Wiederverwendung innerhalb des Softwareentwicklungsprozesses als eine nicht gelöste Aufgabenstellung, die weniger durch technische oder methodische Wiederverwendungs-konzepte gelöst werden kann, sondern eher ein organisatorisches, technisches, fachliches und kulturelles Gesamtkonzept erfordert.[389]

STÜTZLE unterscheidet im Umfeld von Softwarekomponenten zwischen geplanter Wiederverwendbarkeit und ungeplanter Wiederverwendung:[390] Während Wiederverwendung das Ziel hat, Komponenten wieder zu verwenden, die für eine Wiederverwendung nicht konstruiert wurden, ist die Wiederverwendbarkeit eine Eigenschaft derjenigen Komponenten, die für einen geplanten Wiedereinsatz konzipiert wurden.

Letztendliches Ziel der Wiederverwendung ist ein Entwicklungsprozess, in dem Systeme aus bereits bestehenden Komponenten durch Konfigurieren aufgebaut werden und nur noch wenige bis überhaupt keine Systeme von Architekten entworfen werden müssen. Ein derartiger Entwurf von Architekturen verspricht eine ganze Reihe von Vorteilen:[391]

- *erhöhte Produktivität*: Da bereits fertige Komponenten verwendet werden, müssen insgesamt weniger neue Komponenten für das System entworfen werden.
- *höhere Flexibilität*: Durch die Verwendung von Komponenten kann schneller und flexibler auf veränderte Anforderungen und Kundenwünsche reagiert werden.
- *erhöhte Konkurrenzfähigkeit*: Der verringerte Anteil an Neuentwicklungen in einem Projekt senkt die Projektkosten.

[387] Vgl. Mayer (1988)
[388] Vgl. Weber (1992), S. 72
[389] Vgl. Rothe (2004), S. 64
[390] Vgl. Stützle (2002), S. 11ff; ähnlich auch Balzert (1998), S. 639ff
[391] Vgl. Dujmovic (1997)

- *Qualitätsverbesserungen*: Durch die Wiederverwendung von getesteten und qualitativ hochwertigen Komponenten kann die Qualität der neu entstehenden Systeme positiv beeinflusst werden. PIECH beschreibt, dass durch die Wiederverwendung sog. *Gleichteile* über mehrere Fahrzeugklassen hinweg in Fahrzeugen der niederen Preissegmente Bauteile höherer Preissegmente eingebaut werden. Durch die hiermit verbundene Erhöhung der Losgröße können die Preise für derartige Komponenten in solchem Maße gesenkt werden, dass die Produktionskosten insgesamt niedriger sind, als ohne die Verwendung von Gleichteilen:[392] „Einen vorläufigen Höhepunkt an Gleichteile-Effizienz erreichten wir mit der Polo-Generation von 2001. Neue Hinterachse und besonders feinfühlige Lenkung sind hilfreich für ein Fahrgefühl, das es in dieser Klasse noch nicht gab.“[393]

5.3.3.2.3 Muster als Instrument der Wiederverwendung

Ein Instrument der Wiederverwendung im Umfeld konstruierender Disziplinen wie dem Software Engineering oder der Architektur sind die sog. *Muster*. Die Idee der Musterdefinition kann auf den Architekten Christopher Alexander zurückgeführt werden, der in den 70ger Jahren verschiedene Entwurfsmuster für die klassische Gebäudearchitektur definiert hat. In Anlehnung an LEA hat Alexander der Objektorientierten Softwareentwicklung die Inspiration zur Definition von Lösungsideen für spezifische Entwurfsprobleme geliefert.[394]

Die Informatik-Literatur identifiziert vorwiegend drei Typen von Mustern:[395]

- Ein *Architekturmuster* beschreibt eine Lösung für ein Problem der IT-Architektur. Im Gegensatz zu Entwurfsmustern oder Idiomen, die auf die Entwicklung von Software-Programmen ausgerichtet sind, sind Architekturmuster generellerer Art.
- Ein *Entwurfsmuster* (engl. „design pattern“) beschreibt eine Lösung für ein Problem der Software-Architektur. Entwurfsmuster geben die

[392] Vgl. Piech (2004), S. 198ff
[393] Piech (2004), S. 203
[394] Vgl. Lea (1998)
[395] Vgl. Östereich (1998), S. 63ff; Gamma et al. (1995); Gamma (1991); Buschmann et al. (1998)

Struktur für Programmcode vor, ohne Bezug zu einer spezifischen objektorientierten Programmiersprache aufzuweisen.

> Ein *Idiom*[396] ist eine Lösungsbeschreibung für ein spezifisches Programmiersprachenproblem.

ÖSTEREICH umschreibt ein Muster als eine bewährte Lösungsidee für immer wiederkehrende Entwurfsprobleme.[397] Sofern Muster dokumentiert werden, können Sie für spätere Problemstellungen auf ihre Anwendbarkeit überprüft werden. Kann ein dokumentiertes Muster auf Grund seiner Problemadäquanz wiederverwendet werden, reduziert sich die Zeit für einen Neuentwurf auf Null. In diesem Fall entsteht jedoch ein zeitlicher Aufwand für die Adaption des konkreten Musters auf das Problem. Die Adaption wird durch die innerhalb der im Muster enthaltenen Varianz bestimmt, die sich beispielsweise aus der Variabilität der Anzahl der einsetzbaren Komponenten ergibt. Die Adaption eines vorhandenen Musters auf ein Problem ist somit eine Konfiguration der im Muster enthaltenen Elemente.

Anforderungen an Muster

LEA formuliert folgende Eigenschaften, die ein Muster aufweisen sollte:[398]

> *Kapselung*: Jedes Muster kapselt in Anlehnung an PARNAS[399] und NORMAN[400] ein spezifisches Problem und liefert eine adäquate Lösung für dieses Problem. Muster sind unabhängig, spezifisch und präzise formuliert. Sie machen deutlich, für welche Probleme sie Lösungen anbieten.

> *Erzeugbarkeit*: Jedem Muster ist einen selbständiger Prozess zur Realisation der vom Muster beschriebenen Lösung zugeordnet. Diese Eigenschaft beruht auf der Annahme, dass Muster von anderen Ak-teuren als ihren Autoren eingesetzt werden und somit spezifisches Wissen bei diesen Akteuren nicht vorausgesetzt werden kann. LEA vergleicht den Gebrauch eines Musters wie folgt: Ein Experte, der ein Muster verwendet, sollte dies ähnlich einem Rezept verwen-

[396] Aus dem Griechischen: *Eigentümlichkeit*, *Besonderheit*
[397] Vgl. Östereich (1998), S. 63
[398] Vgl. Lea (1998)
[399] Vgl. Parnas (1972)
[400] Vgl. Norman (1988)

den, das ein Chefkoch verwendet – zur Erzeugung einer persönlichen Vision einer spezifischen Lösung bei gleichzeitiger Zugabe kritischer Zutaten in wohlproportionierten Mengen.

- *Gleichgewicht*: Jedes Muster identifiziert einen Lösungsraum, der eine Invariante[401] enthält, die den Konflikt zwischen Möglichkeiten und Grenzen der vom Muster beschriebenen Lösung minimiert. Wird ein Muster angewendet, sorgt das Gleichgewicht in jedem Designschritt für nachvollziehbare Entscheidungswege.
- *Abstraktion*: Muster sind Abstraktionen empirischer Erfahrungen und alltäglichen Wissens. Sie sind innerhalb ihres Einsatzgebietes allgemein gehalten. Sie sind aber notwendiger Weise nicht universell. Die Erzeugung eines Musters ist ein iterativer, sozialer Prozess, bei dem Erfahrungen und Wissen gesammelt, ausgetauscht und gestärkt werden.
- *Offenheit*: Muster sollten ein gewisses Detailmaß enthalten. Muster werden innerhalb eines Entwicklungsprozesses verwendet, indem die Anforderungen an eine Lösung den vom jeweiligen Muster gebotenen Eigenschaften gegenübergestellt werden. Muster können auch andere Muster enthalten. Zwischen den Mustern entstehen Hierarchien.
- *Komponierbarkeit*: Durch die Hierarchie zwischen Mustern entstehen Relationen. Die Komposition einzelner Muster zu einem jeweilig hierarchisch höher angesiedelten Muster ist eine Grundvoraussetzung für die Wiederverwendbarkeit von Teillösungen.

Prozess

Muster enthalten nach LEA rezeptartige Informationen bezüglich ihrer Anwendung. Musterbasierte Entwurfsaktivitäten sind resistent gegenüber einer Anpassung an lineare Entwicklungsprozesse und eröffnen neue Möglichkeiten der Konstruktion von Prozessmodellen. Ein Prozess, der der Konstruktion von Mustern dient, besitzt folgende Eigenschaften:[402]

---

[401] Anmerkung: Eine Invariante ist eine Bedingung, die in bestimmten Zuständen eines Systems erfüllt sein muss.
[402] Vgl. Lea (1998)

- *Kollektive Entwicklung*: Entwicklung ist ein sozialer Prozess. Die Lösung zukunftsfähiger Probleme erfordert die die Teilnahme diverser Akteure (Anwender, Richtlinienbeauftragte usw.) an dem Prozess. Die Akteure müssen allgemeine und individuelle Anforderungen abwägen und eine Entscheidungsbasis für die spätere Wiederverwendung der Muster aufbauen.
- *Partipitionierter Entwurf*: Anwender können dabei helfen, den Entwurf von Mustern im Vorfeld in eine sinnvolle Richtung zu lenken, indem sie ihre Anforderungen spezifizieren. Ein Architekt, der ein Muster entwirft, sollte Anforderungen eines Anwenders nur dann missachten, sofern sein wissen in einer spezifischen Domäne wesentlich größer ist als das des Anwenders.
- *Verantwortung*: Architekten tragen die finanzielle Verantwortung für die Konsequenzen ihrer Entscheidungen. Hieraus ergibt sich eine autoritäre und verantwortungsvolle Rolle innerhalb des Entwicklungsprozesses.
- *Dezentralisation*: Durch die Aufteilung der Aufgaben auf einzelne, dezentrale Aufgabenträger wird der Prozess positiv beeinflusst.
- *Integration von Rollen*: Die Zuweisung von Rollen an Personen ist nicht fix. Die Rolle eines Architekten kann (und muss) auch ein „Handwerker“[403] übernehmen und umgekehrt, da Letzterer näher an den Problemen steht als andere.
- *Integration von Aktivitäten*: Der Konstruktionsprozess unter Verwendung von Mustern besteht aus den Teilaktivitäten *kontinuierliche Analyse und Fehlerkorrektur* sowie *Abstimmung von Einzelnem und dem Ganzen*.
- *Schrittweise Konstruktion*: Ein Muster ist das Ergebnis eines schrittweise durchgeführten Entwurfs. Der Entwurf eines Musters kann den Entwurf eines noch nicht entworfenen Musters erfordern. Ebenso können bereits entworfene Muster in anderen Mustern wieder verwendet werden. Dies erfordert Kreativität.

[403] Anmerkung: Lea nutzt im Englischen den Begriff „Builder“. In der Informatik wird diese Rolle von Softwareentwicklern oder technischen Spezialisten ausgefüllt.

5.3.3.2.4 Abstraktionsmittel der UML

Beim Entwurf eines Systems werden Sachverhalte abgebildet. Die Beschreibung dieser Sachverhalte erfordert einen gewissen Grad der Abstraktion. Die UML liefert primär zwei Abstraktionsmittel:[404]

- Eine *Part-of Beziehung* (alias *Aggregation*) zwischen zwei Modellelementen fasst mehrere Einzelteile zu einem Ganzen zusammen, so dass die Einzelteile Teil eines Ganzen sind. Eine spezielle Ausprägung dieses Beziehungstyps ist eine *Komposition*, bei der die Einzelteile existenzabhängig von dem zusammenfassenden Ganzen sind. Beispiel: Ein Auto besteht aus Motor, Karosserie, Sitzen etc.
- Eine *Is-a Beziehung* (alias *Vererbung*) zwischen zwei Modellelementen fasst mehrere Arten oder Varianten von Einzelteilen unter einem Begriff zusammen. Ein Einzelteil „ist eine“ Variante eines anderen Einzelteils. Beispiel: Ein Cabriolet ist ein Auto. Ein Auto ist ein Fahrzeug. Demzufolge ist ein Cabriolet auch ein Fahrzeug.

Zum Verständnis der weiteren Ausführungen sind folgende Definitionen von Nöten:[405]

- Eine *Klasse* ist die Definition der Attribute, Operationen und der Semantik einer Menge von Objekten. Alle Objekte, die einer Klasse zugeordnet werden können, entsprechen dieser Definition.
- Eine *Instanz* steht synonym für ein Objekt. Ein *Objekt* ist eine konkret vorhandene und agierende Einheit mit eigener Identität und definierten Grenzen. Es kapselt einen Zustand und ein Verhalten und ist Instanz einer Klasse. Das definierte Verhalten gilt für alle Instanzen einer Klasse gleichermaßen. Ebenso verhält es sich mit der Struktur der Attribute. Die Attributwerte sind jedoch Objekt-individuell ausgeprägt.
- Ein *Stereotyp* ist eine projekt-, unternehmens- oder methodenspezifische Erweiterung vorhandener Modellelemente des Meta-Modells der UML. Ein Stereotyp kennzeichnet ein Modellelement und beeinflusst das Modellelement semantisch. BOOCH beschreibt einen Stereotyp wie folgt: „An extension of the vocabulary of the UML,

[404] Vgl. Östereich (1998), S. 29
[405] Vgl. Östereich (1998), S. 337ff

which allows you to create new kinds of building blocks that are derived from existing ones but are specific to your problem."[406] Nach MEHNER und WAGNER ist ein Stereotyp ein Erweiterungsmechanismus der UML, mit dessen Hilfe neue Sprach-elemente von gegebenen UML-Modellierungselementen abgeleitet und an einen spezifischen Anwendungsbereich angepasst werden können.[407]

- Eine *Meta-Klasse* ist eine Klasse, deren Instanzen wiederum Klassen sind. Eine Meta-Klasse ist Teil eines Meta-Modells.
- Ein *Meta-Modell* ist ein Modell, das eine Sprache definiert, mit der ein Modell definiert werden kann.

5.3.3.2.5 Meta-Modellierung

Die mit dem objektorientierten Paradigma verbundenen Konzepte (z. B. Klassen, Objekte, Vererbung, Polymorphismus und Relationen) werden von WEBSTER als Kernkonzepte der Wiederverwendung, Adaptierbarkeit und Komplexitätsbeherrschung bezeichnet.[408]

ROBAK ET AL. verweisen jedoch darauf, dass das primäre Ziel des objektorientierten Ansatzes – und somit der UML – die Entwicklung eines Systems zu einem Zeitpunkt ist. Somit unterstützt die Objektorientierung weder die gezielte Wiederverwendung von Entwürfen noch eine Entwicklung mit Wiederverwendung. Die fehlende Fähigkeit der UML ist die Unterscheidung zwischen der Variabilität verschiedener Systeme und spezifischen Versionen ein- und desselben Systems. Sofern diese Fähigkeit während des Systementwurfs erforderlich ist, muss ein entsprechendes UML-konformes Meta-Modell entworfen werden.[409]

Modell <> Meta-Modell

In der UML 2.0 wird grundsätzlich zwischen einem Meta-Modell und einem Modell unterschieden. Ein Modell kann als Instanz eines Meta-Modells fungieren. Gleichzeitig kann ein Modell wiederum ein Meta-Modell für andere Modelle darstellen. Ein Modell enthält Elemente. Diese sind Instanzen der Elemente eines Meta-Modells, welche als *Meta-Klassen* bezeichnet werden. Die typische Aufgabe eines Meta-Modells

[406] Booch et al. (1999)
[407] Vgl. Mehner/Wagner (2000)
[408] Vgl. Webster (1995)
[409] Vgl. Robak et al. (2002)

ist die Definition einer Semantik, mit der die Elemente eines Modells instanziiert werden können.[410]

Ein Beispiel für eine derartige Instanziierung ist in Abbildung 5.6 abgedruckt: Elemente des Meta-Modells sind die Typen *Class* und *Association*. In einem konkreten Modell, einem sog. *User Model*, werden die Klassen *Car* und *Person* auf Basis des Meta-Klasse *Class* instanziiert. Die Beziehung *Person.car* zwischen beiden Klassen ist wiederum eine Instanz der Meta-Klasse *Association*.

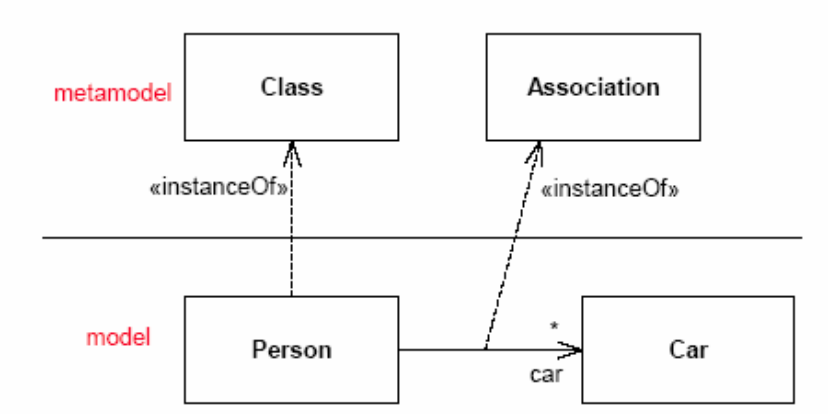


Abbildung 5.6: Ein Modell ist eine Instanziierung eines Meta-Modells (Quelle: OMG (2004), S. 17).

Layer

Die Beschreibung von Sprachen besitzt üblicherweise drei sog. *Layer*: Die Sprachspezifikation in Form des Meta-Modells, die konkrete Spezifikation in Form des Modells und die Objekte dieses Modells. Diese Form der Strukturierung kann mehrfach rekursiv angewendet werden, so dass theoretisch unendlich viele Meta-Layer resultieren können. Das Meta-Modell eines als Meta-Modell fungierenden Modells wird als Meta-Meta-Modell bezeichnet. Je größer der Abstraktionsgrad eines Meta-Modells ist, desto größer ist der Index des Modells. In der UML 2.0 besitzt der konkreteste Modell-Layer den Index M0. Die nächste Abstraktionsebene von M0 besitzt den Index M1. Mit jeder weiteren Abstraktionsebene inkrementiert der Index.[411]

Ein Beispiel für die Zusammenhänge der Modell-Layer ist in Abbildung 5.7 abgedruckt. In diesem Beispiel werden insgesamt 4 Modell-Layer verwendet, wobei eine allgemeine Beschränkung auf eine bestimmte

[410] Vgl. OMG (2004), S. 17ff
[411] Vgl. OMG (2004), S. 18ff

Anzahl an Layern nicht existiert. Ausgangsbasis des Modells ist das sog. *Meta Object Facility* (MOF), die die Meta-Ebene für die eigentliche Sprache UML darstellt und die Ausgangsbasis für Meta-Modelle im Allgemeinen bietet.[412]

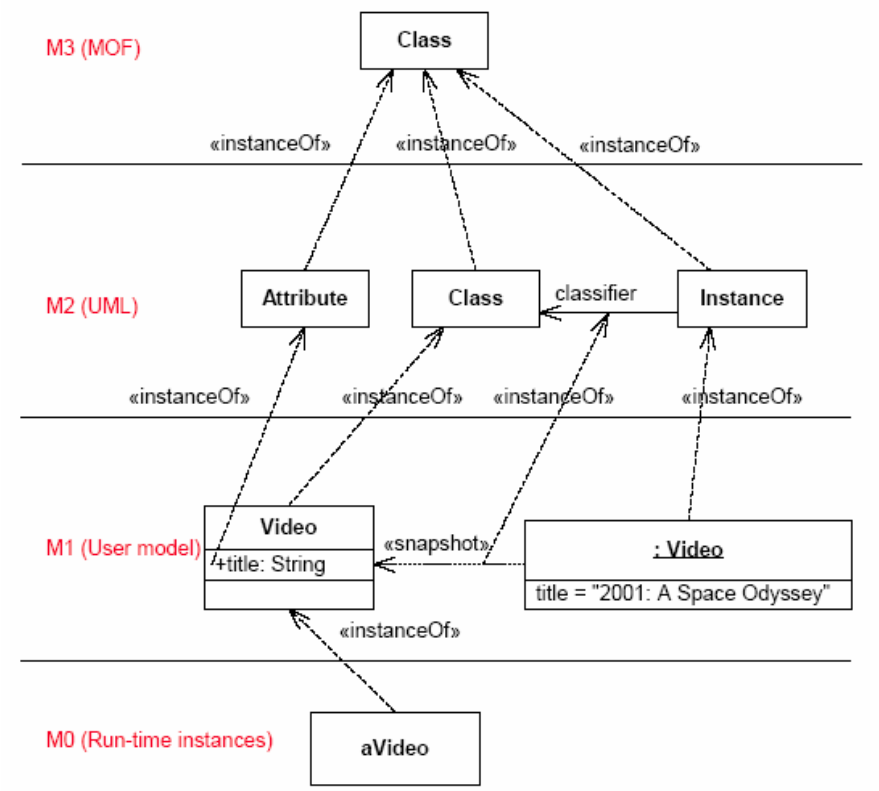


Abbildung 5.7: Zusammenhang der Meta-Modelle (Quelle: OMG (2004), S. 19).

5.3.3.2.6 Eignung der UML zur Abbildung von Konfigurationswissen

Modell

FELFERNIG ET AL. attestieren der UML eine Eignung zur Abbildung von domänenspezifischem Konfigurationswissen. Den Grund sehen die Autoren darin, dass die UML als eine sehr exakte und formale Sprache in der Lage ist Beschreibungen von Problem- und Lösungsräumen in Form von Konfigurationswissen abzubilden.[413]

Die Verringerung und Handhabung der internen Komplexität, die aus den externen Komplexitätstreibern kürzere Produktzyklen, geringere Budgets und gesteigerte Kundenbedürfnisse resultiert, stellt eine große Herausforderung für den Entwicklungsprozess dar. Ein erfolgreicher Ansatz, mit dem eine solche Herausforderung gemeistert werden kann, ist die Entwicklung von Wissensbasierten Systemen, die das Wissen

[412] Vgl. Felfernig et al. (2001)
[413] Vgl. Felfernig et al. (2000); vgl. auch Ardissino et al. (2001)

mit Hilfe von domänenspezifischen und formalen Sprachen abbilden.[414] BAYER ET AL. bezeichneten eine derartige Sprache allgemein DSL bzw. als Cofiguration DSL, sofern die Sprache in einem Konfigurationssystem verwendet wird.[415]

Die Verwendung der UML als Configuration DSL ist jedoch selten und wird erstmals in Felfernig et al. (2000) diskutiert.

Nach Ansicht von FELFERNIG ET AL. besitzt die UML zwei entscheidende Eigenschaften, die sie für eine Eignung innerhalb des Konfigurationsprozesses prädestinieren:[416]

- Zum einen ist die UML eine weltweit akzeptierte und eingesetzte Sprache zur Dokumentation von Softwaresystemen.
- Zum anderen hat sich gezeigt, dass Entwürfe, die mit der UML beschrieben wurden, als Diskussionsbasis zwischen Domänenexperten besonders geeignet sind.

FELFERNIG ET AL. konzeptionieren ein UML-Modell, das eine generische Produktstruktur bzw. alle möglichen Produktvarianten repräsentiert:[417] Die Menge der Produktvarianten wird durch eine Menge mit Bedingungen, die sich aus Kundenanforderungen, technischen Restriktionen, ökonomischen Einflussfaktoren und Restriktionen des Herstellungsprozesses ergeben, größentechnisch bestimmt. Das UML-Modell basiert auf insgesamt 4-Layern: *MOF* (M3), *UML* (M2), *Schema* (M1) und *Instance Model* (M0). Das Ergebnis eines Konfigurationsprozesses ist die Instanz eines Schemas der Modellebene M1 in der Modellebene M0. FELFERNIG ET AL. erweitern das Meta-Modell der Modellebene M2 indem die Modellebene M1 um ein sog. *Profil* erweitert wird. Ein Profil ist die Verwendung der von der UML in Modellebene M2 angebotenen Elemente (z. B. Klassen, Assoziationen etc.) in Kombination mit spezifischen Stereotypen. Mit Hilfe dieses Profils kann die Modellierung eines spezifischen Domänenwissens erfolgen. Das Profil umfasst folgende Elemente (vgl. Abbildung 5.8):[418]

---

[414] Vgl. Felfernig et al. (2000)
[415] Vgl. Bayer et al. (2005), S. 19ff; siehe auch Abschnitt 5.3.2
[416] Vgl. Felfernig et al. (2000)
[417] Vgl. Felfernig et al. (2001)
[418] Vgl. Felfernig et al. (2001)

- *Komponententypen*: Komponententypen repräsentieren diejenigen Teile, aus denen das Endprodukt zusammengebaut werden kann. Ein Komponententyp besitzt eine Menge von Attributen, denen jeweils eine vorgegebene Wertemenge zugeordnet ist. Beispiele hierfür sind ein Betriebssystem oder eine CPU.
- *Funktionstypen*: Funktionstypen werden zur Modellierung einer funktionalen Architektur verwendet. Auch Funktionstypen besitzen Attribute. Ein typischer Funktionstyp ist die Funktion der Datenspeicherung.
- *Ressourcen*: Teile eines Konfigurationsproblems können als Ressourcenverteilungsproblem betrachtet werden, bei dem einige Komponenten (z. B. Festplatten) Ressourcen (z. B. Festplattenkapazität) produzieren, die von anderen Komponenten (z. B. Softwareprogrammen) konsumiert werden.
- *Generalisierung*: Komponenten können grundlegend gleich sein, sich aber im Detail unterscheiden. Linux und Windows sind beispielsweise beides Betriebssysteme mit einem Anteil gleicher Funktionen. Im Detail unterscheiden sich beide Betriebssysteme.
- *Aggregation*: Mit Hilfe von Aggregationen können Teile-Ganzes Beziehungen erzeugt werden. CPU und RAM-Bausteine sind beispielsweise Teile eines Motherboards. Gleichzeitig setzt sich ein Motherboard u. a. aus einer CPU und RAM-Bausteinen zusammen.
- *Verbindungen und Ports*: Mit Hilfe von Ports können Verbindungen zwischen Komponenten geschaffen werden. Sie beschreiben beispielsweise, wie Komponenten interagieren können. Ein Stecker kann beispielsweise mit einer Steckdose verbunden werden.
- *Kompatibilitätsrelationen*: Zwei Komponenten des gleichen Typs können unter Umständen nicht in der Konfiguration eines Endprodukts verwendet werden. Beispielsweise kann ein PowerPC-Prozessor nicht auf ein Intel-basiertes Motherboard aufgesteckt werden. Andererseits können bestimmte Komponenten (z. B. ein Windows-Betriebssystem) andere Komponenten (z. B. ein Intel-Prozessor) erfordern.

- *Zusätzliche Modellierungskonzepte und Regeln*: Innerhalb eines Konfi-gurationsprozesses kann es vorkommen, dass bestimmte Regeln grafisch nicht ausgedrückt werden können. Für diese Fälle besitzt die UML eine eigene Regelsprache, die sog. *Object Constraint Language* (OCL).
- *Funktionstechnisches Konfigurationswissen*: Typisches Konfigurations-wissen beinhaltet eine strukturelle und funktionale Produktarchitektur, bei der Komponenten Funktionen zugewiesen werden. Eine kundenorientierte Vorgehensweise der Produktkonfiguration ist die Erzeugung einer funktionszentrischen Sicht auf die Komponenten. Die Brücke zwischen einer komponentenzentrischen und einer funktionszentrischen Sicht auf eine Konfiguration wird mit Hilfe einer N:M Verbindung zwischen Funktionstypen und Komponententypen realisiert.[419]
- *Strukturierungsmechanismen*: Die Existenz zweier Sichten auf eine Konfiguration erhöht die Komplexität des Modells. Um diese Komplexität zu beherrschen werden die Modellelemente mit Hilfe von Paketen organisiert, um letztendlich das Gesamtmodell in Teilmodelle aufzuteilen.

[419] Siehe hierzu auch Mittal/Frayman (1989)

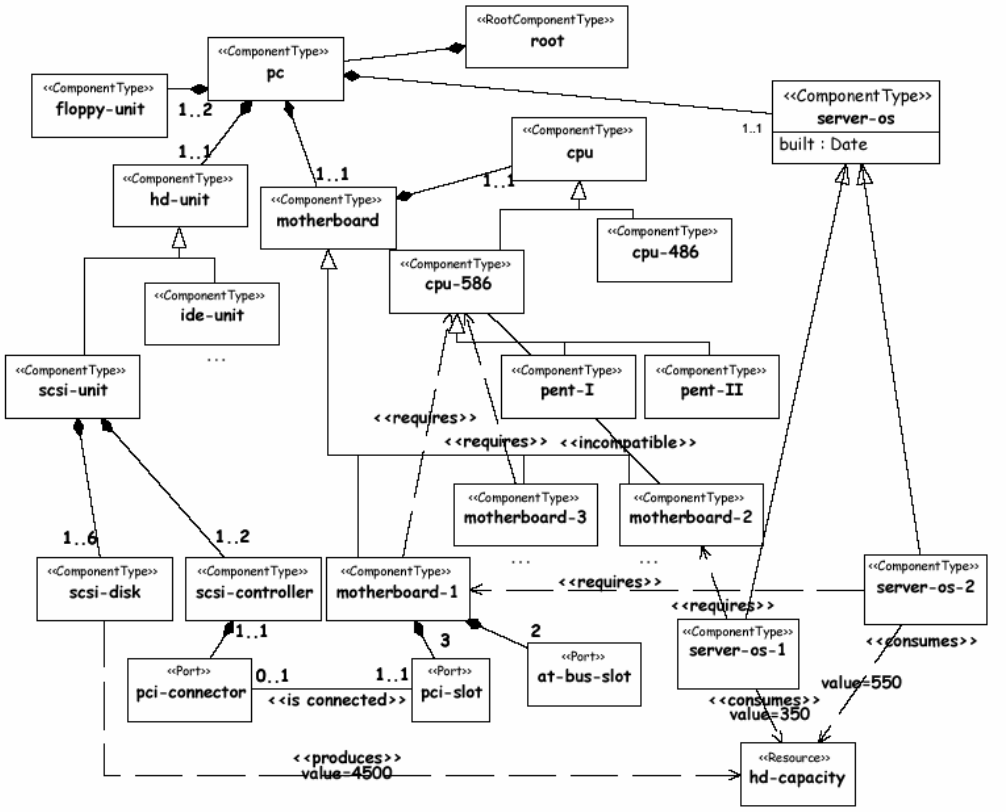


Abbildung 5.8: Konzeptionelles Modell eines Produkts (Quelle: Felfernig et al. (2001)).

5.3.3.3 Matrizen als domänenspezifische Sprache zur Abbildung von Konfigurationswissen

Im Gegensatz zu dem von FELFERNIG ET AL. präsentierten Ansatz einer rein grafisch basierten Abbildung von Konfigurationswissen, setzen die Ansätze von BONGULIELMI und KERSTEN ET AL. auf eine rein Matrix-basierte Abbildung von Konfigurationswissen.

5.3.3.3.1 K- & V-Matrix

Die in Bongulielmi (2002) vorgestellte *Konfigurations- und Verträglichkeitsmatrix*, kurz *K- & V-Matrix*, stellt im klassischen Fall zwei Produktsichten gegenüber: eine funktionale, kundennahe und eine produktnahe, technische Sicht. In jeder der beiden Sichten werden die Informationen, die für die jeweilige Sicht von Interesse sind, strukturiert dargestellt. Die K-Matrix dient der Abbildung der direkten Korrelationen beider Sichten. In der V-Matrix hingegen werden die Elemente der gleichen Sicht auf die gegenseitige Kombinierbarkeit überprüft. Jede Sicht besitzt eine eigene V-Matrix. Der in Abbildung 5.12 dargestellte prinzipielle

Aufbau der K- & V-Matrix zeigt eine sog. *Kundensicht* und eine sog. *Technische Sicht*:[420]

- In der *Kundensicht* werden beschriebene Produktmerkmale und Anforderungen abgetragen, die ein Kunde während des Konfigurationsprozesses zum Zweck der Produktdefinition benötigt. Die Produktmerkmale und Anforderungen werden hierbei in einer von BONGULIELMI als Kundenorientierten Sprache formuliert und abgelegt.
- Die *Technische Sicht* stellt die Komponenten dar, aus denen sich ein Produkt zusammensetzen lässt. Wesentliches Merkmal hierbei ist, dass die Technische Sicht aus der Produktstruktur abgeleitet wird und speziell an die Eigenschaften des Konfigurationsprozesses angepasst wird.

Der Wertebereich der K- & V-Matrizenfelder ist bool'isch. Eine 1 bzw. eine 0 drückt aus, ob zwischen der in X- und der in Y-Richtung abgetragenen Eigenschaft eine Korrelation besteht oder nicht. Konzepte für den generellen Matrizeneinsatz im Umfeld der Produktkonfiguration lassen sich auch bei PAHL und BEITZ sowie BIRKHOFER finden.[421] Eine Verflechtung verschiedener Matrizen, wie sie von BONGULIELMI vorgeschlagen wird, kann jedoch auf AKAO[422] zurückgeführt werden.

Die *K-Matrix* dient hierbei der Gegenüberstellung der Eigenschaften zweier Sichten. In den Matrixfeldern wird die Zuweisung der Merkmale der Kundensicht zu den Komponenten der technischen Sicht dargestellt. Existiert eine Korrelation zwischen einem Merkmal und einer Komponente, so ist die Komponente über das Merkmal „konfigurierbar" und so wird diese Korrelation mit einer 1 in der Matrix festgehalten.

In der *V-Matrix* wird zum Zweck der Überprüfung der gegenseitigen Kombinierbarkeit ein paarweiser Vergleich zwischen Komponenten bzw. Merkmalen abgebildet. Da entweder die Korrelationen zwischen den einzelnen Komponenten oder Korrelationen zwischen den einzelnen Merkmalen festgehalten werden, besitzt die V-Matrix eine symmetrische Form. Insgesamt werden zwei V-Matrizen benötigt. Die V-Matrix, bei der die Komponenten gegenübergestellt werden, wird als erste V-

[420] Vgl. Bongulielmi (2002), S. 62ff
[421] Vgl. Pahl/Beitz (1997) und Birkhofer (1980) (nach Bongulielmi (2002), S. 73)
[422] Akao (1990)

Matrix bezeichnet. Die V-Matrix, bei der die Merkmale Merkmalen gegenübergestellt werden, wird als zweite V-Matrix bezeichnet.

		Fahrer		Personengrösse			Ausführung			Anhänger			Anhängerfarbe				Kindersitz	
		Dame	Herr	< 170 cm	170 cm - 185 cm	> 185 cm	Komfort	Renn	Mountain	geländegängig	nicht vorhanden	strassengängig	rot	blau	gelb	keine	vorhanden	nicht vorhanden
Anhänger-Bereifung	hohes Profil									1								
	Slicks											1						
	normales Profil											1						
	keine										1							
Anhänger-Farbe	blau													1				
	gelb														1			
	rot												1					
	keine															1		
Anhänger-Typ	Anhänger Typ A						1		1	1		1						
	nicht vorhanden						1	1	1		1							
Bereifung	Alu - hohes Profil								1									
	Alu - normales Profil						1											
	Alu - Slicks							1										
	Stahl - hohes Profil								1									
Federung	Federung Typ 1						1		1									
	nicht vorhanden							1										
Kindersitz	Kindersitz Typ 1						1		1								1	
	nicht vorhanden						1	1	1									1
Lenker	Mountain								1									
	Renn							1										
	Tour						1											
Rahmen-Grösse	26"			1														
	28"				1													
	30"					1												
Rahmen-Typ	Damen	1																
	Herren		1															
Schaltung	18-Gang						1											
	24-Gang							1										
	27-Gang								1									

Abbildung 5.9: Die K-Matrix am Beispiel eines Ausschnitts eines fiktiven Fahrrads (Quelle: Bongulielmi (2002), S. 69).

Ein Merkmal (engl.: *feature*) stellt nach KANG ET AL. eine für den Endnutzer sichtbare Charakteristik eines Systems dar.[423] SIMMONS ET AL. definieren ein Merkmal als eine erkennbare Charakteristik eines Konzepts, welches den Interessensbeteiligten von Bedeutung ist.[424] Eine wesentliche Eigenschaft des Systems ist eine Basis zur Verwaltung von wiederverwendbaren und konfigurierbaren Anforderungen.[425] SVAHNBRG und BOSCH differenzieren zwischen Merkmalen und Anforderungen:[426] Während Merkmale kurz und prägnant beschrieben werden und als

[423] Vgl. Kang et al. (1990)
[424] Vgl. Simmons et al. (1996)
[425] Vgl. Simmons et al. (1996) sowie Kang et al. (1990)
[426] Vgl. Svahnbrg/Bosch (1999)

Gruppierung von implementierungsspezifischen Anforderungen dienen, werden Anforderungen unabhängig von Implementierungsdetails sowohl funktional als auch nicht-funktional beschrieben.

Ist eine Komponente mit einer anderen Komponente kombinierbar, so ist sie mit dieser „verträglich“ und so wird in das entsprechende Matrizenfeld der ersten V-Matrix eine 1 eingetragen. Gleiches gilt für die Gegenüberstellung von Merkmalen in der zweiten V-Matrix. Die zweite V-Matrix, bei der die Merkmale paarweise verglichen werden, lässt sich auf Basis der Korrelationen zwischen Merkmalen und Komponenten in der K-Matrix sowie auf Basis der Korrelationen zwischen Komponenten untereinander in der ersten V-Matrix mit Hilfe eines Algorithmus ableiten.

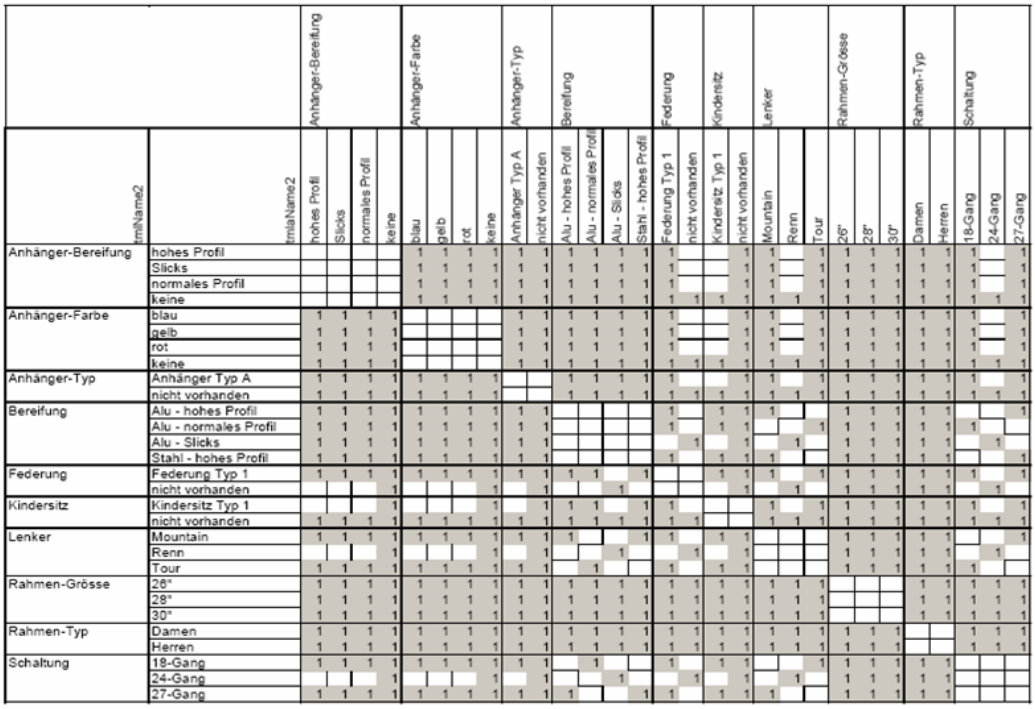


Abbildung 5.10: Ein Beispiel der V-Matrix der Technischen Sicht eines fiktiven Fahrrads (Quelle: Bongulielmi (2002), S. 71).

Vorteil von Matrizen

Matrizen sind besonders für die Darstellung mehrdimensionaler Informationen geeignet. Nach BONGULIELMI können funktionale Sichten, die Beschreibung von Komponenten sowie die Zusammenhänge zwischen Komponenten mit Hilfe von Matrizen abgebildet werden. Zudem bietet die Matrix eine Kompaktheit bezüglich der darzustellenden Informationen, wodurch ein Akteur, der mit dieser Matrix arbeitet, wesentlich schneller das situativ notwendige Wissen erlangen kann. Darüber hi-

naus bieten Matrizen die Möglichkeit einer systematischen Abbildung von Elementen eines Systems sowie die klare und lesbare Darstellung dieser Informationen unabhängig von der jeweiligen Matrixgröße.[427]

		Fahrer		Personengrösse			Ausführung			Anhänger			Anhängerfarbe				Kindersitz	
		Dame	Herr	< 170 cm	170 cm - 185 cm	> 185 cm	Komfort	Renn	Mountain	geländegängig	nicht vorhanden	strassengängig	rot	blau	gelb	keine	vorhanden	nicht vorhanden
Fahrer	Dame			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Herr			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Personengrösse	< 170 cm	1	1				1	1	1	1	1	1	1	1	1	1	1	1
	170 cm - 185 cm	1	1				1	1	1	1	1	1	1	1	1	1	1	1
	> 185 cm	1	1				1	1	1	1	1	1	1	1	1	1	1	1
Ausführung	Komfort	1	1	1	1	1				1	1	1	1	1	1	1	1	1
	Renn	1	1	1	1	1					1					1		1
	Mountain	1	1	1	1	1				1	1	1	1	1	1	1	1	1
Anhänger	geländegängig	1	1	1	1	1	1		1				1	1	1			1
	nicht vorhanden	1	1	1	1	1	1	1	1							1	1	1
	strassengängig	1	1	1	1	1	1		1				1	1	1			1
Anhängerfarbe	rot	1	1	1	1	1	1		1	1		1						1
	blau	1	1	1	1	1	1		1	1		1						1
	gelb	1	1	1	1	1	1		1	1		1						1
	keine	1	1	1	1	1	1	1	1		1						1	1
Kindersitz	vorhanden	1	1	1	1	1	1		1		1					1		
	nicht vorhanden	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		

Abbildung 5.11*: Ein Beispiel der V-Matrix der Kundensicht eines fiktiven Fahrrads (Quelle: Bongulielmi (2002), S. 73).*

In Bezug auf die Abbildung von Konfigurationswissen stellt eine Matrix im Allgemeinen ein Instrument dar, das zwei Aspekte einer Komponente mit einander korelliert. Durch die Vielzahl der Anwendungsgebiete von Matrizen sind Akteure mit dem Umgang Matrix-basierter Informationsdarstellungen vertraut. BONGULIELMI ET AL. sehen hierin einen entscheidenden Vorteil des Einsatzes von Matrizen gegenüber andersartigen Darstellungsformen innerhalb des Modellierungsprozesses von Systemen. Die Kompaktheit, die Matrizen besitzen, stellt einen wesentlichen Vorteil gegenüber grafischen Darstellungsformen dar, insbesondere dann, wenn die zu beschreibenden Systeme sehr komplex sind.[428]

5.3.3.3.2 Modular Engineering

Der von KERSTEN ET AL. vorgestellte Ansatz[429] des *Modular Engineering* soll die Entwicklung modularer Produktarchitekturen im Produktenwick-

[427] Vgl. Bongulielmi (2002), S. 61
[428] Vgl. Bongulielmi et al. (2002)
[429] Vgl. Kersten et al. (2004)

lungs-prozess unterstützen. Das Modular Engineering, das vorwiegend in der klassischen Produktproduktion eingesetzt wird, berücksichtigt zur systematischen Ableitung der Produktstruktur die vom Markt und Wettbewerb, den Abnehmern sowie dem produzierenden Unternehmen an das Produkt gestellten Ansprüche. Ausgangspunkt des Modular Engineering ist eine funktionale Beschreibung des Produktes während der Konzeptionsphase. Insgesamt wird zwischen drei Anforderungskategorien unterschieden, die zum Entwurf einer geeigneten Produktarchitektur dienen und während der Konzeptionsphase relevant sind: Produktstrategie, Kundenanforderungen sowie technisch funktionale Beziehungen.

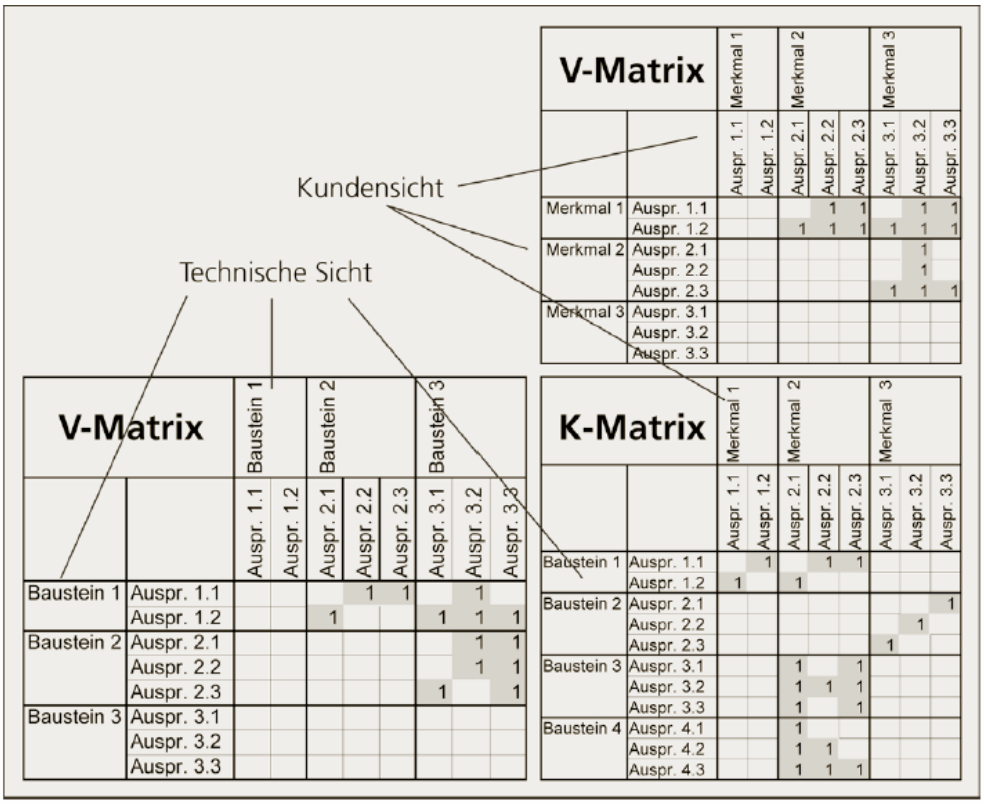


Abbildung 5.12: Der prinzipielle Aufbau der K- & V-Matrix (Quelle: Bongulielmi (2002), S. 74).

Die Kundenanforderungen werden in einer modifizierten QFD-Matrix mit Produktkomponenten korelliert. Mit Hilfe einer Varianzanalyse wird ein Varianzindikator ermittelt, welcher den Beitrag einer Produktkomponente zur Erfüllung stark variierender Kundenanforderungen leistet. Eine Clusteranalyse der QFD-Matrizenergebnisse wird eine Produktarchitektur ermittelt, die die größte Eignung für die Optimierung der Produkteigenschaften in Bezug auf die relevanten Kundenanforderungen besitzt.

Eine weitere Anforderungskategorie ist die Produktstrategie, deren ableitbaren Anforderungen in Form von Modultreibern berücksichtigt werden. In einer sog. *Module Indication Matrix* (MIM), werden die Modultreiber mit den Produktkomponenten korelliert. Das Ergebnis dieser Korellation ist die Isolierung von Produktkomponenten, die in Leistung, Funktionalität oder Design variiert werden müssen.

In einer symmetrischen Matrix, der sog. *Design Structure Matrix* (DSM), findet eine Berücksichtigung der technisch funktionalen Beziehungen zwischen den Produktkomponenten untereinander statt. In der DSM werden die Funktionen und Abhängigkeiten zwischen den Komponenten eingetragen, um anschließend mit diesem Ergebnis eine Modulaufteilung ermitteln zu können.

Die Ergebnismatrizen QFD, MIM und DSM beschreiben Produktarchitekturen, die zu einer Produktstruktur zusammengefasst werden müssen. Die hierzu erforderliche Abwägung unterschiedlicher Sichtweisen wird mathematisch abgebildet. Obwohl KERSTEN ET AL. den Begriff Konfigurationswissen nicht explizit mit der Methode des Modular Engineering in Verbindung bringen, weist die Matrizen-basierte Form der Abbildung von Abhängigkeiten zwischen Komponenten einen starken Zusammenhang mit der Abbildung von Konfigurationswissen auf.

Matrizen als Configuration DSL	**UML als Configuration DSL**
Vorteile: • Allgemeingültiger Ansatz für aus Einzelteilen zusammengesetzte Produkte. • Komprimierte Darstellung und bekannte Notation. • 2-dimensionale Matrizen können in Standardkonfiguratoren verwaltet werden. • Matrizen benötigen keine zusätzlich zu erlernende Sprache. • Die Abbildung von Konfigurationswissen mit Hilfe von Matrizen ist innerhalb einer kurzen Zeit möglich. • Einfache Darstellung von Relationen zwischen zwei Elementen. Nachteile: • Matrizen erlauben lediglich die Gegenüberstellung zweier Dimensionen. • Mehrdimensionalität muss über	Vorteile: • Die Sprache besitzt einen enormen Verbreitungsgrad • UML kann als „bekannt“ vorausgesetzt werden. • UML ist der de facto Standard der Software-Industrie • Sowohl Systemanalytiker als auch Software-Ingenieure verwenden die Sprache aktiv: Sprache der Ingenieure • Durch die Meta-Modellierung bietet die UML eine enorme Flexibilität. • Die Objektorientierung ermöglicht es nicht nur reine Softwaresysteme mit Hilfe der UML zu beschreiben, sondern. Nachteile: • UML ist eine komplexe Symbolsprache mit mehreren Diagrammtypen. • Komplexe Systeme können nur mit-

mehrere 2-dimensionale Matrizen abgebildet werden. Konsistenzprüfungen müssen die Korrektheit der einzelnen Matrizen überprüfen.	tels komplexer Grafiken abgebildet werden. • Besonders Variationen von Aggregationen gleicher Komponenten bedürfen verschiedener komplexer Abbildungen. • Je komplexer die UML-Diagramme sind, desto unübersichtlicher wird das Gesamtsystem. • Die Abbildung von Konfigurationswissen mit Hilfe der UML ist sehr zeitintensiv.

Tabelle *5.3*: Gegenüberstellung der Vor- und Nachteile einer Matrizen- und einer Grafisch-basierten Darstellung von Konfigurationswissen.

5.3.3.4 Vergleich, Bewertung und Analyse

In diesem Abschnitt wurden zwei grundlegend unterschiedliche Ansätze gezeigt, wie Konfigurationswissen sprachlich abgebildet werden kann: zum einen die grafische Abbildung des Konfigurationswissens mit Hilfe der UML, zum anderen die Matrizen-basierte Abbildung des Konfigurations-wissens. Die Vor- und Nachteile beider Ansätze werden nachfolgend in Tabelle 5.3 gegenübergestellt.

UML

Ein wesentlicher Vorteil der UML ist ihr enormer Verbreitungsgrad in der Software Industrie. Dadurch dass die UML nicht nur eine Sprache für die reine Softwareentwicklung ist, sondern auch in anderen Disziplinen des Software Engineerings ihren Einsatz findet und somit als größtenteils „bekannt“ anzunehmen ist, erweist sich als großer Vorteil gegenüber anderen Sprachen welche von den beteiligten Akteuren erst neu erlernt werden müssten.[430] Die UML ist somit eine Sprache von Software Ingenieuren für Software Ingenieure. Das Meta-Modell der UML bietet eine enorme Flexibilität bzgl. der darzustellenden Inhalte. Ein entsprechend formuliertes Meta-Modell nutzt lediglich die „Infrastruktur“ der UML um mit derselben eine eigene Sprache zu definieren. Das Paradigma der Objektorientierung, welches der UML zugrunde liegt, erfasst die beschreibbaren Elemente einer Domäne, weist ihnen Eigenschaften zu und beschreibt ihre Zusammenhänge. Grundsätzlich bedeutet die Darstellung komplexer Zusammenhänge mit der UML aber auch komplexe Diagramme. Insbesondere die Abbildung variantenrei-

[430] Vgl. Mehner/Wagner (2000)

cher Komponentenkombinationen erfordert mehrere komplexe Diagramme bzw. eine große, allumfassende und sehr komplexe Darstellung.

Matrizen

Der Umstand, dass die mit der UML beschriebenen Systeme eine hohe grafische Komplexität aufweisen und ihre Erstellung somit sehr zeitintensiv ist, ist der wesentliche Nachteil dieser Sprache. Demgegenüber steht der Matrizen-basierte Ansatz, dessen wesentlicher Vorteil die Kompaktheit ist. Mit Hilfe von 2-dimensionalen Matrizen können auf einfachste Art und Weise Zusammenhänge zwischen zwei Elementen (dem der X- und dem der Y-Achse) abgebildet werden. Matrizen werden in vielen unterschiedlichen Fächerübergreifenden Disziplinen eingesetzt, wie zum Beispiel dem QFD.[431] Daher benötigt auch ihre Definition keine umfangreiche Sprache. Matrizen bieten den Vorteil, sehr komplexe Zusammenhänge sehr einfach und zeitsparend darzustellen. Jedoch erfordern mehrdimensionale Informationen (wie z. B. die Abbildung von Konfigurationswissen) mehrere Matrizen, deren Inhalte Matrizenübergreifend konsistent sein müssen. Dies ist gleichzeitig der Nachteil dieser Abbildungsform.

Insgesamt ergeben sich die in Tabelle 5.3 gegenübergestellten Vor- und Nachteile beider Darstellungsformen. Interessant ist jedoch zu hinterfragen, inwieweit die vorgeführten Darstellungsformen die Anforderungen an eine Configuration DSL für den Bereich Infrastrukturarchitektur erfüllen. Auf Basis der in Abschnitt 5.3.2 festgelegten Anforderungen an eine Sprache zur Abbildung von Konfigurationswissen der Infrastrukturarchitektur kann nun die Bewertung durchgeführt werden. Für die Bewertung können Punkte der Menge {0; 1; 3; 9} abgegeben werden, wobei 9 die höchste zu erreichende Punktzahl darstellt. Insgesamt ergeben sich die in Tabelle 5.4 abgedruckten, ungewichteten und subjektiven Bepunktungen.

Das Ergebnis dieser Bewertung ist, dass die Matrizen-basierte Darstellungsform mit einem Anteil von 60% der maximal erreichbaren Punktezahl eine geeignetere Configuration DSL darstellt als die UML-basierte Darstellungsform, die einen Anteil von 48% aufweist. Dennoch ist dieses Ergebnis nicht zufrieden stellend. Wie die Gegenüberstellung zeigt,

[431] Vgl. Akao (1990)

punktet die Matrizen-basierte Darstellungsform in der Regel bei denjenigen Anforderungen, bei denen die UML-basierte Darstellungsform einen geringeren Abdeckungsgrad aufweist – und umgekehrt.

Wünschenswert wäre daher eine Methodik, die die Vorteile der Matrizen-basierten Darstellungsform mit denen der UML-basierten Darstellungsform kombiniert. Eine geeignete Methodik, die durchgängig einen hohen Abdeckungsgrad bzgl. der Anforderungen an eine Configuration DSL aufweist, muss demzufolge hybrid sein.

Der im nachfolgenden Kapitel vorgestellte *SI-Katalog* versucht mit einem hybriden Ansatz die Vorteile beider Darstellungsformen zu vereinen und somit eine maßgeschneiderte Lösung für die Abbildung des Konfigurationswissens der Infrastrukturarchitektur von Finanzdienstleistern zu repräsentieren.

Anforderung	**Matrix**	**UML**
Leicht erlernbare, möglichst intuitiv verständliche Modellierungssprache.	9	3
Minimierung des Dokumentations- und damit Änderungsaufwands durch Vermeidung redundanter oder nicht adäquat nutzbarer Information. Dies erfordert eine Abwägung zwischen exakter Modellierung und geringerem Dokumentationsaufwand.	3	1
Strukturierte und modularisierte Datenhaltung, in der das für einen bestimmten Sachverhalt relevante Wissen kompakt dargestellt ist oder einfach generiert werden kann.	9	1
Veränderung können leicht angepasst werden: Kleine Produktänderungen erfordern auch nur kleine Änderungen an der Dokumentation.	9	3
Algorithmisch effizient auswertbar.	9	1
Infrastruktursysteme sind Kompositionen von Hardware- und Softwarekomponenten. Eine Configuration DSL speziell für den Bereich der Infrastrukturarchitektur muss zwischen diesen beiden Komponenten differenzieren können.	3	9
Jede Komponente besitzt ein spezifisches Einsatzgebiet. Beispielsweise dient eine Firewall der Unterbindung eines unerlaubten Datenaustauschs zwischen zwei Rechnern. Das Einsatzgebiet entscheidet über die Selektion einer Komponente und muss abgebildet werden.	1	9
Komponenten können unterschiedlich komponiert werden. Jede Kompositionsvariante kann der gleichen Obergruppe (z. B. „Datenbanksystem) untergeordnet sein, die variationsspezifische Fähigkeit der jeweiligen Komposition kann jedoch unterschiedlich sein (z. B. „Skalierbares Datenbanksystem“ oder „Verfügbares Datenbanksystem“).	3	9
Die Selektion einer Komponente kann die Selektion anderer Komponenten erfordern. Beispielsweise erfordert ein Datenbanksystem zur Lauffähigkeit ein Betriebssystem. Angenommen es existieren die Betriebssysteme BS_1 und BS_2 sowie die	3	3

Datenbanksysteme $DBMS_1$ und $DBMS_2$. Sofern eine strategische Entscheidung den Betrieb des Datenbanksystems $DBMS_1$ mit dem Betriebssystem BS2 vorsieht, muss die Configuration DSL eine Muss-Abhängigkeit zwischen $DBMS_1$ und BS_2 abbilden können. Existiert keine derartige Entscheidung müssen Kann-Abhängigkeiten zwischen jedem Datenbanksystem und jedem Betriebssystem abgebildet werden.		
Summe:	55	39
Anteil an der maximal erreichbaren Punktezahl:	60%	48%

Tabelle *5.4*: Bewertung der Matrizen- und Grafisch-basierten Darstellung von Konfigurationswissen bzgl. der Abdeckung der Anforderungen an eine Configuration DSL für Infrastruktursysteme.

5.4 Bestehende Ansätze zur Strukturierung der Infrastruktur

Liu

In seiner Arbeit[432] *A Practial Framework for Discussing IT Infrastructure* präsentiert LIU eine Strukturierungshilfe zur Komplexitätsbewältigung im Umfeld der IT Infrastruktur. Wie in Abbildung 5.13 dargestellt, wird das Gesamtsystem Infrastruktur in mehrere logische Schichten zerlegt, denen unterschiedliche Infrastrukturkomponenten zugewiesen werden können. LIU definiert IT Infrastruktur als „[…] a set of IT ressources and organizational capabilities that employees share across the organization."[433] Die von LIU vorgenommene Strukturierung erfasst jedoch nur die „essentiellen Elemente" der Infrastruktur. Somit handelt es sich hierbei nicht um ein vollständiges Modell, sondern eher um einen Wegweiser – eine Strukturierungshilfe.

Im Einzelnen teilen sich die von LIU ausgewählten Schichten auf eine organisatorische und 7 technische Schichten wie folgt auf:

- *IT human/management competence*: Diese Schicht umfasst alle Elemente, die organisatorisch notwendig sind, um die existierenden Infrastrukturkomponenten effektiv planen, einsetzen und entwerfen zu können.
- *Vertical business applications*: Die vertikal auf das Framework „aufgesetzten" Geschäftsapplikationen sind die für eine Organisation spezifischen Teile eines unternehmensspezifischen Informations-

[432] Vgl. Liu (2002)
[433] Vgl. Liu (2002)

systems. Als Beispiel für eine derartige Applikation nennt LIU die Kundenkontenverwaltung, die speziell in Banken eingesetzt wird.

- *Horizontal infrastructure applications*: Die horizontalen Infrastrukturapplikationen, auf die die vertikalen Geschäftsapplikationen aufsetzen, stellen Unternehmensübergreifende Funktionen bereit, zählen aber dennoch zu den Geschäftsapplikationen im Allgemeinen. Ein Beispiel für derartige Unternehmensübergreifende Applikationen ist ein SAP-System, das einen Branchenübergreifenden Einsatz findet (z. B. Rechnungswesen).
- *Database, Application & Integration Servers*: Datenbanksysteme (z. B. Oracle Datenbankserver oder Microsoft SQL Server) speichern Informationen, die von anderen Applikationen mittels Abfragemechanismen zur weiteren Verarbeitung wieder ausgelesen werden können. Applikationsserver (z. B. JBoss Applikationsserver oder IBM WebSphere) stellen Laufzeitumgebungen für andere Applikationen bereit. Sie stellen diesen Applikationen eine bestimmte Menge an grundlegenden administrativen Funktionen zur Verfügung, die diese während der Laufzeit in Anspruch nehmen können. Integrationsserver (wie beispielsweise der Host Intregration Server) der Firma Microsoft haben die Aufgabe die Zusammenarbeit und den Datenaustausch der diversen Applikationen zu gewährleisten.
- *Common services/middleware*: In die Kategorie der allgemeinen Dienste fallen Applikationen, die diejenigen Funktionen ausführen, die niederer Bedeutung sind. Als Beispiel für eine derartige Applikation nennt LIU eine Firewall. Middleware Produkte ermöglichen eine technologische Entkopplung der Client- und Serverseite.
- *Operating Systems*: Betriebssysteme stellen die Kernfunktionalitäten zur Lauffähigkeit und Administrierbarkeit der Applikationen bereit.
- *Hardware platforms*: Die Gruppe der Hardware-Plattformen umfasst alle Hardware-Produkte, die für den Betrieb der Applikationen benötigt werden. LIU zählt neben Rechnern mit CPUs auch z. B. Storage Systeme zu den Hardware-Plattformen.

- *Network communications*: Physisch voneinander getrennte Applikationen, die untereinander kommunizieren, benötigen Netzwerkkomponenten, die derartige Dienste zur Verfügung stellen. Zu dieser Kategorie zählt LIU sowohl Hardware-Komponenten (wie beispielsweise Router und Switches) und auch Protokolle (z. B. das Internet Protocol).

Dern

Eine ähnliche Unterteilung wie die von LIU lässt sich bei DERN finden:[434] Der von DERN als IT-Basisinfrastruktur bezeichnete Bestandteil einer Architektur umfasst alle Hardware- und systemnahen Softwarekomponenten. Nach DERN ist es die Aufgabe der Architektur diese Komponenten zu sog. *Basisplattformen* zu gruppieren, welche die Ziel-Infrastruktur eines Informationssystems bilden.

Eine Basisplattform besteht nach DERN aus folgenden Bestandteilen:[435] *Management and Operations*, *Security Environment*, *Enterprise Application and Middleware*, *Application Server*, *Communication and Collaboration*, *Datatbase Management Systems*, *Operation Systems*, *Storage Systems*, *Network Systems*. Was sich genau hinter diesen Bestandteilen verbirgt bleibt offen. Es kann jedoch vermutet werden, dass diese zu Basisplattformen gruppierten Bestandteile eine ähnliche Funktion innerhalb des Software Engineering Prozesses besitzen, wie die Automobilplattformen, die innerhalb der von Volkswagen Mitte der 90ger eingeführten Plattformstrategie entworfen wurden:[436] Sie sind von Ingenieuren vordefinierte Kompositionen von Komponenten, die von Ingenieuren während des Entwurfs neuer Software Systeme bzw. Automobile verwendet werden.

[434] Vgl. Dern (2003), S. 27
[435] Vgl. Dern (2003), S. 28
[436] Vgl. Abschnitt 5.1.1

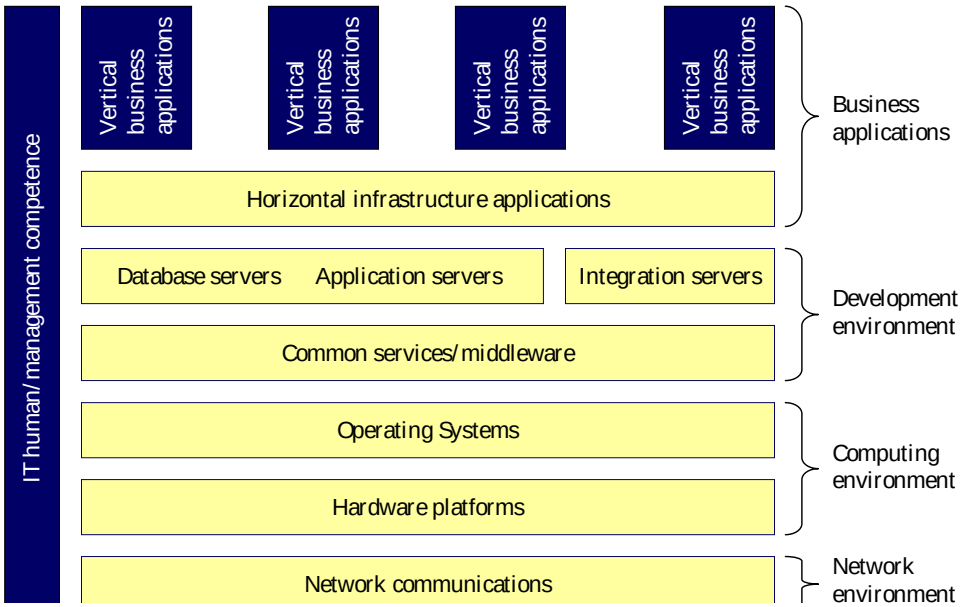


Abbildung 5.13: IT infrastructure framework (Quelle: In Anlehnung an Liu (2002)).

Penzel

Die von DERN als Basisplattformen bezeichneten Kompositionen von Infrastrukturkomponenten lassen sich in vergleichbarer Form auch bei PENZEL finden:[437] Kernidee der von PENZEL als sog. *Technologie-Sets* bezeichneten Kompositionen von Infrastrukturkomponenten ist die Zentralisierung, Standardisierung und Reduktion der Zahl der Betriebsumgebungen. Grund für die Bildung der Technologie-Sets ist die Vielzahl der in einem Rechenzentrum existierenden Betriebsumgebungen. Als Betriebsumgebungen werden hierbei Kompositionen von Infrastrukturkomponenten bezeichnet, die in Anlehnung an die Definition von DERN die IT-Basisinfrastruktur bilden: „Mit dem Host, den Client/Server-Welten auf Basis von UNIX/Java, den Client/Server-Welten auf Basis von Microsoft und spezifischen Standardsoftware-Umgebungen (z. B. SAP oder PeopleSoft) sind nur die grundlegenden Umgebungen genannt. In jeder Umgebung werden […] ein oder auch mehrere Produkte verwendet. Fast jedes Produkt ist in vielfältigen – älteren und neueren – Release-Ständen im Einsatz, die vielleicht nicht einmal mehr vom Hersteller unterstützt werden.“[438] Im Fall der HVB erwähnt PENZEL 150 Varianten von Betriebsumgebungen.

Auf Grund dieser Vielfalt hat die HVB die Bildung der Technologie-Sets eingeführt, die zunächst für Host, UNIX/Java-Umgebungen, Microsoft-

[437] Vgl. Penzel (2004)
[438] Penzel (2004)

Umgebungen und SAP-Umgebungen. Wie aus Abbildung 5.14 hervorgeht, besteht ein Technologie-Set aus unterschiedlichen Infrastrukturkomponenten, die in insgesamt 6 logische Ebenen in der Vertikalen eingeteilt sind. In der Regel gibt es zwei Varianten ein- und desselben Technologie-Sets – eine Standardvariante und eine hochverfügbare Variante. Beide Varianten unterscheiden sich somit sowohl in der Leistung als auch in ihrer Architektur. Eine detailliertere Beschreibung der Technologie-Sets wird nicht geliefert.

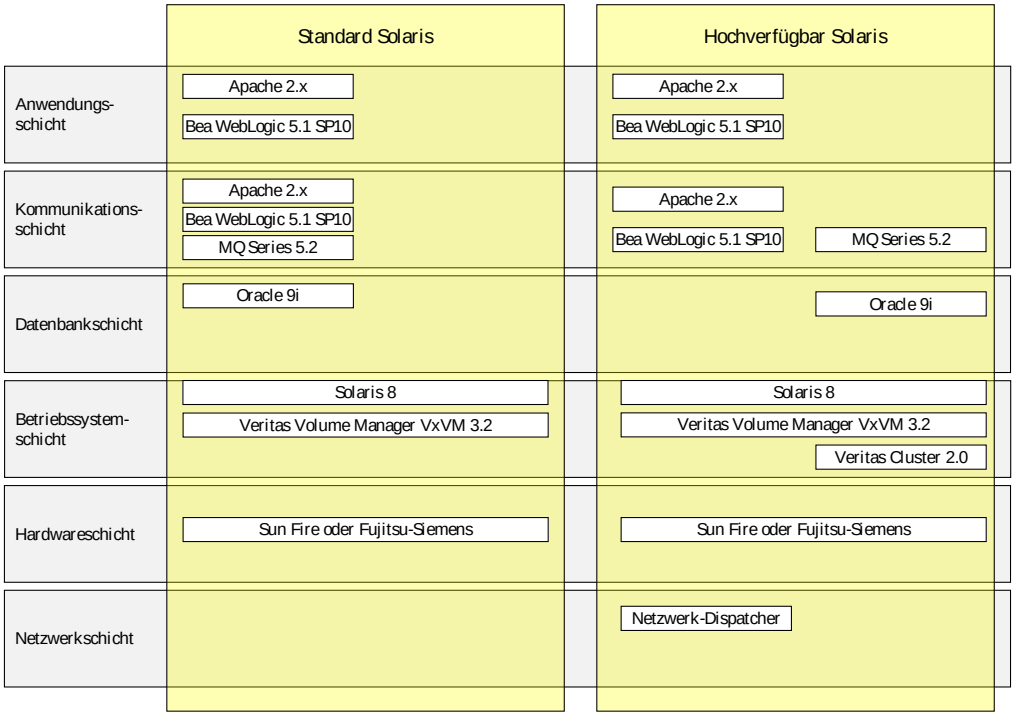


Abbildung 5.14: Technologie-Sets (in Anlehnung an Penzel (2004)).

Greiner et al.

Um eine Architektur bzgl. ihrer Strategiekonformität bewerten zu können, strukturieren GREINER ET AL. Technologiearchitekturen wie folgt:[439] Eine Technologiearchitektur folgt einem Modell, das in drei Strukturierungs-ebenen aufgeteilt ist: *Layer*, *Funktionalgruppe* und *Element*. Ein Element ist hierbei die atomare Einheit einer Gliederungsebene und beschreibt eine Technologie. Diese kann durch folgende Attribute definiert werden: eine Bezeichnung (z. B. „DB2“), eine Versionsnummer (z. B. „8“) und einen Hersteller (z. B. „IBM“). Falls erforderlich kann diese Definition um weitere Attribute erweitert werden. Eine Funktionalgruppe (z. B. „DBMS“) fasst mehrere Technologien zusammen und stellt eine

[439] Vgl. Greiner et al. (2005)

Funktionalität zur Verfügung. Ein Layer (z. B. „Datenmanagement“) bildet die oberste Gruppierungsebene und umfasst mehrere Funktionalgruppen. Ein Layer repräsentiert einen Ordnungsrahmen.

5.5 Über Strategie

Der Duden definiert Strategie wie folgt: „[ein] genauer Plan des eigenen Vorgehens, der dazu dient, ein militärisches, politisches, psychologisches o. ä. Ziel zu erreichen, und in dem man diejenigen Faktoren, die die eigene Aktion hinterspielen könnten, von vornherein einzukalkulieren versucht.“[440]
Strategisches Handeln wird im weiteren Verlauf der Arbeit immer wieder verwendet, um Entscheidungen zu begründen. Daher wird dem Thema Strategie mit diesem Abschnitt besondere Bedeutung zugewiesen. Es ist insofern relevant, da strategische Entscheidungen Teil der Methodik sind und der Einsatz der Methodik selbst eine strategische Entscheidung voraussetzt.

5.5.1 Etymologische Herleitung

Der Begriff *Strategie* lässt sich etymologisch auf die beiden altgriechischen Wörter „strategos“ (Heerführer) und „strategia“ (Feldherrnkunst) zurück-führen. CARL V. CLAUSEWITZ formulierte den Begriff Strategie im militärischen Sinn 1832 wie folgt: „Die Strategie ist der Gebrauch des Gefechts zum Zwecke des Krieges, sie entwirft den Kriegsplan, und an dieses Ziel knüpft sie eine Reihe der Handlungen an, welche zu demselben führen sollen, d. h. sie macht die Entwürfe zu den einzelnen Feldzügen und ordnet in diesen die einzelnen Gefechte an“.[441] Ein im Zusammenhang mit dem Begriff Strategie abzugrenzender Begriff ist der Begriff *Taktik*. Hierunter versteht V. CLAUSEWITZ die „Lehre vom Gebrauch der Streitkräfte im Gefechte“.[442] Er beschreibt mit dem Begriff Strategie das Verhalten der Truppenführung und der Truppe auf dem

[440] Duden (1990), S. 745
[441] Vgl. Clausewitz (2002), S. 155
[442] Vgl. Clausewitz (2002), S. 26

Kampffeld. In der Strategie gibt es laut v. CLAUSEWITZ keinen Sieg. Ein strategischer Erfolg hat für ihn zwei Seiten. Auf der einen Seite ist ein strategischer Erfolg die Benutzung des erfolgten Sieges. Und auf der anderen Seite ist ein strategischer Erfolg die exakte Vorbereitung des taktischen Sieges.[443] HINTERHUBER schließt daraus, dass es in der Strategie immer ein „morgen“ gibt, in der Taktik kann der Misserfolg entscheidend sein.[444]

Ursprung

Strategie ist also ein Mittel, um ein Ziel zu erreichen. Taktik ist ein Mittel zur Umsetzung der Strategie.[445] Strategie ist laut v. CLAUSEWITZ durch ein Gesamtkonzept zur Erreichung eines Ziels gekennzeichnet, wobei mit größeren Raum- und Zeiteinheiten sowie aggregierte Größen zum Einsatz kommen.[446] Aufgabe einer Strategie ist es, aufzuzeigen, was machbar ist und was nicht, sowie klar zu machen, welche Folgen damit verbunden sind.[447] Die Entwicklung einer Strategie sollte dabei auf folgenden Anforderungen beruhen:[448]

- Förderung der Schaffung, Nutzung und langfristigen Erhaltung der Erfolgspotentiale, wobei eine Bandbreite möglicher Entwicklungspfade zugestanden werden. Konkret bedeutet dies den Aufbau von Stärken, die Vermeidung von Schwächen, die Konzentration der Kräfte sowie den Aufbau und die Ausnutzung von Synergieeffekten.
- Vermeidung unnötiger Risiken sowie die Streuung der Risiken in Bezug auf Handlungen.
- Die Grundsätze verschiedener alternativer Lösungswege sollten flexibel und anpassungsfähig bleiben.

5.5.2 Betriebswirtschaftliche Bedeutung

Der Strategiebegriff hat auch in anderen Disziplinen seinen Platz gefunden; unter anderem auch in der Betriebswirtschaft. Als Beleg für den Zusammenhang von Betriebswirtschaft und der ursprünglichen militäri-

[443] Vgl. Clausewitz (2002), S. 403
[444] Vgl. Hinterhuber (1990), S. 59
[445] Vgl. Hinterhuber (1990), S. 62
[446] Vgl. Clausewitz (2002), S. 318
[447] Vgl. Hinterhuber (1990), S. 57
[448] Vgl. Götze (1994)

schen Strategie werden in der Literatur häufig die Dialoge zwischen Sokrates und Nichomachides genannt. In diesen Dialogen verweist Sokrates darauf, dass die Pflichten eines Generals durchaus mit denen eines Geschäftsmanns vergleichbar seien: beide beinhalten die Planung der Verwendung von Ressourcen, um bestimmte Ziele zu erreichen.[449]

Als weiterer Hinweis auf die militärisch geprägten Einflüsse auf betriebswirtschaftliches Denken und Handeln liefert die Tatsache, dass viele Autoren der US-amerikanischen Managementliteratur eine Ausbildung an der West Point Offizierschule genossen. Ein Blick in die Wirtschaftsteile aktueller Zeitschriften macht dies ebenfalls deutlich: hier stehen Begriffe wie „feindliche Übernahme", „Preiskampf", „Übernahmeschlacht" oder „Kostenführerschaft" stets in den Überschriften.

An einer Klärung, welche Ausprägungen der Begriff Strategie im betriebswirtschaftlichen Umfeld besitzt, versuchen sich diverse Autoren.[450] Oftmals wird Strategie jedoch mit seinem militärischen Ursprung erläutert. So zitiert GRANT beispielsweise den Chinesen Sun Tzu, der um ca. 500 vor Christus ein Werk über die Kriegskunst verfasst hat: „Strategy is the great work of the organization. In situations of life or death, it is the Tao[451] of survival or extinction. Its study cannot be neglected."[452] GRANT verbindet den militärischen Ursprung des Strategiebegriffs mit der betriebswirtschaftlichen Auffassung, in dem er zwischen Strategie und Taktik unterscheidet:[453]

Strategie <> Taktik

> Strategie ist der übergreifende Plan zur Nutzung und Aufstellung der eigenen Ressourcen, um eine vorteilhafte Position zu aufzubauen. Im militärischen Sinn ist dies der Gewinn eines Krieges. Im betriebswirtschaftlichen Sinn ist dies die Erreichung einer überlegenen Wettbewerbsposition.

---

[449] Vgl. Bracker (1980)
[450] Beispiele sind (in Anlehnung an Nippa (1999), S. 4): Ansoff (1965); Grant (1998), S. 14ff; Mintzberg (1994), S. 23ff; Porter (1996); Quinn (1996), S. 3ff
[451] Anmerkung: Tao, wörtlich „Weg" meint in der chinesischen Philosophie ein der ganzen Welt zugrunde liegendes, alldurchdringendes Prinzip der Bewegung, des Prozesses. Dieses erzeugt alles und ordnet, ähnlich einem Naturgesetz, alle Abläufe im Kosmos
[452] Grant (1998), S. 3
[453] Vgl. Grant (1998), S. 14

> Taktik hingegen ist der Plan für eine spezifische Aktion. Im militärischen Sinn ist dies der Gewinn einer Schlacht (von vielen innerhalb) des Krieges. Im betriebswirtschaftlichen Sinn ist dies die Erreichung eines spezifischen Wettbewerbsvorteils (z. B. Markt- oder Kostenführerschaft in einem spezifischen Marktsegment).

Ähnlich wird in Porter (1996) differenziert: Zu den Grundbausteinen eines Wettbewerbsvorteils zählt PORTER diejenigen Aktivitäten, die dem Erzeugen, Produzieren, Verkaufen oder Ausliefern von eines Produktes oder Dienstes zugeordnet werden können. Sofern diese Aktivitäten schneller oder mit weniger Ressourcen und Fehlern ausgeführt werden, als dies die Konkurrenz kann, agiert das Unternehmen *operativ effektiv*. Als Beispiele für derart operativ effektiv operierende Unternehmen nennt PORTER die japanischen Firmen, die zwischen 1970 und 1990 Mittel wie z. B. Total Quality Management oder kontinuierliche Verbesserung einsetzten. Das Problem der operativen Effektivität ist, dass die hierbei praktizierten Verfahrensweisen schnell als sog. *Best Practices* von der Konkurrenz adaptiert werden. Unternehmen geraten somit schnell an eine fixe *Produktivitätsgrenze*. Die Folge ist, dass Unternehmen, die lediglich ihre Produktivität verbessern, über kurz oder lang an diese Grenze stoßen. Ihr Vorsprung gegenüber der Konkurrenz hält nur solange an, bis diese die Verfahren der operationalen Effektivitätssteigerung adaptiert hat und ebenfalls an diese Grenze stößt. Dies zeigt, dass eine permanente Effektivitätssteigerung zwar notwendig ist, dass der aus ihr resultierende Vorteil aber nur von begrenzter Dauer ist.

Als Folge des allein auf Produktivitätsoptimierung ausgerichteten Handelns ergibt sich eine Endlosspirale mit immer geringen Gewinnmargen. Da die Gewinne sinken versuchen die Unternehmen die Produktivität zu senken, indem sie die Aktivitäten, die zu operationaler Effektivität führen, an Dritte auszulagern.[454] Doch hierdurch wird laut PORTER der Konvergenzprozess an die Produktivitätsgrenze noch weiter beschleunigt, da diese Dritten die Aktivitäten noch schneller optimieren bzw. von anderen Dritten, mit denen sie wiederum in Konkurrenz stehen, adap-

[454] oder neudeutsch : outzusourcen.

tieren. Am Ende kommt es wie es kommen musste: Die Großen kaufen die Kleineren auf.

Strategische Positionierung eines Unternehmens

Den Ausweg aus dieser Spirale bietet nur die strategische Positionierung eines einzelnen Unternehmens bzw. dessen Geschäftsmodell. So heißt es bei PORTER: „Strategic positioning attempts to achieve sustainable competitive advantage by preserving what is distinctive about a company. It means performing different activities from rivals, or performing similar activities in other ways."[455]

NIPPA hebt in seinem Arbeitspapier jedoch diese desillusionierende Stimmung, indem er darauf verweist, dass strategische Unternehmensführung weniger von theoretischen Konstrukten als eher von pragmatischem Handeln geprägt ist. Der von PORTER initiierte Fokus auf Wettbewerbs-relevanz von Industrie- und Branchenstrukturen sei durch die Konzentration auf die ursprünglichen Unternehmenspotentiale abgelöst worden. Hierzu zählen z. B. Kernkompetenzen oder die Ressourcen-orientierte Unter-nehmenssicht.[456]

Strategieverständnis dieser Arbeit

Zusammenfassend wird der Begriff Strategie im Rahmen dieser Arbeit in Anlehnung an GÖTZE mit folgenden Aktivitäten verbunden:[457]

- Förderung der Schaffung, Nutzung und langfristigen Erhaltung der Erfolgspotentiale.
- Aufbau von Stärken, die Vermeidung von Schwächen, die Konzentration der Kräfte sowie den Aufbau und die Ausnutzung von Synergieeffekten.
- Vermeidung unnötiger Risiken sowie die Streuung der Risiken in Bezug auf Handlungen.

Da Aktivitäten eine Form des Handelns sind, und strategische Entscheidungen im Umfeld des sog. Top-Managements getroffen werden, ist Strategie ein Teil des Managements.

Strategie ist ein Richtungsgeber für das Management. Genauer betrachtet ist Strategie ein Teil des Managements, dessen konkrete Ausprägung von den strategischen Zielen und Visionen abhängt.

---

[455] Porter (1996)
[456] Vgl. Nippa (1999), S. 6
[457] Vgl. Götze (1994)

6 Gestaltungskonzept – Die Methodik

Das Ergebnis der vorherigen Kapitel ist die Feststellung, dass die Handhabung der Komplexität im Bereich der Infrastrukturarchitektur von Finanzdienstleistern eine Methodik erfordert, die den speziellen Belangen dieser Domäne gerecht wird. Kern dieser Arbeit ist die Vorstellung einer ebensolchen Methodik. Die präsentierte Methodik ist das Ergebnis der Aktionsforschung bei der Postbank Systems. In mehreren Iterations-schritten wurden die real existierenden Zusammenhänge der Infrastrukturarchitektur aufgenommen und analysiert. Auf Basis der Analysen wurde die Methodik entworfen und kontinuierlich anhand neu gewonnener Erkenntnisse und Fehlschläge sowie der Leittheorien zum Modellentwurf modifiziert. Die Methodik wird im weiteren Verlauf mit dem bei der Postbank Systems eingeführten Terminus *SI-Katalog* benannt.

6.1 Methodik: Der SI-Katalog

Aufgabe dieses Kapitels soll es sein, das Konzept des *Standard-Infrastruktur-Katalogs* (im Folgenden kurz *SI-Katalog*) vorzustellen. Der SI-Katalog besteht im Wesentlichen aus zwei Elementen: sog. *Standard-Infrastruktur-Komponenten* (im Folgenden kurz *SI-Komponenten* genannt) und sog. *Standard-Infrastruktur-Architekturmustern* (im Folgenden kurz *SI-Architekturmuster* oder einfach *Architekturmuster* genannt).

Der SI-Katalog wird als ein Organisationsinstrument zur strukturierten und Kontextbezogenen Dokumentation der Infrastruktur-Komponenten betrachtet. Der aus dem griechischen stammende Begriff *Katalog* bezeichnet hierbei ein mit einer bestimmten Systematik aufgebautes Verzeichnis, das vor allem der Übersicht über Sammlungen dient.[458]

Die in Abschnitt 5.3.3.2.3 gewonnenen Erkenntnisse zeigen, dass Muster dazu geeignet sind, Konfigurationswissen abzubilden. Da diese Arbeit die Infrastrukturarchitektur bzw. den Entwurf von Infrastruktursys-

[458] Vgl. Duden (1990), S. 394

temen fokussiert, werden Architekturmuster dazu verwendet, um das Konfigurationswissen bzgl. der Infrastrukturkomponenten katalogartig abzubilden. Der SI-Katalog wird als ein „lebendes“ Dokument betrachtet welches einem Lebenszyklus unterliegt. Bevor der SI-Katalog jedoch einem solchen Lebenszyklus der kontinuierlichen Aktualisierung und Konsistenzprüfung unterworfen werden kann, muss er initial aufgebaut, d. h. „entworfen“ werden.

Der Lebenszyklus des SI-Katalogs ist in Abbildung 2.8 dargestellt: Insgesamt drei Phasen sind für die Ausgestaltung der Inhalte des SI-Katalogs verantwortlich. Der Eintritt in eine jeweilige Phase erfolgt als Reaktion auf ein ausgelöstes Ereignis. Die Phasen sind somit lose voneinander gekoppelt. Folgende Ereignisse können zu einem Phaseneintritt führen:

- *E1*: Der SI-Katalog soll initial angelegt werden
- *E2*: Der SI-Katalog soll in einem Software Engineering Prozess verwendet werden
- *E3*: (Ein vorgegebener zeitlicher Zyklus ist beendet) oder (Neue Infrastrukturkomponenten bzw. Infrastrukturkomponentenvarianten wurden eingeführt) oder (Ein neues Architekturmuster muss eingepflegt werden)

Jede Phase besteht aus mehreren Prozessschritten. Jeder Prozessschritt kann sich wiederum in Unterprozessschritte aufteilen. Die Prozesselemente werden in der Form *<Phase>.<Subprozess>.<Prozessschritt>* notiert, so dass eine eindeutige Identifikation erfolgen kann. Phasen werden hierbei in römischen Ziffern, Subprozesse alphanumerisch und Prozessschritte numerisch gekennzeichnet.

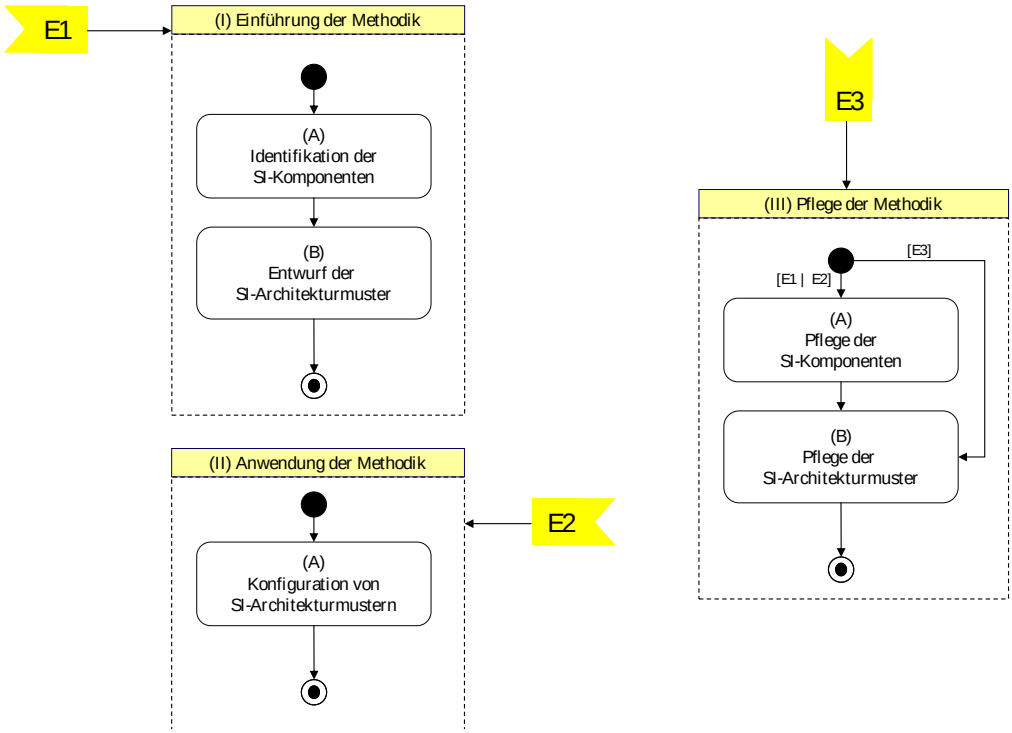


Abbildung 6.1: Der Lebenszyklus des SI-Katalogs. (Quelle: Eigene Darstellung)

6.2 Meta-Ebenen innerhalb des SI-Katalogs

Wie die Zusammenhänge der in Abbildung 6.2 abgedruckten Meta-Ebenen des SI-Katalogs zeigen, ist das *Meta-Modell* auf Layer M2 eine Konkretisierung der UML Spezifikation, deren Modellierungskonzepte selbst wiederum in der MOF auf Layer M4 definiert sind.[459] Das Meta-Modell spezifiziert ein Regelwerk für SI-Architekturmuster, SI-Komponenten und Qualitätsattribute. Im Meta-Modell des SI-Katalogs werden die abstrakten Zusammenhänge derjenigen Meta-Elemente beschrieben, die innerhalb des Layers M1 die Architekturmuster komponieren. Wird ein Architekturmuster innerhalb eines Software Engineering Projekts instanziiert, so wird diese Laufzeitinstanz auf Layer M0 konkretisiert indem eine qualitative bzw. quantitative Konfiguration eines oder mehrerer Architekturmuster erfolgt.

[459] Bzgl. MOF siehe Felfernig et al. (2001)

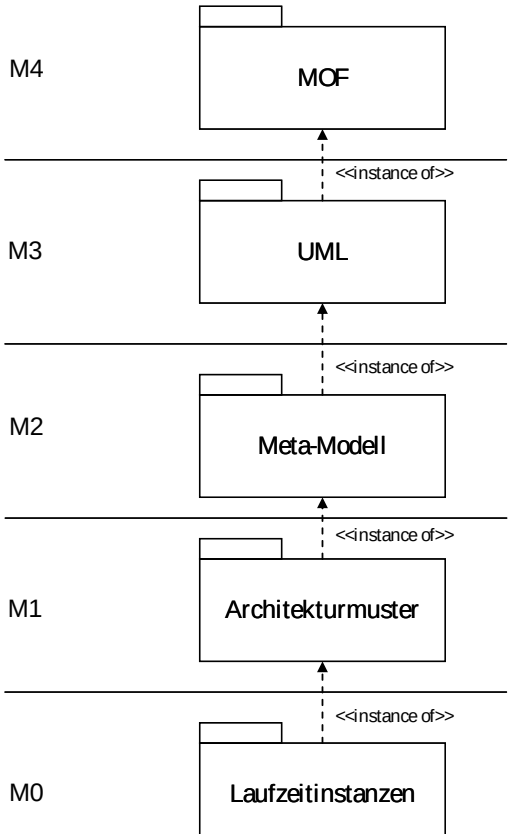


Abbildung 6.2: Meta-Ebenen des SI-Katalogs. (Quelle: Eigene Darstellung)

Stereotypen

Um die im weiteren Verlauf verwendeten (Meta-)Klassen einem spezifischen Layer zuordnen zu können, werden Stereotypen verwendet (Tabelle 6.1). Im Wesentlichen konzentriert sich die Modellierungsarbeit innerhalb des SI-Katalogs auf die Layer M0, M1 und M2, da das vorhandene Regelwerk der UML lediglich ergänzt wird.

Layer	Bezeichnung	Beschreibung
M2	*Meta-Klasse*	Eine als *Meta-Klasse* gekennzeichnete Klasse des Layers M2 ist eine konkrete Instanz einer UML-Klasse und beschreibt, wie sich eine innerhalb des Layers M1 konkretisierte *Design-Klasse*, ein *Design-Muster* oder eine *Port-Klasse* zu anderen Elementen des Layers M2 verhält.
	Parameter-Klasse	Eine *Parameter-Klasse* spezifiziert eine *qualitative Ausprägung*, die innerhalb des Layers M2 einem *Design-Klasse* bzw. einem *Design-Muster* zugeordnet werden kann.
M1	*Design-Klasse*	Eine *Design-Klasse* ist eine Konkretisierung einer *Meta-Klasse* und kennzeichnet eine Infrastrukturkomponente, die zur Komposition eines Architekturmusters herangezogen werden kann.
	qualitative Ausprägung	Eine *qualitative Ausprägung* konkretisiert eine *Parameter-Klasse* in Layer M1 und beschreibt entweder die Qualität einer als *Design-Muster* oder als *Design-Klasse* gekennzeichneten Instanz in Layer M1.

	Design-Muster	Ein *Design-Muster* kennzeichnet eine spezifische *Meta-Klasse*, die als Architekturmuster fungiert und Beziehungen zu Klassen vom Typ *Design-Klasse* und *qualitative Ausprägung* aufweist.
	Port-Klasse	Eine *Port-Klasse* ist eine spezifische Instanz einer *Meta-Klasse*, die zwischen mindestens zwei Klassen vom Typ *Design-Klasse* eine kommunizierende Rolle übernimmt.

Tabelle *6.1*: Stereotypen der Layer M0, M1 und M2.

6.3 Rollen innerhalb des SI-Katalogs

6.3.1 Das Rollenkonzept im Allgemeinen

Eine *Rolle* ist eine Menge von zusammengehörenden Aufgaben, die mindestens einer Person zugewiesen werden. Eine Rolle trägt die funktionale Verantwortung für die Erledigung dieser Aufgaben.[460] Eine allgemeine Antwort auf die Frage, wie Menschen den Ansprüchen einer Organisation gerecht werden können, wird in dem sog. *Rollenkonzept* diskutiert:[461] STEIGER und LIPPMANN schreiben diesbezüglich, dass in Systemen einzelne Stellen mit bestimmten Positionen und bestimmten Rangordnungen durch die Aufgaben- und Machtteilung entstehen. Diejenige Person, die eine solche Position einnimmt, erhält von sog. *Rollensendern* Aufgaben, an die spezifische Erwartungshaltungen geknüpft sind. Diese Erwartungen werden als *Rolle* bezeichnet.

6.3.2 Die Rolle des Architekten

Der Entwurf von Architekturen für Informationssysteme wird durch eine eigene Rolle abgedeckt: die Rolle des IT-Architekten (bzw. im Folgenden vereinfacht *Architekt*).

DERN differenziert zwei Rollen: den Architekten auf Unternehmensebene und den Architekten auf Projektebene. Der *Architekt auf Unternehmensebene* fokussiert eine Projektübergreifende Planung und Steuerung von Architekturentwicklungen. Planungs- und Steuerungsbasis der Ausführung dieser Rolle sind die Informationsarchitektur und die IT-

[460] Vgl. Feiler/Humphrey (1992)
[461] Vgl. Steiger (1999), S. 56ff

Basisinfrastruktur. Im Gegensatz dazu hat der *Architekt auf Projektebene* die Aufgabe, auf Basis der Vorgaben des Architekten auf Unternehmensebene sowie auf Basis eines spezifischen systemtechnologischen Wissens konkrete Architekturen im Kontext einzelner Informationssysteme zu entwickeln.[462]

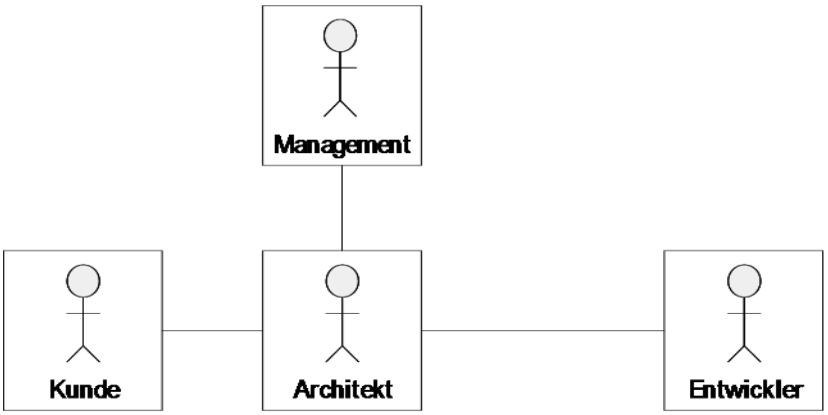


Abbildung 6.3: Der Architekt als Mediator (in Anlehnung an Keller/Wendt (2003)).

Nach KELLER und WENDT wird die Rolle des Architekten entweder von einer Person oder einem Team ausgefüllt. Mit seiner organisatorisch-zentralen Position zwischen dem Kunden, dem Management[463] und den Entwicklern übernimmt ein Architekt die Verantwortung für das Gesamtkonzept bzw. die Architektur eines Systems (Abbildung 6.3). Als Mediator übersetzt der Architekt die Elemente der Problemdomäne des Kunden in die Elemente der Lösungsdomäne der Entwickler.[464] Um das Konzept des zu entwickelnden Systems bereits in einem frühen Stadium mit dem Kunden diskutieren zu können, abstrahiert er die Funktionalität des Systems in die Sprache des Kunden. Gleichzeitig muss der Architekt die vom Kunden aufgenommenen Informationen in die Sprache des Technikers, des Entwicklers übersetzen. Er füllt somit die Lücke zwischen dem spezifischen Wissen des Technikers und dem spezifischen Wissen des Kunden, kombiniert beide Wissensdomänen und berät auf Basis dieser Kombination das Management.[465]

---

[462] Vgl. Dern (2003), S. 31
[463] Anmerkung: KELLER und WENDT definieren nicht explizit, ob die als „Management" bezeichnete Gruppe Teil der Auftraggeberseite (auf der der Kunde sitzt) oder Teil der Auftragnehmerseite (auf der die Entwickler sitzen) ist. Auf Grund ihrer Ausführungen ist jedoch davon auszugehen, dass das beschriebene Modell alle beteiligten Rollen (Kunde, Management, Architekt und Entwickler) als Teil eines Unternehmens sieht.
[464] Vgl. Dern (2003)
[465] Vgl. Keller/Wendt (2003)

6.3.3 Die Rolle des Infrastrukturarchitekten

Die Rolle des Architekten, wie sie von KELLER und WENDT bzw. von DERN beschrieben wird, wird im Kontext des SI-Katalogs mit Hilfe von zwei Teilrollen spezialisiert (Abbildung 6.4): die Rolle *Applikationsarchitekt* sowie die Rolle *Infrastrukturarchitekt*. Die Separation wird durchgeführt, da in dieser Arbeit davon ausgegangen wird, dass die beiden Wissensdomänen Infrastruktur und Applikationsentwicklung jeweils sehr komplex sind.

Die von DERN vorgeschlagene Architekturpyramide[466] differenziert zwischen der Business-Architektur, der Informationsarchitektur, der IT-Architektur und der IT-Basisinfrastruktur. Alle Aufgaben, die die IT-Basisinfrastruktur betreffen, werden im Rahmen dieser Arbeit von der Rolle des Infrastrukturarchitekten übernommen. Alle anderen Tätigkeiten fallen in das Aufgabegebiet und die Verantwortung des Applikationsarchitekten.[467]

Im Zuge einer solchen Rollenspezialisierung muss das Bild des Architekten, wie es von KELLER und WENDT gezeichnet wird, erweitert werden: Der von KELLER und WENDT präsentierte Ansatz des Architekten als Mediator sieht eine Kommunikation desselben mit denjenigen Akteuren vor, die die Rollen des Kunden, des Managements und des Entwicklers innehaben. Insbesondere der Infrastruktur von Finanzdienstleistern ist jedoch zu Eigen, dass sie auf Grund ihrer Komplexität in Rechenzentren betrieben wird.[468] Das Betreiben der Applikationen, das u. U. auch die Administration einschließt, ist eine eigenständige, logistische und betriebswirtschaftliche Disziplin. Es kann vermutet werden, dass Kosten für den Betrieb einer Infrastruktur proportional zu der Komplexität der zu betreibenden Infrastruktur steigen. Sofern es die Aufgabe der Architektur ist, Systeme unter wirtschaftlichen Gesichtspunkten zu pflegen bzw. weiterzuentwickeln, muss ein Infrastrukturarchitekt im gleichen Maße mit dem jeweiligen Betreiber der Infrast-

[466] Vgl. Dern (2003), S. 11ff
[467] Anmerkung: Inwieweit diese Aufgaben noch genauer zu differenzieren sind, wird in dieser Arbeit nicht diskutiert.
[468] Vgl. Penzel (2004)

ruktursysteme kommunizieren wie ein Applikationsarchitekt dies mit den Entwicklern praktiziert.

Somit kann die Rolle des Infrastrukturarchitekten für den weiteren Verlauf dieser Arbeit wie folgt definiert werden: Als Infrastrukturarchitekt wird diejenige Person bezeichnet, die die Rolle mit der Aufgabe des Entwurfs und der Pflege der Infrastrukturarchitektur unter Berücksichtigung wirtschaftlicher Aspekte übernimmt. Der Infrastrukturarchitekt entwirft und pflegt den SI-Katalog.

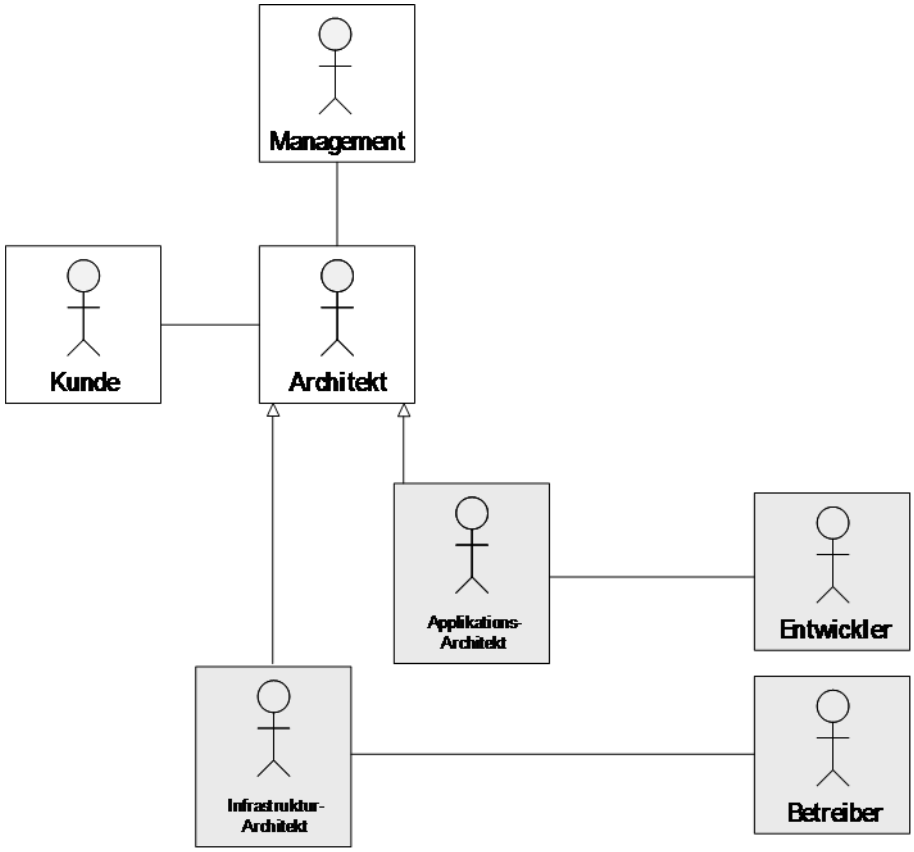


Abbildung 6.4: Die Rolle des Infrastrukturarchitekten im Kontext des SI-Katalogs. (Quelle: Eigene Darstellung)

7 Gestaltungskonzept – Einführung der Methodik

Bevor der SI-Katalog aktiv verwendet werden kann, muss er initial angelegt werden. Diese Phase wird im Kontext dieser Arbeit als *Entwurf* bezeichnet. Die Phase Gestaltungskonzept gliedert sich in die nachfolgend beschriebenen Unterprozessschritte *Identifikation der SI-Komponenten* und *Entwurf der SI-Architekturmuster* auf. Ergebnis dieses Prozessschritts ist eine Menge von Architekturmustern, die in der Phase *Anwendung* innerhalb von Software Engineering Projekten Verwendung findet.

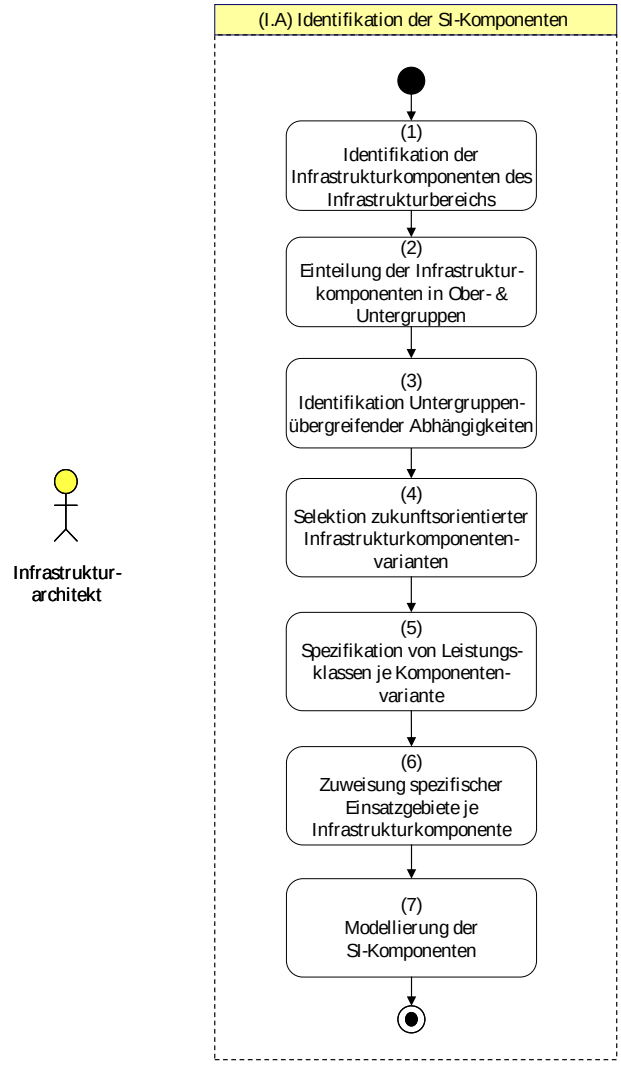


Abbildung 7.1*: Der Subprozess* Identifikation der SI-Komponenten*. (Quelle: Eigene Darstellung)*

7.1 Identifikation der SI-Komponenten

Die SI-Architekturmuster, die im Prozessschritt I.B entworfen werden, sind Kompositionen selektierter Infrastrukturkomponenten. Bevor eine derartige Komposition jedoch stattfinden kann, müssen die entsprechenden Infrastrukturkomponenten im ersten Schritt identifiziert werden, um im zweiten Schritt in der Menge dieser Infrastrukturkomponenten die SI-Komponenten selektieren zu können.
Die Identifikation der SI-Komponenten unterteilt sich in die in Abbildung 7.1 dargestellten Subprozessschritte.

7.1.1 Identifikation der Infrastrukturkomponenten des Infrastrukturbereichs

Das komplexe System Finanzdienstleistungsinformatik, in dem eine Methodik zur Komplexitätsbeherrschung der bestehenden Infrastruktur eingesetzt werden soll, und das bereits über betriebene Informationssysteme verfügt, besitzt eine bestimmte Anzahl an Infrastruktursystemen. Ein Infrastruktursystem ist eine Aggregation von Infrastrukturkomponenten, die Teil der Infrastruktur sind, und die in Summe eine spezifische Aufgabe für mindestens eine Applikation als Teil eines Informationssystems übernehmen.
Bevor eine derartige Methodik eingesetzt werden kann, müssen die zum Zeitpunkt der Analyse bekannten Infrastrukturkomponenten identifiziert worden sein. Grundlegende Voraussetzung für den Entwurf des SI-Katalogs ist demnach die in Prozessschritt A.1 stattfindende *Identifikation der Infrastrukturkomponenten des Infrastrukturbereichs*.
Per Definition kann eine Infrastrukturkomponente sowohl eine Hardware- als auch eine systemnahe Softwarekomponente sein. Infrastrukturkomponenten werden jeweils von einem Hersteller geliefert und besitzen einen eindeutigen Produktnamen. Softwarekomponenten zeichnen sich dadurch aus, dass Produkte mit gleichem Namen in unterschiedlichen Versionen ausgeliefert werden. Diese Eigenschaften führen zu dem in Abbildung 7.2 abgedruckten Ausschnitt des Meta-Modells.

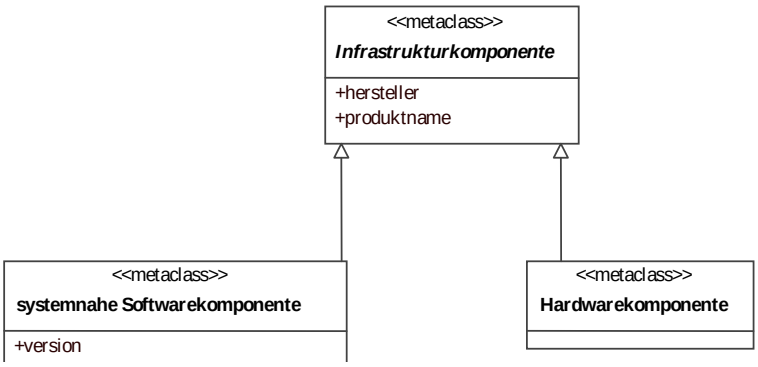


Abbildung 7.2: Ausschnitt des Meta-Modells für Prozessschritt I.A.1. (Quelle: Eigene Darstellung)

Die Erfassung der in Abbildung 7.2 dargestellten Informationen kann tabellarisch erfolgen. Ein Beispiel für eine Ist-Aufnahme sei der Inhalt der nachfolgend abgedruckten Tabelle 5.2.

#ID	Hersteller	Produktname	Version	Typ[469]
1	H1	H1/P1	V1	SW
2	H1	H1/P1	V2	SW
3	H2	H2/P1	-	HW
4	H3	H3/P1	-	HW
5	H3	H3/P2	-	HW
…	…	…	…	…

Tabelle *7.1*: In der Tabelle werden die Informationen der Ist-Aufnahme im Prozessschritt I.A.1 eingefügt.

7.1.2 Einteilung der Infrastrukturkomponenten in Ober- & Untergruppen

Die Ist-Aufnahme sämtlicher Infrastrukturkomponenten im Prozessschritt I.A.1 liefert eine Vielzahl unterschiedlicher Produkte verschiedenster Hersteller. Viele Produkte werden in einem spezifischen Bereich der Infrastruktur eingesetzt. Ein Storage-System ist beispielsweise eine Hardwarekomponente und hat die Aufgabe, Daten jeglicher Form zu speichern. Ein Betriebssystem hingegen ist eine Softwarekomponente, die den Betrieb eines Computers ermöglicht. Beide Infrastrukturkomponenten können jeweils einer Obergruppe (z. B. Storage bzw. Betriebssysteme) zugewiesen werden. Die Produkte der Obergruppe Betriebssysteme können jedoch wiederum diversen Untergruppen zuge-

[469] Anmerkung: HW = Hardwarekomponente, SW = systemnahe Softwarekomponente

wiesen werden. Unter-gruppen für Betriebssysteme können beispielsweise Linux, Unix oder Windows sein.
Eine standardisierte Einteilung der Gruppen existiert zum heutigen Zeitpunkt in der Literatur jedoch nicht.[470]

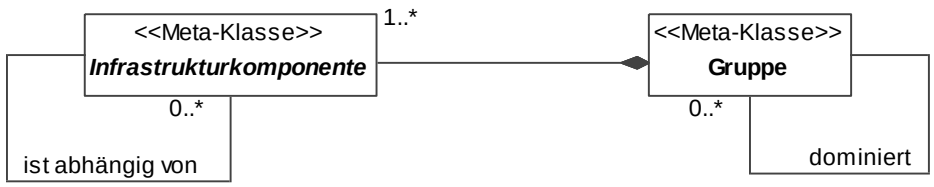


Abbildung 7.3: Ausschnitt des Meta-Modells für Prozessschritt I.A.2. (Quelle: Eigene Darstellung)

7.1.2.1 Einteilung in Ober- & Untergruppen innerhalb des Meta-Modells

Die Ansätze von LIU, DERN, PENZEL sowie GREINER ET AL. teilen eine Infrastrukturarchitektur in mehrere Ebenen auf.[471] Mit Hilfe von Layern, Schichten oder Ebenen werden einzelne Infrastrukturkomponenten separiert und in eine Rangfolge gebracht. PENZEL, DERN sowie GREINER ET AL. aggregieren Infrastrukturkomponenten zu Gruppen: Technologie-Sets, Basisplattformen und Layern.

Gruppenzugehörigkeit			**#ID**	**…**
OG_1	UG_1		1	…
	UG_2		2	…
			3	…
OG_2			4	…
OG_3	UG_1	UG_1	5	…
…	…	…	…	…

Tabelle *7.2*: Einteilung in Ober- & Untergruppen.

Auf Basis dieser Erkenntnisse wird die Identifikation von Ober- und Untergruppen als ein analytischer inkrementeller Prozessschritt aufgefasst. Die Erfassung der in Abbildung 7.3 dargestellten Informationen erfolgt tabellarisch. Wird in Prozessschritt I.A.2 auf die Tabelle 7.1 zu-

[470] Vgl. Liu (2002)
[471] Vgl. Greiner et al. (2005), Penzel (2004), Dern (2003), S. 27 sowie Liu (2002); vgl. auch Abschnitt 5.4

rückgegriffen, so kann diese um entsprechende Spalten für die Gruppenzugehörigkeit gemäß Tabelle 7.2 erweitert werden.
Sofern die Einteilung in Ober- und Untergruppen die Erkenntnis liefert, dass eine Untergruppe wiederum in Untergruppen aufgeteilt werden kann (wobei jeder Untergruppe mindestens ein Produkt zugewiesen werden muss), so ist eine weitere Untergliederung legitim. Dies führt zu dem in Abbildung 7.3 abgedruckten Ausschnitt des Meta-Modells.

7.1.3 Identifikation Untergruppenübergreifender Abhängigkeiten

Der Einsatz einer Infrastruktur-Komponente kann den Einsatz mindestens einer anderen Infrastruktur-Komponente implizieren. Ein SAP-System ist beispielsweise ohne ein Betriebssystem nicht lauffähig. Ebenso erfordert der Einsatz eines SAP-Systems grundsätzlich einen Datenbankserver. Ein Datenbankserver benötigt wiederum Festplattenspeicherplatz, der von einem Storage-System geliefert wird. Sofern diese drei Komponenten an physisch unterschiedlichen Orten betrieben werden, müssen Netzwerk-komponenten die ortsübergreifende Kommunikation gewährleisten.
So kann festgestellt werden, dass zwischen den jeweiligen Untergruppen (z. B. SAP-Systeme, Datenbankserver, Storage-Systeme, Netzwerkkompo-nenten) Abhängigkeiten existieren. Diese Abhängigkeiten lassen Dominanzen erkennen, die dazu führen, dass beispielsweise die Auswahl eines SAP-Systems gleichzeitig eine Auswahl eines Datenbanksystems beinhaltet, welches gleichzeitig eine Auswahl für ein Storage-System beinhaltet, und so weiter. Vergleichbar ist dies mit der Auswahl eines Autos, die gleichzeitig die Auswahl eines Motors (Benzin, Diesel) beinhaltet, welcher wiederum die Auswahl eines Getriebes (Automatik, Schaltgetriebe) erfordert. Durch die Kennzeichnung dominanter Gruppen wird ein hierarchischer Mechanismus bereitgestellt, der eine Top-Down-Auswahl unter den in Prozessschritt I.A.2 identifizierten Gruppen ermöglicht. Im Fall des SAP-Systems muss erst die Variante desselben ausgewählt werden, bevor mit der Auswahl des geeigneten Datenbanksystems begonnen werden kann.

7.1.3.1 Identifikation von Abhängigkeiten und Dominanzen

7.1.3.1.1 Abhängigkeitsmatrix

Das Ergebnis des Prozessschritts I.A.2 sind gruppierte Infrastrukturkom-ponenten. Um die Abhängigkeiten zwischen den einzelnen Komponenten identifizieren zu können, eignet sich eine tabellarische Gegenüberstellung der in Prozessschritt I.A.1 identifizierten Infrastrukturkomponenten.

In Tabelle 7.3 stehen sich die Infrastrukturkomponenten gruppiert gegenüber. Die Sensitivität der jeweiligen Abhängigkeit kann unterschiedliche Formen annehmen. Der in der Tabelle verwendete Platzhalter Ω darf wie folgt ersetzt werden:

- Ω = ⊙: Die Infrastrukturkomponente I1 (z. B. Betriebssystem Solaris) ist direkt abhängig von der Infrastrukturkomponente I2 (z. B. SPARC-Server). Die Inbetriebnahme von I1 ist ohne die Inbetriebnahme von I2 aus Sicht des Infrastrukturarchitekten sinnlos. Die Abhängigkeit wird im Folgenden als *Muss-Abhängigkeit* bezeichnet.
- Ω = ○: Die Infrastrukturkomponente I_1 kann zwecks ihrer Inbetriebnahme um die Inbetriebnahme der Infrastrukturkomponente I_2 erweitert werden, sofern der Einsatz von I_2 aus Sicht des Infrastrukturarchitekten sinnvoll ist. I_1 ist nicht direkt abhängig von I_2. Die Abhängigkeit wird im Folgenden als *Kann-Abhängigkeit* bezeichnet.
- Ω = [○: Diese und die nachfolgenden Infrastrukturkomponenten in horizontaler Richtung (einschließlich der mit Ω = ○] gekennzeichneten Infrastrukturkomponente) stellen eine Gruppe Γ' von Infrastruktur-komponenten dar, von denen eine Infrastrukturkomponente für die Inbetriebnahme der Infrastrukturkomponente I_1 erforderlich ist. Die Abhängigkeit wird im Folgenden als *XOR-Abhängigkeit* bezeichnet.
- Ω = ○]: Diese Infrastrukturkomponente markiert das Ende der Gruppe Γ'.
- Ω = Leer: In diesem Fall besteht keine direkte bzw. sinnvolle Abhängig-keit zwischen zwei Infrastrukturkomponenten.

Zum Verständnis der weiteren Ausführungen wird der Begriff Abhängigkeit nun wie folgt definiert: *Eine Abhängigkeit ist entweder eine Muss-, eine Kann- oder eine XOR-Abhängigkeit*.

Die Tabelle, in der die Abhängigkeiten der Infrastrukturkomponenten festgehalten werden können, besitzt nachfolgend abgedrucktes Raster. Diese Form der Abhängigkeitsidentifikation setzt voraus, dass die Abhängigkeiten nur innerhalb eines 2-diemensionalen Raums auftreten. 2-Dimensionalität bedeutet in diesem Zusammenhang, dass bei einer Infrastrukturkomponente I_1 entweder keine Abhängigkeit oder eine Muss-, Kann- bzw. XOR-Abhängigkeit mit einer Infrastrukturkomponente I_2 identifiziert werden kann. Schließt jedoch eine Abhängigkeit zwischen I_1 und I_2 die Möglichkeit der Abhängigkeit zwischen I_1 oder I_2 mit einer Infrastrukturkomponente I_3 aus, so bewegt sich diese Abhängigkeit im 3-dimensionalen Raum.

Eine weitere Voraussetzung für die Anwendbarkeit des Modells ist der Ausschluss von Zyklen: Ist eine Infrastrukturkomponente I_1 der Gruppe G_1 abhängig von der Infrastrukturkomponente I_2 der Gruppe G_2, so darf keine Infrastrukturkomponente der Gruppe G_2 eine Muss-, Kann- oder XOR-Abhängigkeit mit einer beliebigen Infrastrukturkomponente der Gruppe G_1 aufweisen.

			Ist abhängig von ▸			
			OG_1		OG_2	…
			UG_1	UG_2		…
		ID	#1	#2	#3	…
OG_1	UG_1	#1	Ω	Ω	Ω	…
	UG_2	#2	Ω	Ω	Ω	…
OG_2		#3	Ω	Ω	Ω	…
…	…	…	…	…	…	…

Tabelle *7.3*: Identifikation von Abhängigkeiten zwischen Infrastrukturkomponenten mit Hilfe der Abhängigkeitsmatrix.

Beispiel

Die Identifikation von Abhängigkeiten zwischen Infrastruktur-

komponenten wird nun an einem abstrakten Beispiel ausschnittsweise vorgeführt (vgl. Tabelle 7.4): Gegeben sind 12 Infrastrukturkomponenten (I_1 bis I_{12}), die vier Obergruppen mit je bis zu zwei Untergruppen zugeordnet worden sind.

Wie aus Tabelle 7.4 hervorgeht, besteht eine Muss-Abhängigkeit zwischen I_1 und I_4. Die Inbetriebnahme von I_1 kann optional auch um die Inbetriebnahme von I_9 ergänzt werden.

Im Fall von I_3 kann eine Muss-Abhängigkeit zwischen I_3 und I_8 festgestellt werden. Die Inbetriebnahme von I_3 kann optional auch um die Inbetriebnahme von I_9 ergänzt werden. Zudem existiert eine XOR-Abhängigkeit zwischen I_3 und einer Komponente der Menge $\{I_4, I_5, I_6\}$.

			Ist abhängig von ▸											
			OG_1			OG_2				OG_3			OG_4	
			UG_1		UG_2	UG_1			UG_2	UG_1		UG_2		
		ID	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
OG_1	UG_1	#1				⊙					○			
		#2				⊙								
	UG_2	#3				[○	○	○]		⊙	○			
OG_2	UG_1	#4										○	[○	○]
		#5										○		
		#6										○		
	UG_2	#7										○	○	⊙
OG_3	UG_1	#8				⊙								
		#9				⊙								
	UG_2	#10												
OG_4		#11												
		#12												

Tabelle *7.4*: Abstraktes Beispiel einer Identifikation von Abhängigkeiten zwischen Infrastrukturkomponenten.

7.1.3.1.2 Identifikation von Gruppendominanzen

Nach der Identifikation der Abhängigkeiten unter den Infrastrukturkomponenten ist eine Analyse bezüglich dominanter Gruppen möglich. Im Folgenden wird der Begriff *Gruppendominanz* wie folgt verwendet:

Begriffsdefinition: *Gruppendominanz*

Existiert eine Abhängigkeit zwischen einer Infrastrukturkomponente Ia einer Gruppe Gx und einer Infrastrukturkomponente Ib einer Gruppe Gy, so dominiert Gy die Gruppe Gx. Ist die Infrastrukturkomponente Ib abhängig von der Infrastrukturkomponente Ic einer Gruppe Gz, so dominiert Gz die Gruppen Gy und Gx. Die Dominanz einer Gruppe gegenüber einer anderen wird als *Gruppendominanz* bezeichnet.

Dominiert eine Gruppe G_x eine Gruppe G_y so sind die Infrastrukturkomponenten der Gruppe G_x Muss-/Kann- oder XOR abhängig von Infrastrukturkomponenten der Gruppe G_y. Somit ist G_x von G_y Muss-abhängig, wenn mindestens eine Infrastrukturkomponente der Gruppe G_x Muss-abhängig von einer Infrastrukturkomponente der Gruppe G_y ist. Ferner ist G_x von G_y Kann-abhängig, wenn keine Infrastrukturkomponente der Gruppe G_x Muss-abhängig von einer Infrastrukturkomponente der Gruppe G_y ist und wenn mindestens eine wenn Infrastrukturkomponente der Gruppe G_x Kann- oder XOR-abhängig von einer Infrastrukturkomponente der Gruppe G_y ist.

Notationstechnisch wird im Folgenden geschrieben:

$G_y < G_x$: G_y dominiert G_x

$G_y = G_x$: G_y und G_x sind gleich dominant

$G_y \circ G_x$: G_y ist Kann-abhängig von G_x

$G_y \odot G_x$: G_y ist Muss-abhängig von G_x

Die Gruppendominanzen, die durch die zuvor identifizierten Abhängigkeiten hervorgerufen werden, können durch einen Vorranggraphen identifiziert werden: Ein Vorranggraph $G = \{V, E, t\}$ ist ein gerichteter

Graph ohne Zyklen, mit einer Menge $V = \{1, \ldots, n\}$ von Knoten, einer Kantenmenge $E = \{(i, j) \mid i, j \in V\}$ und einem Vector $t \in IRn$ von Bewertungen. Hierbei nennt man zu einem Knoten i die Knoten aus $Fi = \{j \mid (i, j) \in E\}$ die Nachfolger von i und die Knoten aus $Si = \{j \mid (j, i) \in E\}$ die Vorgänger von i.[472]

Im Fall der Identifikation der Gruppendominanzen repräsentieren die Knoten des Graphen *G* die jeweils untersten Gruppen einer Obergruppe. Besitzt eine Obergruppe keine Untergruppen, so repräsentiert ein Knoten die Obergruppe. Die Kanten (i, j) stellen die Abhängigkeiten einzelner Infrastrukturkomponenten einer durch einen Knoten repräsentieren Gruppe zu Infrastrukturkomponenten einer anderen durch einen Knoten repräsentierten Gruppe. Die Bewertung *t* hat einheitlich den Wert 1. Sei I(i) die Menge aller Infrastrukturkomponenten, die der durch den Knoten i repräsentierten Gruppe zugeordnet sind und sei I(Fi) bzw. I(Si) die Menge aller Infrastrukturkomponenten, der durch die Knotenmenge Fi bzw. Si repräsentierten Gruppen.

Rang

Bezüglich der in Tabelle 7.4 identifizierten Abhängigkeiten zwischen den Infrastrukturkomponenten kann der in Abbildung 7.4 abgedruckte Vorranggraph gezeichnet werden. Die Anzahl der in einen Knoten eingehenden Kanten bestimmt die Pfadlänge von einem Knoten zu einem anderen. Die Pfadlänge wird im Folgenden als *Rang* bezeichnet. Die Ränge der Untergruppen sind:

- $Rang(OG_1/UG_1) = Rang(OG_1/UG_2) = Rang(OG_2/UG_2) = 0$
- $Rang(OG_3/UG_1) = Rang(OG_2/UG_1) = 2$
- $Rang(OG_3/UG_2) = Rang(OG_4) = 4$

Je länger der Pfad zu einem Knoten i ist, desto größer ist der Rang der von i repräsentierten Gruppe und desto größer ist die Abhängigkeit der Infrastrukturkomponenten der Menge I(Fi) von den Infrastrukturkomponenten der Menge I(i). Je kürzer der Pfad zu einem Knoten i ist, desto geringer ist der Rang der von i repräsentierten Gruppe und desto höher ist die Abhängigkeit der Infrastrukturkomponenten der Menge I(i) von den Infrastrukturkomponenten der Menge I(Si). Im Kontext der Begriffs-

[472] Vgl. Mastor (1970)

definition *Gruppendominanz* bedeutet dies, dass die Vorgänger eines Knoten i ihre Nachfolger dominieren: Eine Gruppe G_x dominiert eine andere Gruppe G_y, wenn der Rang von G_x kleiner ist als der von G_y.
Die zuvor ermittelten Ränge beziehen sich primär auf Untergruppen. Diese sind jedoch wiederum Obergruppen zugeordnet. Die Ränge der Obergruppen ermitteln sich aus den Summen der Ränge ihrer Untergruppen. Hieraus ergeben sich folgende Ränge der Obergruppen:

- $Rang(OG_1) = Rang(OG_1/UG_1) + Rang(OG_1/UG_2) = 0$
- $Rang(OG_2) = Rang(OG_2/UG_2) + Rang(OG_2/UG_1) = 2$
- $Rang(OG_3) = Rang(OG_3/UG_1) + Rang(OG_3/UG_2) = 6$
- $Rang(OG_4) = 4$

Somit ergeben sich in Bezug auf die in Tabelle 7.4 identifizierten Abhängigkeiten auf Basis der ermittelten Ränge folgende Untergruppendominanzen: (OG1/UG1 = OG1/UG2 = OG2/UG2) < (OG3/UG1, OG2/UG1) < (OG3/UG2) = (OG4). Als Obergruppendominanzen ergeben sich: (OG1) < (OG2) < (OG4) < (OG3).

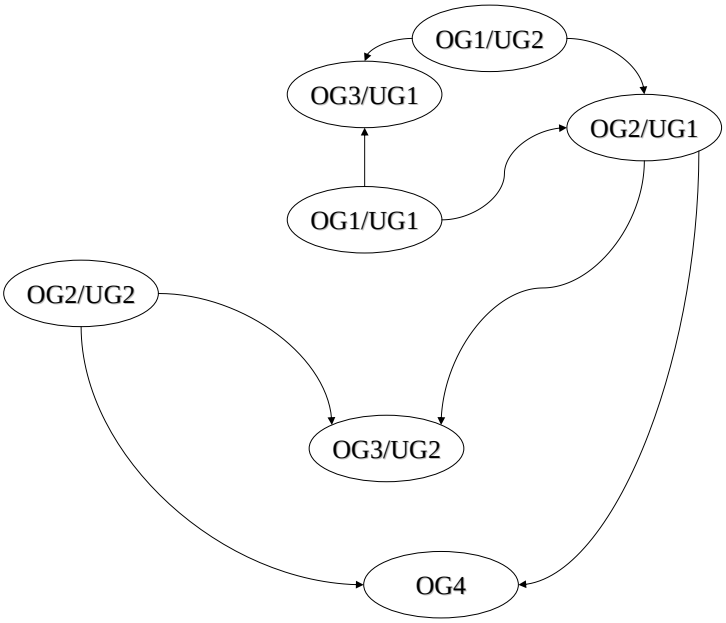


Abbildung 7.4: Vorranggraph der in Tabelle *7.4 identifizierten Abhängigkeiten zwischen den Infrastrukturkomponenten. (Quelle: Eigene Darstellung)*

Die Identifikation der Gruppendominanzen ist für die spätere Auswahl von Infrastrukturkomponenten innerhalb des Architekturentwurfs relevant. Durch die Berücksichtigung der Gruppendominanzen stellt der Infrastrukturarchitekt eine vorsortierte Auswahl von Infrastrukturkompo-

nenten zur Verfügung. Durch die identifizierten Abhängigkeiten der Infrastrukturkomponenten wird die Menge der infrage kommenden und mit der ausgewählten Komponente kombinierbaren Infrastrukturkomponenten reduziert. Das Wissen bezüglich der Zusammenhänge von Infrastrukturkomponenten ist somit implizit in den SI-Katalog integriert.

7.1.3.2 Meta-Modell

Auf Basis dieser Erkenntnisse kann das Meta-Modell wie folgt abgeändert werden (Abbildung 7.5): Ist eine Infrastrukturkomponente abhängig von einer oder mehreren spezifischen Infrastrukturkomponenten, so ist auch diese Abhängigkeit eine wesentliche Information, die in den Architekturentwurf mit einfließen muss. Ist eine Infrastrukturkomponente einer Gruppe G1 direkt abhängig von einer Infrastrukturkomponente einer anderen Gruppe G2, so muss G1 die Gruppe G2 dominieren.

7.1.4 Selektion zukunftsorientierter Infrastrukturkomponentenvarianten

Eine Variante einer Hardware- bzw. Software-technischen Komponente ist eine andere Hardware- bzw. Software-technische Komponente gleichen Zwecks, die sich in mindestens einer Eigenschaft oder einer Relation zu einer anderen Komponente unterscheidet. Ein Beispiel für eine Hardware-Komponentenvariation ist ein Server, der entweder eine SPARC- oder eine Intel-basierte Prozessorarchitektur verwendet. Der Zweck beider Varianten ist der technische Betrieb eines Softwaresystems. Die Eigenschaften, die sich bei beiden Varianten unterscheiden, sind beispielsweise der Hersteller oder die Prozessorarchitektur.
Bei Software-Komponenten ist es üblich, dass deren Hersteller mit der Zeit neue Versionen seiner Produkte ausliefert. Bei der Ist-Aufnahme der existierenden Infrastrukturkomponenten in Prozessschritt I.A.1 ist es wahrscheinlich, dass ein- und dieselbe Software-Komponente in unterschiedlichen Versionen im Einsatz ist. Es kann daher durchaus sinnvoll sein, neu zu implementierende Informationssysteme nur mit der gültigen letzten Version einer Software-Komponente zu realisieren. Unterschiedliche Versionen einer Software-Komponente sind per Definiti-

on Varianten einer Software-Komponente, da sich die Eigenschaften einer Software-Komponente von Version zu Version ändern. Im Rahmen einer *Selektion zukunftsorientierter Infrastrukturkomponentenvarianten* im Prozessschritt I.A.3 ist demzufolge auch die Aktivität der Selektion eventuell vorhandener Versionen von Softwarekomponenten eingeschlossen.

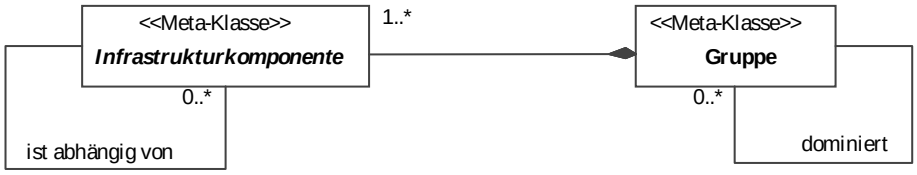


Abbildung 7.5: Ausschnitt des Meta-Modells. (Quelle: Eigene Darstellung)

Bei Hardware-Komponenten werden in der Regel keine Versionsnummern sondern neue Namen für Produktnachfolger vergeben. Auch in diesem Fall handelt es sich per Definition um Varianten einer Hardware-Komponente, die innerhalb des Selektionsprozess analysiert werden.

Warum eine Selektion?

Bei der im Prozessschritt I.A.1 stattgefundenen *Identifikation der Infrastrukturkomponenten des Infrastrukturbereichs* wurden unterschiedliche Software- und Hardware-Komponenten identifiziert, die sich zum Zeitpunkt der Ist-Aufnahme im Betrieb befanden. Viele der Komponenten könnten nach Ansicht des analysierenden Infrastrukturarchitekten als „veraltet“ gelten und/oder „strategisch unerwünscht“ sein.[473] Hierzu drei Szenarien:

- Für eine Software-Komponente I werden drei Varianten Version a, b und c (mit a < b < c) identifiziert. Es ist erwünscht, für zukünftige Entwicklungsaufgaben nur die aktuellste Version (also c) zu verwenden. Daher müssen die Komponentenvarianten in Form der Versionen a und b entsprechend als „veraltet“ kenntlich gemacht werden. Gleiches gilt auch für Hardware-Komponenten.
- Die Identifikation der Infrastrukturkomponenten stellt die Existenz dreier Komponentenvarianten eines Betriebssystems fest: I_1, I_2 und I_3. Auf Grund einer strategischen Entscheidung (z. B. auf Basis einer Markttendenz) sollen alle zukünftigen Applikationen nur noch die Betriebssysteme I_2 und I_3 verwenden. Aus diesem Grund muss die

[473] Vgl. Penzel (2004), S. 127

Komponentenvariante I_1 als „strategisch unerwünscht“ gekennzeichnet werden.

> Eine Infrastrukturkomponentenvariante der Hersteller 1 und 2 wird identifiziert. Auf Basis einer strategischen Entscheidung (z. B. auf Grund des hohen Insolvenzrisikos eines Herstellers) ist es erwünscht, für zukünftige Entwicklungsaufgaben nur noch die Produkte von Hersteller 2 einzusetzen. Daher muss die Infrastrukturkomponente von Hersteller 1 als „strategisch unerwünscht“ gekennzeichnet werden.

7.1.4.1 Selektionsmatrix

Im Rahmen des Entwurfs des SI-Katalogs werden in Prozessschritt I.A.1 die zum Zeitpunkt der Analyse bekannten Infrastrukturkomponenten identifiziert. Die drei zuvor abgedruckten Szenarien zeigen jedoch, dass eine Selektion von Infrastrukturkomponenten sinnvoll ist.

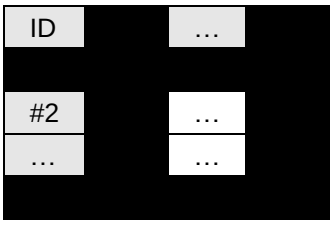
Eine Selektion umfasst zwei Tätigkeiten: zum einen den gezielten Ausschluss und zum anderen die gezielte Bevorzugung einzelner Infrastrukturkomponenten. Ein Ausschluss bzw. eine Bevorzugung kann sowohl auf ganze Infrastrukturkomponenten als auch gezielt auf Abhängigkeiten zwischen Infrastrukturkomponenten angewendet werden. Um die Modifikationen an den Ergebnissen des Prozessschritts I.A.3 transparent zu machen, wird der Hintergrund modifizierter Zeilen und Spalten der Abhängigkeitsmatrix (Tabelle 7.4) geschwärzt.

Ausschluss & Bevorzugung

Die vier in Frage kommenden Fälle einer Selektion werden nun am Beispiel der in Tabelle 7.4 abgetragenen Inhalte der Abhängigkeitsmatrix untersucht:

> *Ausschluss einer Infrastrukturkomponente*: Wird eine Infrastrukturkomponente von ihrem Einsatz ausgeschlossen, so werden ihre Zeile und ihre Spalte schwarz markiert. Die eventuell in den markierten Zellen vorhandenen Abhängigkeitssymbole bleiben zum Zweck einer späteren Nachprüfbarkeit bestehen. Der nachfolgende Ausschnitt der Tabelle 7.5 zeigt den Ausschluss der Infrastrukturkomponenten I1 und I9. Der Ausschluss signalisiert, sowohl I1 als auch I9 in neuen Architekturentwürfen keine Verwendung mehr finden dürfen. Durch

die zuvor identifizierten Abhängigkeiten wird dem Infrastrukturarchitekten transparent gemacht, ob der Ausschluss einer Infrastrukturkomponente überhaupt Sinn macht. Im Fall von I3 wurde eine Muss-Abhängigkeit zwischen I3 und I8 festgestellt. Das bedeutet, dass für den Betrieb der Komponente I3 die Inbetriebnahme von I8 erforderlich ist (Muss-Abhängigkeit). Würde die I8 innerhalb des Selektionsprozesses ausgeschlossen, könnte I3 nicht in Betrieb genommen werden, sofern für I8 keine Alternative Infrastrukturkomponentenvariante zur Verfügung stünde (XOR-Abhängigkeit). Existiert eine Muss-Abhängigkeit zwischen einer Infrastrukturkomponente Ix und einer auszuschließenden Infrastrukturkomponente Iy, so muss entweder Ix ebenfalls ausgeschlossen werden oder von einem Ausschluss der Infrastrukturkomponente Iy ist abzusehen.



> *Bevorzugung einer Infrastrukturkomponente*: Wird eine Infrastrukturkomponente bevorzugt, so werden ihre Zeile und ihre Spalte gelb markiert. Die eventuell in den markierten Zellen vorhandenen Abhängigkeitssymbole bleiben wie im Fall eines Ausschlusses bestehen. Wird eine Infrastrukturkomponente in Form mehrerer Varianten Ix (mit $x \in IN>0$) der Menge V vertreten, kann eine strategische Entscheidung einzelne Infrastrukturkomponenten aus V gegenüber ihren Varianten zwecks ihrer Inbetriebnahme bevorzugen. Die Varianten Ix aus V, die nicht bevorzugt werden, können nur dann ausgeschlossen werden, wenn keine andere Infrastrukturkomponente durch eine Muss-Abhängigkeit mit ihnen verknüpft ist. Nachstehend abgedruckten Fall betrachtend soll $I5 \in V$, mit $V = \{I4, I5, I6\}$, strategisch den Varianten I4 und I6 vorgezogen werden. Da I6 strategisch ausgeschlossen wurde, stehen nur noch die Alternativen I4 und I5 zur Auswahl. Durch die Bevorzugung von I5 wird die XOR-Abhängigkeit

zwischen I3 und I4 ausgeschlossen und geht in eine Muss-Abhängigkeit zwischen I3 und I5 über.

ID		#2	#3	#4	#5
#2				⊙	
#3				[○	○
#4					
#5					

- *Ausschluss einer Abhängigkeit zwischen zwei Infrastrukturkomponenten*: Sofern keine Infrastrukturkomponente von ihrer Inbetriebnahme ausgeschlossen werden soll, sondern lediglich ihre Wiederverwendung in Abhängigkeit einer anderen Infrastrukturkomponente betroffen ist, so ist die Zelle der Abhängigkeit entsprechend zu markieren. Betrifft der Ausschluss wie nachfolgend dargestellt eine XOR-Abhängigkeit, in der zwei Infrastrukturkomponentenvarianten zur Auswahl stehen, so erhält die nicht ausgeschlossene Abhängigkeit automatisch eine Muss-Abhängigkeit. Ein Beispiel für einen derartigen Fall ist die Abhängigkeit von I4 und I12.

ID	…	#11	#12
…	…	…	…
#4	…	[○	
#5			

- *Bevorzugung einer Abhängigkeit zwischen zwei Infrastrukturkomponenten*: Ist eine Infrastrukturkomponente Ix (mit x $\in$ IN>0) von einer Infrastrukturkomponente abhängig, welche durch mehrere Varianten Iy (mit x $\neq$ y und y$\in$ IN>0) der Menge V vertreten ist, und ist die Abhängigkeit zwischen Ix und den Varianten als XOR-Abhängigkeit identifiziert worden, kann eine strategische Entscheidung einzelne Abhängigkeiten zwischen Ix und Iy aus V bevorzugen. In diesem Fall sind die sind die nicht-strategischen Abhängigkeiten auszuschließen.

Dokumentation von Ausschlüssen

Wird eine Infrastrukturkomponente ausgeschlossen bzw. bevorzugt, ist der Grund für eine solche Entscheidung transparent festzuhalten. Die

Form der Dokumentation ist frei wählbar und nicht Bestandteil des SI-Katalog Entwurfs.

			Ist abhängig von ▸											
			OG_1			OG_2				OG_3			OG_4	
			UG_1		UG_2	UG_1			UG_2	UG_1		UG_2		
		ID		#2	#3	#4	#5		#7	#8		#10	#11	#12
OG_1	UG_1													
		#2				⊙								
	UG_2	#3				[○	○			⊙				
OG_2	UG_1	#4										○	[○	
		#5										○		
	UG_2	#7										○	○	⊙
OG_3	UG_1	#8				⊙								
	UG_2	#10												
OG_4		#11												
		#12												

Tabelle *7.5*: Selektionsmatrix der zukunftsorientierten Infrastrukturkomponenten auf Basis der Ergebnisse aus *Tabelle 7.4*.

7.1.4.2 Meta-Modell

Die drei zuvor vorgestellten Selektionsszenarien führen zu einer Erweiterung des Meta-Modells (Abbildung 7.6). Um das Modell möglichst flexibel zu gestalten, soll der Entscheidungsgrund für oder gegen eine Infrastrukturkomponente variabel definiert werden können. Eine Infrastrukturkomponente ist daher entweder Teil einer Positiv-Liste, zu der alle „zukunftsorientierten“ Komponenten gehören, oder sie ist Teil einer Negativ-Liste, zu der alle ausgeschlossenen Komponenten zählen, deren Einsatz nicht erwünscht ist. Falls der Einsatz einer Infrastrukturkomponente nicht erwünscht ist, so ist dies mittels einer Bemerkung kenntlich zu machen. Gleiches gilt für den Ausschluss einer Abhängigkeit.

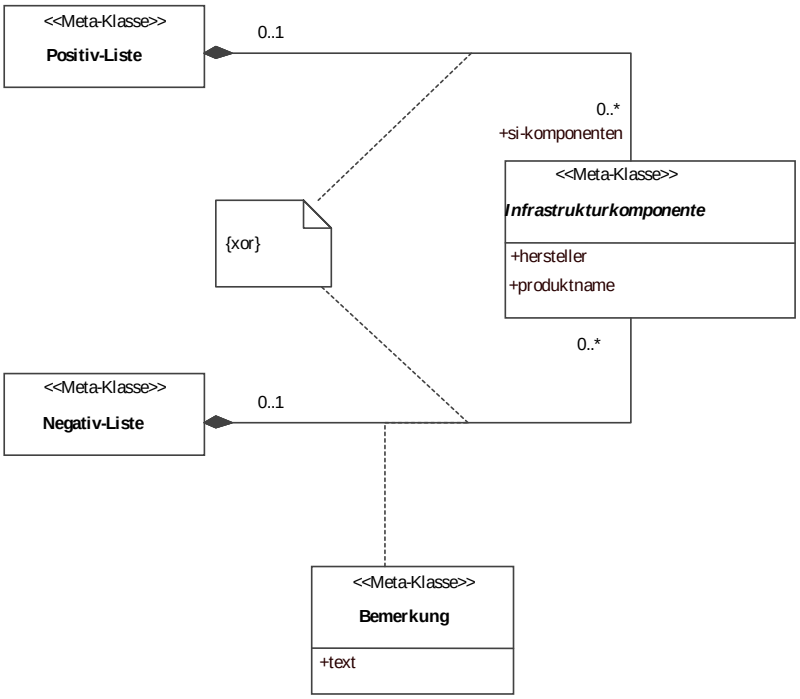


Abbildung 7.6: Ausschnitt des Meta-Modells. (Quelle: Eigene Darstellung)

Infrastrukturkomponenten, deren Einsatz zukünftig erwünscht ist, gelten für die Organisation als standardisiert. Daher werden alle Komponenten, die innerhalb des Selektionsprozesses der Positiv-Liste zugeordnet wurden, im Folgenden als Standard-Infrastrukturkomponenten (oder kurz SI-Komponenten) bezeichnet.

7.1.5 Spezifikation von Leistungsklassen je Komponentenvariante

Jede Infrastrukturkomponente ist ein Produkt eines Herstellers und gehört zu genau einer Gruppe. Eine häufige Eigenschaft von Produkten ist, dass sie in nahezu derselben Ausführung von unterschiedlichen Herstellern bezogen werden können. So kommt es vor, dass ein Hersteller bereits qualitativ unterschiedliche Produkte einer Gruppe am Markt anbietet. Verschiedene Hersteller von Storage-Systemen liefern beispielsweise ihre Produkte in den 3 vordefinierten Leistungsklassen *Enterprise*, *Mid Range* und *Low End* aus. Qualitatives Unterscheidungsmerkmal ist in diesem Fall die Geschwindigkeit, mit der Daten von dem Storage-System gelesen werden bzw. auf das Storage-System geschrieben werden können.

Um die Produktvarianten einer jeweiligen Gruppe qualitativ unterscheiden zu können, werden je Leistungsklasse einer Gruppe für alle in der Gruppe enthaltenen Produktvarianten eindeutige Leistungsattribute benötigt. Hierbei kann es sich sowohl um eine nicht weiter unterteilbare Gruppe als auch um eine Gruppe mit Untergruppen handeln. Jedes Leistungsattribut ist Teil einer Leistungsklasse. Umgekehrt besitzt jede Leistungsklasse ihre eigene Menge an Leistungsattributen. Da jede Infrastrukturkomponente

n-viele Leistungsattribute besitzt, bzw. umgekehrt jedem Leistungsattribut m-viele Infrastrukturkomponenten zugewiesen werden können, muss jede Beziehung zwischen diesen beiden Entitäten durch eine spezifische Leistungsausprägung definiert sein.

Ziel des Prozessschritts I.A.4 ist eine qualitative Bewertung grundsätzlich vergleichbarer SI-Komponenten. Als grundsätzlich vergleichbar sind die Infrastrukturkomponentenvarianten einer Gruppe anzunehmen. Eine entsprechende *Einteilung der Infrastrukturkomponenten in Ober- & Unter-gruppen* hat in Prozessschritt I.A.2 stattgefunden.

Der Vorteil einer Einteilung von Infrastrukturkomponentenvarianten in Leistungsklassen ist eine Abstraktion und gleichzeitige Gruppierung von Herstellerspezifika. Sei der Fall angenommen, dass die Hersteller 1, 2 und 3 einer Infrastrukturkomponente I die Varianten I_1, I_2 und I_3 anbie-

ten. Jeder Infrastrukturkomponente der Menge M = {I_1, I_2, I_3} kann eine Menge Q mit Leistungsattributen zugewiesen werden. Ein Leistungsattribut ist eine spezifische, qualitative Eigenschaft, die allen Infrastrukturkomponenten der Menge M wertmäßig in Form einer Leistungsausprägung zugewiesen werden kann. Durch ein Clustering der Leistungsausprägungen können Leistungsklassen für M identifiziert werden, denen Elemente der Menge M eindeutig zugewiesen werden können. Legitimes Ergebnis eines solchen Clusterings sind 1 bis |M| viele Leistungsklassen.

Die Reduktion der Eigenschaftenmenge einer Infrastrukturkomponente auf eine Menge qualitativer Eigenschaften findet sich auch bei KRCMAR:[474] Der Grund für eine derartige Reduktion auf Qualitätsattribute ist das Aneignen komplexen, technischen Wissens, dessen Aneignung nur dann ratsam ist, wenn das Wissen für das Unternehmen überhaupt relevant ist. Zudem haben die Entscheider, die sich für oder gegen eine Technologie bzw. eine Infrastrukturkomponentenvariante entscheiden, meist keine Zeit, regelmäßig das Wissen bzgl. der IKT aufzufrischen. In den meisten Fällen ist es völlig ausreichend, wenn Entscheider die qualitativen Eigenschaften dieser Komponenten kennen und voneinander unterscheiden können.

7.1.5.1 Spezifikation der Leistungsklassen

Die Spezifikation der Leistungsklassen ist ein analytischer Prozess. Vorausgesetzt, dass im Prozessschritt I.A.2 grundsätzlich vergleichbare Infrastrukturkomponentenvarianten ein- und derselben Gruppe zugeordnet wurden, können den Varianten auf Grund ihrer Vergleichbarkeit qualitative Eigenschaften zugewiesen werden. Qualitative Eigenschaften sind gruppenspezifische Leistungsattribute, die bei jeder Infrastrukturkomponente, die dieser Gruppe zugeordnet wurde, individuell ausgeprägt sind.[475]

Die Zuweisung gruppenspezifischer Leistungsattribute und die Identifikation der Infrastrukturkomponenten-individuellen Leistungsausprägungen ist eine Entscheidungsunterstützung für nachfolgende Planungs-

[474] Vgl. Krcmar (2005), S. 240ff
[475] Vgl. Penzel (2004), S. 124f

prozesse. Zu den Mechanismen komplexer Entscheidungsfindungen zählen Matrix- und Priorisierungsdiagramme, die Teil der sog. *Seven Management and Planning Tools* sind:[476] Ein Matrixdiagramm, das spezialisiert in Form einer Priorisierungsmatrix eingesetzt wird, stellt die Korrelation zwischen zwei Typen von Informationen dar. Die Ausprägung dieser Verbindung, die in der Regel mit den Werten 0, 1, 3 und 9 quantifiziert wird, unterstützt die Priorisierung eines der zwei Informationstypen (die horizontal abgetragenen Informationen) gegenüber den gewichteten Elementen des anderen Informationstyps (die vertikal abgetragenen Informationen).

Beispiel einer Priorisierungsmatrix

Als Beispiel für eine derartige Priorisierungsmatrix soll die in Tabelle 7.6 abgedruckte Gegenüberstellung der SI-Komponenten Iy (y = 1, …, n) der Gruppe OG2 mit den Leistungsattributen LAx (x = 1, …, m) dienen. Die Leistungsattribute LAx können unterschiedliche Relevanz besitzen und werden daher mit Hilfe der Gewichte py priorisiert. Die Korrelation zwischen Iy und LAx wird durch die Gewichte pxy ausgedrückt. HERZWURM ET AL. schlagen als zulässige Wertemenge w_{xy} = {0, 1, 3, 9} vor. Je stärker ein Leistungsattribut LA_x einer Infrastrukturkomponente I_y ausgeprägt ist, desto größer ist ihre Korrelation w_{xy}. Das absolute Gewicht $aw(I_y)$ der SI-Komponenten errechnet sich aus $aw(I_y) = w_{xy} * p_y$. Das relative Gewicht $rw(I_y)$ der SI-Komponenten errechnet sich aus $rw(I_y) = aw(I_y) / \Sigma[aw(I_y)]$.

Die Identifikation der gesuchten Leistungsklassen wird durch die relativen Gewichte $rw(I_y)$ unterstützt. Ergebnis dieses Identifikationsprozesses dürfen maximal drei Leistungsklassen sein. Die Einteilung der Infrastrukturkomponenten I_y in Leistungsklassen ist ein subjektiver Prozess, der vom Infrastrukturarchitekten durchgeführt wird. Auf Basis der Verhältnisse zwischen den berechneten relativen Gewichten kann eine subjektive Klassenbildung vorgenommen werden. Das Ergebnis der Priorisierungsmatrix in Tabelle 7.6 sind die relativen Gewichte rw = {0,39; 0,39; 0,22}. Da die Gewichte von $rw(I_4)$ und $rw(I_5)$ identisch sind und einen wesentlichen Abstand zu $rw(I_7)$ aufweisen, kann der Infrastrukturarchitekt sich dazu entscheiden, die beiden Infrastrukturkompo-

[476] Vgl. Herzwurm et al. (2000), S. 21

nenten I_4 und I_5 der Leistungsklasse LK_1 und die Infrastrukturkomponente I_7 der Leistungsklasse LK_2 zuzuweisen. Wären jedoch die Gewichte rw = {0,33; 0,33; 0,33} bzw. rw = {0,1; 0,35; 0,55} das Ergebnis dieses Prozesses, so wäre eine Einteilung in eine bzw. in drei Leistungsklassen denkbar.

			Leistungsattribute ▸				
			LA1	**LA2**	**LA3**	aw	rw
		Gewicht:	0,6	0,3	0,1	1	
Gruppe ▾							
OG2	UG1	**#4**	9	3	3	6,6	0,39
		#5	9	1	9	6,6	0,39
	UG2	**#7**	3	9	9	3,6	0,22
					Σ	16,8	1

Tabelle *7.6*: Priorisierungmatrix der Infrastrukturkomponenten und der Leistungsattribute.

7.1.5.2 Meta-Modell

Die Feststellungen des vorherigen Abschnitts führen zu einer Erweiterung des Meta-Modells gemäß Abbildung 7.7:

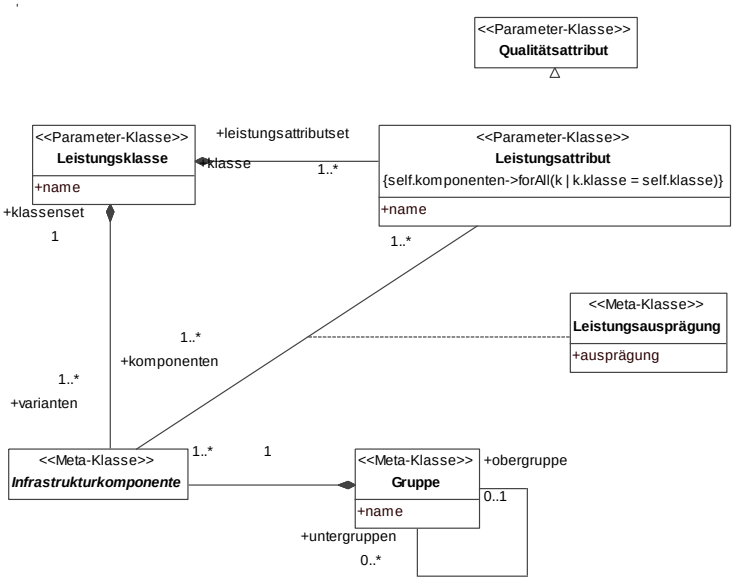


Abbildung 7.7: Ausschnitt des Meta-Modells. (Quelle: Eigene Darstellung)

7.1.6 Zuweisung spezifischer Einsatzgebiete und Regeln

Jede Infrastrukturkomponente eignet sich für ein oder mehrere spezifische Einsatzgebiete (Abbildung 7.8). Ein Storage-System beispielswei-

se eignet sich für die physische Speicherung von Daten. Microsoft Windows 2000 wird vornehmlich zur Steuerung Pentium-kompatibler Prozessoren verwendet. Die verbale Beschreibung derartiger spezifischer Einsatzgebiete soll dazu dienen, Außenstehenden ein kurzes aber prägnantes Wissen über die betrachtete SI-Komponente zu vermitteln.

Neben der Zuweisung eines spezifischen Einsatzgebietes werden in diesem Prozessschritt auch spezifische Regeln mit der SI-Komponente verknüpft. Aus Sicht des Infrastrukturarchitekten können strategische Entscheidungen ein dominantes Kriterium für oder gegen eine SI-Komponente darstellen. Beispielsweise kann strategisch entschieden werden, dass alle Applikationen eines Informationssystems, das Kundendaten verarbeitet, auf einem ausgewählten Betriebssystem ausgeführt werden müssen. Angenommen die beiden SI-Komponenten I_1 und I_2 der Gruppe G sind grundsätzlich vergleichbar. In Prozessschritt A.5 wurde ihnen das Leistungsattribut LA mit den Ausprägungen I_1.LA und I_2.LA (mit I_2.LA > I_1.LA) zugewiesen. Auch für den Fall, dass die qualitativen Anforderungen an eine geeignete SI-Komponente der Gruppe G für die Auswahl von I_2 sprechen, kann eine strategische Entscheidung den Einsatz von I_1 verlangen.[477]

Das Meta-Modell des SI-Katalogs verlangt die Zuweisung strategischer Entscheidungen in Form von Regeln: Grundsätzlich wird zwischen zwei Typen von Regeln unterschieden. Die *Default Regel* beschreibt die Selektion einer SI-Komponente anhand der qualitativen Ausprägungen. Die *Strategische Regel* ist eine im Einzelfall speziell zu formulierende Regel, die den Einsatz der assoziierten SI-Komponente mit Bedingungen verknüpft. Ein Beispiel für eine *Strategische Regel* ist die Bedingung: „Diese Komponente wird ausschließlich in Testsystemen eingesetzt." Jeder SI-Komponente ist mindestens eine Regel zugeordnet. Wird einer Komponente keine *Strategische Regel* zugeordnet, so wird sie mit Hilfe der *Default Regel* qualitativ beurteilt.

---

[477] Anmerkung: Angenommen, die Infrastrukturarchitekten bewerteten das Leistungsattribut *Performance* des JREs der Firma BEA qualitativ mit einem Wert von 9. Die Variante in Form des JREs der Firma Sun Microsystems wurde bzgl. desselben Leistungsattributs qualitativ mit dem Wert 3 bewertet. Obwohl eine Anforderung an ein neu einzusetzendes JRE den Wert 9 aufweist und somit die Variante der Firma BEA qualitativ die geeignetere wäre, kann eine strategische Entscheidung die Variante der Firma Sun Microsystems bevorzugen. Die in der Anforderung geforderte Performance kann in diesem Fall durch größere Hardwareaufwände realisiert werden.

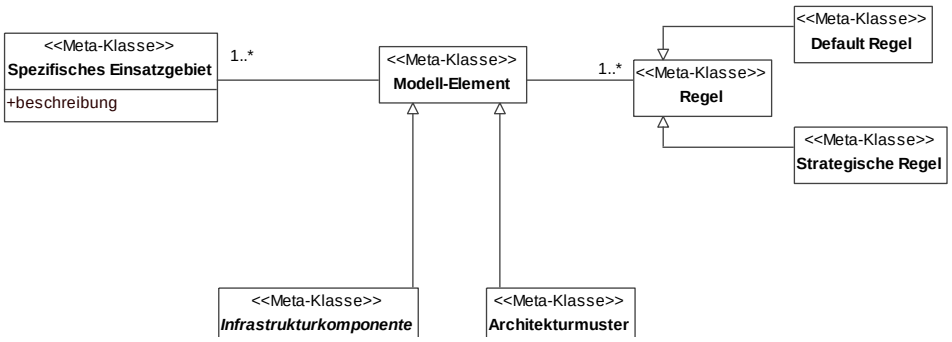


Abbildung 7.8: Ausschnitt des Meta-Modells. (Quelle: Eigene Darstellung)

7.1.7 Modellierung der SI-Komponenten

Eine Wissensreduktion zur Entlastung der kognitiven Komplexität der Infrastruktur kann durch die Verwendung von Symbolen realisiert werden.[478] Auf Basis der Spezifikation des Meta-Modells auf der Meta-Ebene M2 kann mit der Modellierung der konkreteren Modellebene M1 begonnen werden. Das Ergebnis des Prozessschritts I.A.4 sind selektierte SI-Komponenten und deren jeweilige Abhängigkeiten. Alle SI-Komponenten, die innerhalb des Selektionsschritts der Positiv-Liste zugeordnet worden sind, sollen in einem Software-Engineering Prozess eingesetzt werden können.

Die Komplexität der Infrastruktur ergibt sich zum einen aus der Vielfalt der in einem Software-Engineering Prozess verwendbaren Infrastrukturkomponentenvarianten und zum anderen aus der Vielfalt ihrer Kombinierbarkeit. Das Ergebnis des Prozessschritts I.A.2 sind in Ober- und Untergruppen eingeteilte Infrastrukturkomponentenvarianten, deren Anzahl in Prozessschritt I.A.4 reduziert werden kann. Die Durchführung der Prozessschritte I.A.5 und I.A.6 ergänzt das Meta-Modell der Ebene M2 um die Zuweisung qualitativer Eigenschaften zu den selektierten SI-Komponenten. In der Modellebene M1 werden die Meta-Klassen der Ebene M2 instanziiert. Aufgabe dieses Prozessschritts soll es sein, alle selektierten SI-Komponenten in Ebene M1 zu erfassen und sie mit Hilfe der qualitativen Eigenschaften zu klassifizieren.

Bevor die Klassifizierung der SI-Komponenten durchgeführt wird, müssen diese mit Hilfe von Klassen spezifiziert werden. Um eine visuelle

[478] Vgl. Wegner (2002)

Trennung der Modellebene M2 von der Modellebene M1 sicherstellen zu können, wird der Stereotyp *<<Design-Klasse>>* eingeführt, der in diesem Prozessschritt allen Klasseninstanzen der Meta-Klassen *Gruppe* und *Infrastrukturkomponente* (bzw. deren Spezialisierung auf der Modellebene M2) zugewiesen wird. Da die Abhängigkeiten zwischen den einzelnen SI-Komponenten als Komplexitätstreiber identifiziert wurden, werden hieraus resultierende Assoziationen zwischen den Klassen der SI-Komponenten in diesem Prozessschritt explizit nicht modelliert.

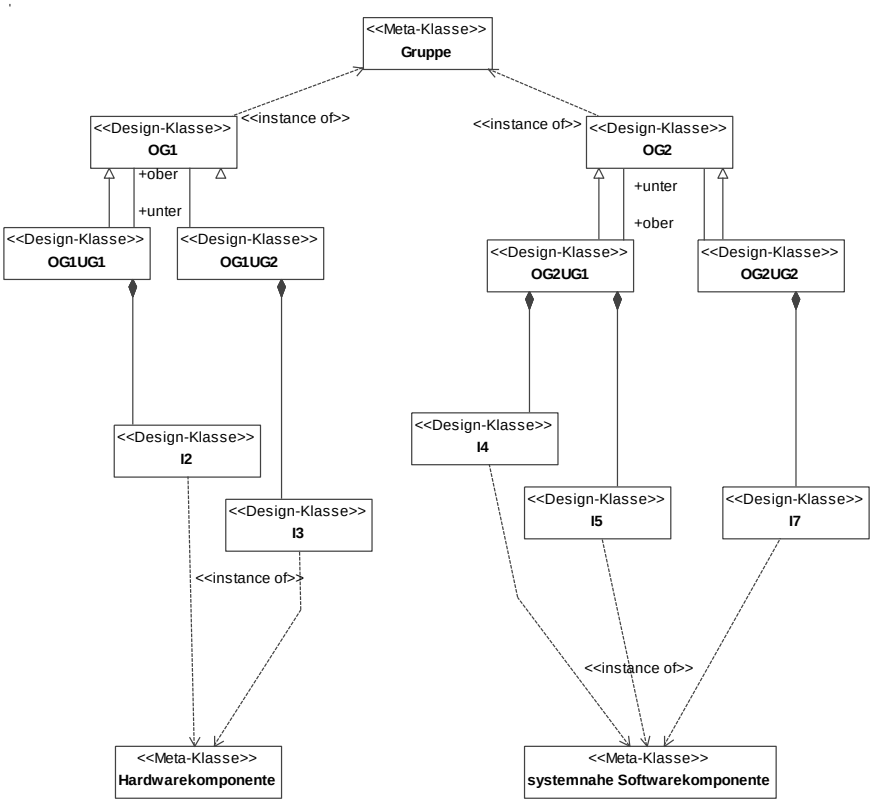


Abbildung 7.9: Definition der Gruppen und SI-Komponenten sowie Einteilung der SI-Komponenten in Gruppen gemäß den Ergebnissen aus Tabelle *7.5. (Quelle: Eigene Darstellung)*

Abstraktes Beispiel

In Anlehnung an die Ergebnisse aus Tabelle 7.5 wird nun die Modellierung der SI-Komponenten am Beispiel der Obergruppen OG1 und OG2 vorgeführt (vgl. Abbildung 7.9): Die SI-Komponenten I_2 und I_3 sind Teil der Untergruppen UG1 und UG2, welche wiederum der Obergruppe OG1 zugeordnet wurden. OG_1 ist eine Instanzklasse der Meta-Klasse *Gruppe*. Um eine Eindeutigkeit der Untergruppen von OG1 und OG2 zu erzielen, werden die Namen OG_1UG_1 und OG_1UG_2 vergeben. Das Meta-Modell erfordert eine Assoziationsbeziehung zwischen OG_1 und den

zugeordneten Untergruppen. Um kenntlich zu machen, dass OG_1UG_1 und OG_1UG_2 Instanzklassen der Meta-Klasse *Gruppe* sind, werden sie von OG_1 abgeleitet. Die Zugehörigkeit von I_2 zu OG_1UG_1 und I_3 zu OG_1UG_2 wird in Anlehnung an das Meta-Modell durch eine Komposition gekennzeichnet. Als Instanzen einer Infrastrukturkomponente der Meta-Ebene M2 besitzen I_2 und I_3 den Stereotyp *Design-Klasse*. Analog verläuft die Modellierung der Obergruppe OG_2 und der ihr zugehörigen SI-Komponenten.

Die Zuordnung der Klassen zu einer Positiv-Liste wird nicht modelliert, da die Anforderung des Prozessschritts lautet, nur die selektieren SI-Komponenten zu verwenden. Eine solche Zugehörigkeit ist also implizit anzunehmen.

Ergebnis des Prozessschritts A.5 ist eine qualitative Beschreibung und Typisierung der SI-Komponenten mittels Leistungsklassen und –Ausprägungen. Zusammen mit den in Prozessschritt A.6 definierten spezifischen Einsatzgebieten einer jeden Infrastrukturkomponente stellen diese Informationen qualitative Ausprägungen dar, die als solche mit dem gleichnamigen Stereotyp gekennzeichnet werden. Die Modellierung dieser qualitativen Ausprägungen auf Basis der Ergebnisse der Tabelle 7.6 ist in Abbildung 7.10 am Beispiel der SI-Komponenten I4, I5 und I7 dargestellt.

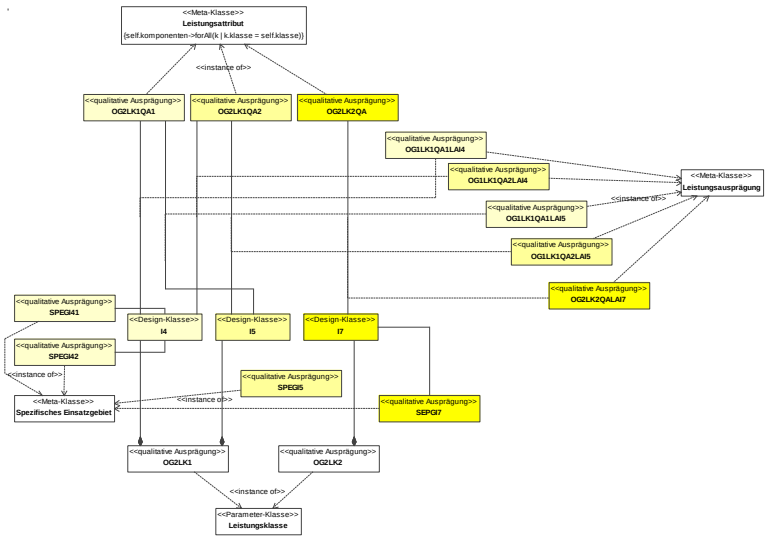


Abbildung 7.10: Einteilung der SI-Komponenten in Leistungsklassen und Zuweisung der qualitativen Eigenschaften auf Modellebene M1. (Quelle: Eigene Darstellung)

7.2 Entwurf der SI-Architekturmuster

Nach der *Identifikation der SI-Komponenten* im Prozessschritt I.A wurden diejenigen SI-Komponenten aus der Gesamtmenge der Infrastrukturkomponenten selektiert, deren Einsatz in zukünftigen Entwicklungsprojekten erwünscht ist.

7.2.1 SI-Architekturmuster

7.2.1.1 Überblick

Ein Architekturmuster ist eine Lösungsvorschrift für ein typisches, komplexes Entwurfsproblem, welches wiederholt auftritt und mit den Mitteln der Objektorientierung lösbar ist:[479] Im Bereich der Softwareentwicklung steht im Gegensatz zu Klassenbibliotheken und Frameworks nicht eine konkrete Implementierung im Vordergrund sondern vielmehr die Erfahrung, wie ein bestimmtes Entwurfsproblem gelöst wird. Muster sind eine Möglichkeit, Designerfahrungen so festzuhalten, dass sie auch von unerfahrenen Entwicklern wiederverwendet werden können. Mittlerweile sind Patternbeschreibungen verfügbar, die den Softwaredesigner auf unterschiedlichen Abstraktionsebenen unterstützen. So existieren Architekturmuster, die beim Entwurf einer globalen Softwarearchitektur helfen. Oder es existieren sog. *Idiome*, die programmiersprachenabhängige Details beschreiben.[480] Ein Muster wird im Allgemeinen durch ein festgelegtes Beschreibungsschema beschrieben, welches bei den einzelnen Autoren leicht variiert, jedoch immer folgende Elemente enthält:[481]

- *Mustername*: Der Name des Musters versucht die Kernpunkte des Problems in wenigen Worten zu übermitteln.
- *Problembeschreibung*: Die Problembeschreibung klärt die Frage, in welchem Kontext das Muster eingesetzt werden kann. Das kann entweder durch eine ausführliche Problembeschreibung oder durch Auflisten von Vorbedingungen geschehen.

[479] Vgl. Gamma (1991)
[480] Vgl. Abschnitt 5.3.3.2.3
[481] Vgl. Gamma (1991)

- *Problemlösung*: Die Problemlösung beschreibt die einzelnen Lösungselemente, ihre Zuständigkeiten und ihr Zusammenspiel.
- *Konsequenzen*: Der Konsequenzenabschnitt beschreibt die Folgen, die mit dem Einsatz eines Musters verbunden sind und weist auf mögliche Nachteile hin. Wegen ihrer strukturierten Beschreibung sind Muster besonders gut in Katalogen organisierbar.

In Anlehnung an die Konzepte zur Komposition von Infrastrukturkomponenten zu Basisplattformen und Technologie-Sets von DERN und PENZEL,[482] enthält der SI-Katalog sog. *SI-Architekturmuster*, die wie folgt definiert werden:

Definition: *SI-Architekturmuster*

Ein *SI-Architekturmuster* ist ein Muster, das den Kriterien[483] nach LEA genügt und eine Lösung für ein wiederkehrendes Problem im Umfeld des Infrastrukturarchitekturentwurfs darstellt. Ein SI-Architekturmuster ist eine Komposition aus SI-Komponenten und anderen SI-Architekturmustern.

Im Folgenden werden die Begriffe SI-Architekturmuster und Architekturmuster synonym verwendet.

[482] Siehe Abschnitt 5.4
[483] Siehe Abschnitt 5.4

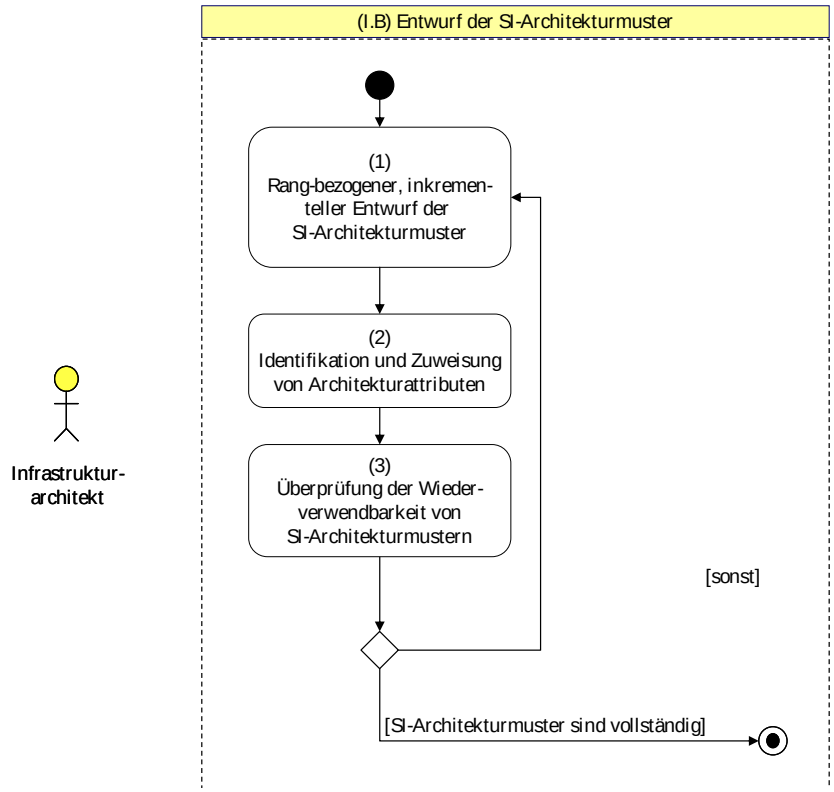


Abbildung 7.11: Der Subprozess Entwurf der Architekturmuster. (Quelle: Eigene Darstellung)

7.2.1.2 Architekturmuster und Designplattformen

Die im Rahmen des SI-Katalogs beschriebenen Architekturmuster müssen von Plattformen unterschieden werden, die im Rahmen der Vorstellung verschiedener Methoden des Umgangs mit Komplexität diskutiert wurden:[484] Die dem von HOFER beschriebenen Plattformkonzept zugrunde liegende Idee ist die Trennung einer Produktstruktur in Plattform- und Nicht-Plattform-Elemente („Rest“). Plattformen sind nach MAYER und LEHNERD Sätze von Subsystemen und Schnittstellen, die die Basis für abgeleitete Produkte bilden.[485] Nach HOFER umfassen Plattformen Komponenten, Funktionen, Schnittstellen und Designregeln.[486] Das Konzept der Plattformbildung entkoppelt die Elemente einer Produktstruktur nach Varianz- und Stabilisierungspunkten. Die Plattform basiert hierbei auf einer Kombination von Modulen, die keine Variation aufweisen. Der von HOFER als *Rest* bezeichnete Differenzierungsanteil

[484] Vgl. Abschnitt 5.2.2; vgl. Hofer (2001), S. 34ff
[485] Vgl. Meyer/Lehnerd (1997), S. 7 und S. 39 (nach Hofer (2001), S. 34)
[486] Vgl. Hofer (2001), S. 34

einer Produktstruktur setzt sich hingegen aus einer variablen Kombination von Modulen zusammen, wodurch eine Variation gegeben ist.[487]

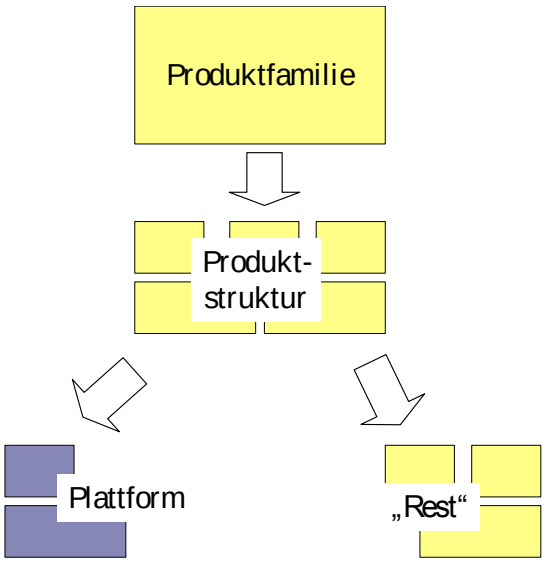


Abbildung 7.12: Mechanik des Plattformkonzepts (Quelle: In Anlehnung an Hofer (2001), S. 40).

Plattformen repräsentieren eine Hierarchiestufe einer sog. *Produktfamilie*, welche sich aus einer Menge individueller Produkte zusammensetzt, die auf einer gemeinsamen Basis – der Plattform – aufbaut (Abbildung 7.12).[488] Die Trennung zwischen Plattform- und Nicht-Plattform-Elementen unterliegt hierbei den Differenzierungsanforderungen und Vereinheitlichungspotentialen einer spezifischen Produktfamilie. Dies führt nach HOFER zu einer bewussten Reduktion der Produktvarianz, deren Folge eine Komplexitätsreduktion ist.

Die Wiederverwendung gleicher Komponenten über mehrere Produkttypen hinweg erfordert eine gezielte Vorausplanung. Ein Ziel dieser Vorausplanung ist es, bereits entworfene Komponentenstrukturen in neuen Entwicklungsvorhaben wiederverwenden zu können. Ein solches Vorgehen erfordert jedoch eine Vorhabenübergreifende Strategie, eine sog. *Plattformstrategie*, die die Ausrichtung von Produkten auf die Bedürfnisse verschiedener Kunden durch die Senkung der für die Weiterentwicklung notwendigen Grenzkosten, durch die Reduktion der Ent-

[487] Vgl. Hofer (2001), S. 54
[488] Vgl. Hofer (2001), S. 32ff

wicklungsdauer und durch die Reduktion der Produkt- und Prozesskomplexität vereinfacht.[489]

Produkt- <> Designplattform

Grundsätzlich kann zwischen zwei Plattformtypen unterschieden werden:[490] Eine *Produktplattform* bildet ein fixes, physisches Basisprodukt, das für alle Produkte einer Produktfamilie genutzt wird. Dem gegenüber steht eine *Designplattform*, die ein fixes Layout einer Produktfamilie vorgibt und die die Anordnung und den Typ von Elementen definiert. Die Elemente einer Produktstruktur, die das höchste Vereinheitlichungspotential besitzen, eignen sich zur Fixierung in einer Produktplattform. Die Produktplattform bildet hierbei eine physische Basis, die sich modular zusammensetzen kann, und auf die die restlichen, individuellen Elemente „aufgebaut" werden können. Ein derartiger Aufbau wird durch standardisierte Schnittstellen erreicht. Als Beispiele für Produktplattformen nennt HOFER die Plattformstrategie von Volkswagen[491] sowie die Swatch-Uhr. Wie die einzelnen Elemente und Elementtypen eines Produkts architektonisch angeordnet werden, wird mit Hilfe einer Designplattform beschrieben. Im Gegensatz zu einer Produktplattform, die eine physische und produzierbare Struktur besitzt, sind Designplattformen Baupläne. HOFER stellt fest, dass Designplattformen insbesondere dann geeignet sind, wenn komplexe Produkte beschrieben werden müssen, die höchst individuell spezifiziert werden und nicht ausschließlich aus standardisierten Bausteinen zusammengesetzt werden können.

Common IT Platform

Ein Beispiel für die Anwendung des Plattformkonzepts im Informationstechnischen Bereich ist die sog. *Common IT Platform*:[492] Auf Grund des zunehmenden Kostendrucks im Umfeld der Fluggesellschaften haben die Star-Allicance-Mitglieder Lufthansa (LH), United Airlines (UA) und Air Canada (AC) im Mai 2002 eine gemeinsame Initiative mit dem Namen Common IT Platform ins Leben gerufen. Ziel des auf ein Jahrzehnt angelegten Großprojekts ist es, die zentralen Kernfunktionen der Fluggesellschaften in den Bereichen „Reservierung", „Inventory- und Code-

[489] Vgl. Hofer (2001). S. 35
[490] Vgl. Hofer (2001), S. 58f
[491] Siehe Abschnitt 5.1.1
[492] Vgl. Ganswindt (2004)

share[493]-Management“, „Ticket-Management“ und „Check-in/Flugabfertigung“ künftig über eine gemeinsame Hard- und Softwareplattform zu betreiben.

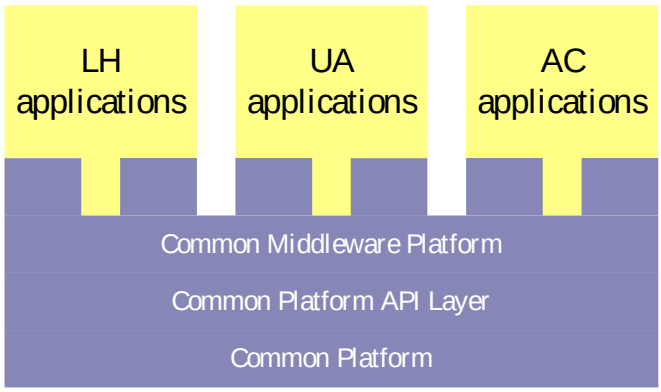


Abbildung 7.13: Architektur für die Common IT Platform (Quelle: In Anlehnung an Ganswindt (2004)).

Die Architektur dieser Plattform, die in Abbildung 7.13 skizziert ist, fußt auf einer fixen Komposition von Infrastrukturkomponenten, der sog. *Common Platform*. Die zentralen Kernfunktionalitäten, die bei jeder Fluggesellschaft identisch sind, werden in Form von Applikationen innerhalb des sog. *Common Platform API Layer* implementiert. Der Zugriff auf diese Geschäftlogik wird über einheitliche Middleware-Technologien angeboten, die zusammen die *Common Middleware Platform* bilden. Die funktionale Individualität, die die einzelnen Fluggesellschaften benötigen, wird außerhalb der Plattformen implementiert. Diese bilden den von HOFER als „Rest“ bezeichneten Teil einer Produktstruktur. In Anlehnung an GANSWINDT wurde eine Migration auf eine derartige Plattformarchitektur notwendig, da die zuvor betriebenen Applikationen der jeweiligen Fluggesellschaften auf Grund ihres Alters (zwischen 15 und 30 Jahre) unwirtschaftlich waren. Des Weiteren waren sie nicht in der Lage die Anforderungen an fachliche Funktionen der heutigen Zeit applikationstechnisch abzubilden.

Architekturmuster, Designplattformen

Die von HOFER als Designplattformen bezeichneten Baupläne für fixe Kompositionen von Systemelementen sind mit den im Rahmen des SI-Katalogs verwendeten Architekturmustern vergleichbar. Architekturmus-

[493] Anmerkung: Innerhalb eines Bündnisses bieten die Fluggesellschaften nicht selten einen Flug mit eigener Flugnummer (Code) an, den dann ein anderes Allianzmitglied ausführt.

ter sind ein Bauplan für ein wiederkehrendes Problem im Umfeld des Infrastrukturarchitekturentwurfs. Die von ihnen komponierten Bauteile sind die SI-Komponenten. Architekturmuster sind jedoch von Produktplattformen zu unterscheiden, da Produktplattformen physisch sind und sich aus einer fixen Anzahl und einer fixen Anordnung vorgegebener Bauteile zusammensetzen. Sofern wie im Fall der Common IT Platform eine strategische Vorgabe bzgl. der in zukünftigen Projekten einzusetzenden Infrastrukturkomponenten existiert, kann die Komposition dieser Komponenten mit Hilfe von Architekturmustern beschrieben werden.

7.2.1.3 Meta-Modell

In Prozessschritt I.A.2 wurden Obergruppen identifiziert, denen später eine hierarchische Abhängigkeit zugewiesen wurde. Der Grad der Abhängigkeit wurde als *Rang* bezeichnet. Je stärker eine Gruppe von anderen Gruppen dominiert wird, desto höher ist ihr Rang. Je niedriger der Rang einer Gruppe ist, desto seltener werden die Architekturmuster, die dieser Gruppe zugeordnet wurden, in anderen Gruppen wieder verwendet.

Der Prozessschritt B wird als inkrementeller Entwurf von Architekturmustern betrachtet, da eine Wiederverwendung von Rang-niedrigeren in Rang-höheren Architekturmustern der jeweiligen Gruppen stattfindet. Wie in Abbildung 7.14 dargestellt, ist ein Architekturmuster Teil einer Gruppe, wobei jeder Gruppe mindestens ein Architekturmuster zugeordnet sein muss. Jedes Architekturmuster kann in anderen Architekturmustern eine Verwendung finden, muss es aber nicht. Ein Architekturmuster, das keine anderen Architekturmuster verwendet, muss per Definition den geringsten Rang besitzen. Es verwendet ausschließlich SI-Komponenten. Durch die Wiederverwendung von Rang-höheren Architekturmustern in Rang-niederen Architekturmustern, werden SI-Komponenten implizit in Architekturmustern jeglichen Rangs verwendet.

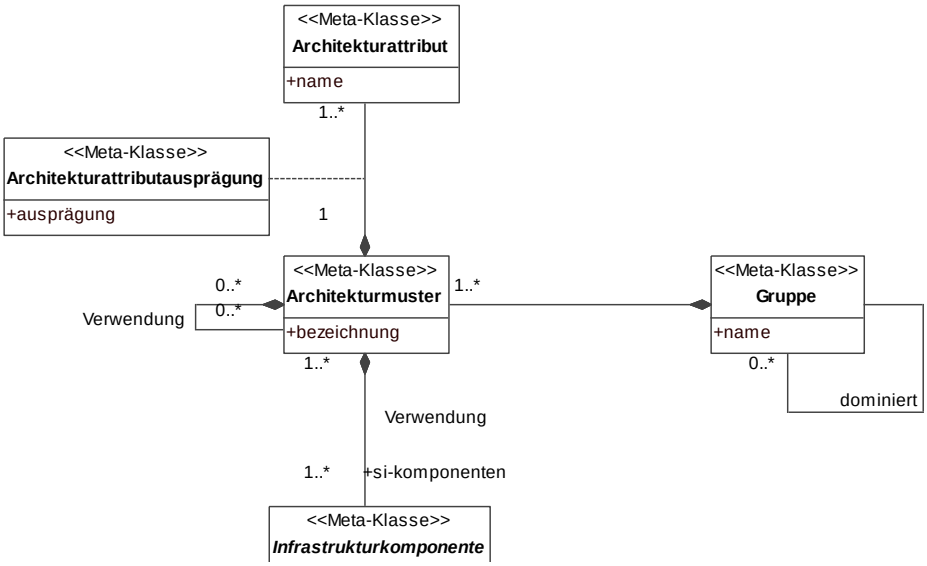


Abbildung 7.14: Ausschnitt des Meta-Modells für Prozessschritt I.B. (Quelle: Eigene Darstellung)

7.2.2 Rang-bezogener, inkrementeller Entwurf von SI-Architekturmustern

7.2.2.1 Motivation für einen Rang-bezogenen, inkrementellen Entwurf der SI-Architekturmuster

Die in Prozessschritt I.A.3 identifizierten Untergruppenübergreifenden Abhängigkeiten zeigen, inwieweit eine Infrastrukturkomponente eine Muss-, Kann- oder XOR-Abhängigkeit zu einer anderen Infrastrukturkomponente aufweist. Benötigt beispielsweise ein Datenbankmanagementsystem ein Betriebssystem, welches wiederum einen Server benötigt, so ergibt sich eine hierarchische Reihenfolge von einzelnen in den Architekturentwurf der Infrastruktur eingehenden Komponenten. Ohne den architektonischen Entwurf möglicher Server auf Basis der identifizierten SI-Komponenten kann der Entwurf der Betriebssysteme nicht erfolgen. Ohne den Entwurf der Betriebssystemarchitekturen macht der Entwurf des Datenbankmanagementsystems keinen Sinn. Ein Rang-bezogener, inkrementeller Entwurf von SI-Architekturmustern in dieser Phase kann lediglich auf Erfahrungswerten eines Infrastrukturarchitekten basieren.

Ergebnis des Prozessschritts I.A.3 (Abschnitt 7.1.3) sind die Gruppendominanzen und die daraus resultierenden Ränge der Gruppen. Je hö-

her der Rang einer Gruppe ist, desto höher ist der Grad der Wiederverwendung der dieser Gruppe zugeordneten SI-Komponenten. Die sich hieraus ergebenden Gruppendominanzen liefern einen Hinweis darauf, in welcher Reihenfolge die Architekturmuster der Obergruppen entwickelt werden müssen.

Die in Prozessschritt I.A.3 ermittelten Obergruppendominanzen sind: $(OG_1) < (OG_2) < (OG_4) < (OG_3)$. Sie deuten darauf hin, dass die SI-Komponenten der Obergruppen OG_1 bis OG_4 auf die der Obergruppe OG_2 angewiesen sind, bzw. dass die SI-Komponenten der Obergruppe OG_1 von denen der Obergruppen OG_3 und OG_4 abhängig sind.

Hierbei muss beachtet werden, dass die Gruppendominanzen der Obergruppen sich aus der Summe der Gruppendominanzen der einzelnen Untergruppen errechnen. Sind beispielsweise die SI-Komponenten der Untergruppe OG_xUG_a bzw. OG_xUG_b abhängig von den SI-Komponenten der Untergruppe OG_yUG_a und sind die SI-Komponenten der Untergruppe OG_yUG_b abhängig von den SI-Komponenten der Untergruppe OG_xUG_b abhängig, so ergeben sich folgende Untergruppendominanzen: $(OG_xUG_a = OG_yUG_b = 0) < (OG_xUG_b = 1) < (OG_yUG_a = 2)$. Hieraus resultiert die Obergruppendominanz $(OG_x = 1) < (OG_y = 2)$. Obwohl die Obergruppe OG_x die Obergruppe OG_y dominiert, sind die SI-Komponenten der Untergruppe OG_yUG_b auf die SI-Komponenten der Untergruppe OG_xUG_b angewiesen. Aus diesem Grund sind die in Prozessschritt I.A.3 ermittelten Gruppendominanzen lediglich als Orientierungshilfe geeignet, nicht jedoch als maßgebende Größe. Der Prozess des Entwurfs von Architekturmustern muss daher unter der Beachtung der Untergruppenabhängigkeiten inkrementell durchgeführt werden.

7.2.2.2 Die Verwendungsmatrix

Jede Obergruppe bietet die ihr zugeordneten SI-Komponenten ausschließlich in Form von Architekturmustern zur Verwendung an. Die direkte Verwendung einer SI-Komponente einer Obergruppe OGj in einer anderen Obergruppe OGj wird somit ausgeschlossen. In Anlehnung an das Meta-Modell (Abbildung 7.14) wird gewährleistet, dass das Angebot der SI-Komponenten durch qualitative Architekturattribute un-

terstützt wird, welche in Entscheidungsprozessen für oder gegen spezifische SI-Komponenten sprechen sollen.

		Verwendet ▸							
		OG_1				OG_2			
Architekturmuster ▾		P1	P2	UG_1 #2	UG_2 #3	P1	P2	UG_1 #4	… …
OG1	P1	Π	Π	Π	Π	Π	Π		
	P2	Π	Π	Π	Π	Π	Π		
OG_2	P1	Π	Π			Π	Π	Π	…
	P2	Π	Π			Π	Π	Π	…
…	…	…	…	…	…	…	…	…	…

Tabelle *7.7*: Verwendungsmatrix.

Als Strukturierungsinstrument für den inkrementellen Entwurf von Architekturmustern dient die in Tabelle 7.7 abgedruckte Verwendungsmatrix. Ihr Aufbau ermöglicht es, die Verwendung von Architekturmustern und SI-Komponenten in anderen Architekturmustern transparent zu machen. Das Füllen dieser Matrix basiert auf einem inkrementellen Analyseprozess und ist Aufgabe des Infrastrukturarchitekten. Der Analyseprozess dient der Schaffung eines Überblicks, welche Architekturkomponenten wie häufig eine Verwendungsbeziehung zu einem Architekturmuster aufweisen. Die Sensitivität der Verwendung kann unterschiedliche Formen annehmen. Der in der Verwendungsmatrix verwendete Platzhalter Π darf wie folgt ersetzt werden:

- Π = ⊙: Das Architekturmuster Px,i der Obergruppe OGi verwendet das Architekturmuster Py,j der Obergruppe OGj (mit $x \neq y$) bzw. die SI-Komponente Ii (mit Ii ∈ OGi). Diese Form der Verwendung wird im Folgenden als *Muss-Verwendung* bezeichnet.

- Π = [○: Dieses und die nachfolgenden Architekturmuster bzw. diese und die nachfolgenden SI-Komponenten in horizontaler Richtung (einschließlich des bzw. der mit Π = ○] gekennzeichneten Architekturmusters bzw. Infrastrukturkomponente) stellen eine Gruppe Γ'' von Architekturkomponenten dar, von denen eine von dem Architekturmuster verwendet wird. Die Verwendung wird im Folgenden als *XOR-Verwendung* bezeichnet.

- Π = ○]: Diese Architekturkomponente markiert das Ende der Gruppe Γ".
- Π = Leer: In diesem Fall besteht keine Verwendungsbeziehung zwischen einem Architekturmuster und einer Architekturkomponente.

Zum Verständnis der weiteren Ausführungen werden die Begriffe *Verwendung* und *Architekturkomponente* nun wie folgt definiert:

Begriffsdefinition: *Verwendung*

Eine Verwendung ist entweder eine Muss- oder eine XOR-Verwendung.

Begriffsdefinition: *Architekturkomponente*

Wird ein Architekturmuster bzw. eine SI-Komponente von einem Architekturmuster verwendet, so werden die verwendeten Elemente als Architekturkomponenten bezeichnet.

Architekturkomponenten können mehrfach in einem Architekturmuster verwendet werden. Wird eine Architekturkomponente (z. B. ein Architekturmuster der Obergruppe *Storage-Systeme*) in einem Architekturmuster mehrfach verwendet (z. B. weil das betrachtete Architekturmuster ein System beschreibt, das auf zwei Rechenzentrumsstandorte aufgeteilt ist und somit in jedem Rechenzentrum dieselben Storage-System Architekturen verwendet werden), so ist diese Mehrfachverwendung zu kennzeichnen. Die Kardinalität der Mehrfachverwendung wird mittels der Notationen {n} und {m..n} zugewiesen.

Das nachfolgend abgedruckte Beispiel soll den Gebrauch der Notation vorführen. Das Architekturmuster P2 kann entweder mit 2 SI-Komponenten der SI-Komponente I_a oder mit 3 SI-Komponenten der SI-Komponente I_b instanziiert werden. Das Architekturmuster P1 hingegen verwendet sowohl das Architekturmuster P2 (und somit implizit die

Komponenten I_a oder I_b) als auch die SI-Komponente I_c. Zur Instanziierung von P1 kann I_c beliebig oft verwendet werden:[494]

	P1	P2	#a	#b	#c
P1		⊙			⊙{1..n}
P2			[○{2}	○{3}]	

Optionale Verwendungen von SI-Komponenten werden in Architekturmustern ausgeschlossen, sofern die SI-Komponenten unterschiedlichen Leistungsklassen angehören. Der Grund hierfür ist die Identifikation spezifischer, qualitativer Eigenschaften der einzelnen Architekturmuster in Form von Architekturattributen. Wird eine Architekturkomponente einem Architekturmuster hinzugefügt, verändert diese das Gesamtverhalten und die Gesamtfunktionalität des architektonisch entworfenen Gesamtsystems. Daher müsste das optionale Hinzufügen und Weglassen von Architekturkomponenten mit Hilfe von Fallunterscheidungen abgebildet und auf die Architekturattribute projeziert werden. Da dies die Komplexität der Methodik erhöhen würde, werden optionale Verwendungen von SI-Komponenten unterschiedlicher Leistungsklassen per Definition ausgeschlossen. Sofern eine derartige optionale Verwendung in einem Architekturmuster abgebildet werden soll, muss ein separates Architekturmuster entworfen werden: eins mit der Verwendung und eins ohne eine solche Verwendung.

7.2.2.3 Identifikation von Architekturattributen und Regeln

Die Architektur eines beliebigen Systems besitzt spezifische qualitative Eigenschaften. Diese Eigenschaften ergeben sich aus der Komposition der einzelnen Systemelemente und fungieren als Selektionskriterium in Entscheidungsprozessen bezüglich der Infrastruktur.[495]
In Prozessschritt I.A.5 wurden einzelnen SI-Komponenten Leistungsattribute zugewiesen. Ein Leistungsattribut wurde als eine spezifische, qualitative Eigenschaft definiert, die allen Infrastrukturkomponenten einer Variantenmenge wertmäßig in Form einer Leistungsausprägung

[494] Anmerkung: Dies ist beispielsweise der Fall, wenn ein System auf Grund von variierenden Auslastungsgraden unterschiedlich skaliert werden kann.
[495] Vgl. Penzel (2004)

zugewiesen werden kann. Diese Leistungsattribute beziehen auf die qualitativen Eigenschaften einer einzelnen SI-Komponente. Die Komposition einzelner SI-Komponenten zu einem Architekturmuster liefert jedoch andere qualitative, kompositionsspezifische Eigenschaften. Eine solche Eigenschaft wird im Folgenden als *Architekturattribut* bezeichnet. Die wertmäßige Ausprägung eines solchen Architekturattributs wird als *Architekturattributausprägung* bezeichnet.

Beispielsweise können die Leistungsattribute „Performance“ und „Datendurchsatz“ bei einem Webserver qualitativ spezifisch ausgeprägt sein. Die qualitative Eigenschaft der sog. *Desaster Recovery* Fähigkeit kann jedoch nur einer Webserver-Architektur zugewiesen werden, deren Aufbau variieren kann. Jede Variation ergibt ein eigenes Architekturmuster. Einem Architekturmuster, das nur aus einem einzelnen Webserver besteht, kann eine spezifische Architekturattributausprägung ebenso zugewiesen werden wie einem Architekturmuster, bei dem einzelne Webserver mit einem Load Balancer zu einer Serverfarm komponiert werden. Die Ausprägung der Leistungsattribute der einzelnen SI-Komponenten bleibt in diesem Fall stets gleichwertig. Die Architekturattributausprägung variiert jedoch von Architekturmuster zu Architekturmuster.

Ziel des Prozessschritts I.B.2 ist daher eine qualitative Bewertung grundsätzlich vergleichbarer Architekturmuster mittels einer *Identifikation von Architekturattributen*. Als grundsätzlich vergleichbar sind die Architekturmuster einer Obergruppe anzunehmen. Allen Architekturmustern einer Obergruppe (z. B. „Webserver“) muss die gleiche Menge an Architekturattributen zugewiesen sein. Jedes Architekturmuster dieser Obergruppe prägt die Architekturattribute unterschiedlich aus.

Die Identifikation der Architekturattribute ist ein analytischer und subjektiver Prozess, der vom Infrastrukturarchitekten durchgeführt wird. Methoden, die eine Identifikation von Architekturattributen beschreiben, sind beispielsweise *The Open Group Architecture Framework* (TOGAF) oder das *Zachman Framework*.[496]

[496] Vgl. Zachman (1987) bzw. Harrison/Varveris (2004)

Vorausgesetzt, dass im Prozessschritt I.B.2 grundsätzlich vergleichbare Architekturmuster ein- und derselben Gruppe zugeordnet werden, können den Architekturmustervarianten auf Grund ihrer Vergleichbarkeit qualitative Eigenschaften zugewiesen werden. Qualitative Eigenschaften sind gruppenspezifische Architekturattribute, die bei jedem Architektur-muster, das dieser Gruppe zugeordnet wurde, individuell ausgeprägt sind.

Die Zuweisung gruppenspezifischer Architekturattribute und die Identifikation der Architekturmuster-individuellen Architekturattributausprägungen ist eine Entscheidungsunterstützung für nachfolgende Planungsprozesse. Im Gegensatz zur Spezifikation der Leistungsklassen in Abschnitt 7.1.5.1, bei der Leistungsattributausprägungen mit Hilfe einer Priorisierungsmatrix dargestellt wurden, wird im Fall der Identifikation der Architekturattribute eine Bool'sche Entscheidungsmatrix eingesetzt, die die Korrelation zwischen den Architekturmustern und den Architekturattributen darstellt. Grund hierfür ist die Beobachtung, dass Architekturattributausprägungen lediglich Ja/Nein-Werte sind. Beispielsweise ist eine Architektur skalierbar oder nicht bzw. sie ist Desaster Recovery fähig oder nicht.

Beispiel einer Entscheidungsmatrix

Ein Beispiel für eine derartige Entscheidungsmatrix soll die in Tabelle 7.8 abgedruckte Gegenüberstellung der Architekturmuster $P_{x,y}$ (y = 1, …, n) der Gruppe OG_x mit den Architekturattributen AA_x (x = 1, …, m) dienen. Die Korrelation zwischen P_y und AA_x besitzt die zulässige Wertemenge $w_{xy} = \{0, 1\}$.

Korrelliert ein Architekturattribut mit einem Architekturmuster nicht, signalisiert dies dem Infrastrukturarchitekten, dass die qualitativen Anforderungen an komponierte SI-Komponenten mit den vorhandenen Architekturmustern nicht abgedeckt werden können und dass ein entsprechendes Architekturmuster zu entwerfen ist.

Wie im Fall der SI-Komponenten ist auch jedem Architekturmuster mindestens eine *Regel* zugeordnet. Das Meta-Modell unterscheidet zwischen zwei Typen einer *Regel*: Die *Default Regel* beschreibt die Selektion eines Architekturmusters anhand der qualitativen Ausprägungen. Die *Strategische Regel* ist eine im Einzelfall speziell zu formulirende Regel, die den Einsatz des assoziierten Architekturmusters mit Bedin-

gungen verknüpft. Ein Beispiel für eine *Strategische Regel* ist die Bedingung: „Bei mehreren Kapazitätsanforderungen mit geringer Performanceanforderung kann dieses Architekturmuster zur Konsolidierung eingesetzt werden.“ Jedem Architekturmuster ist mindestens eine Regel zugeordnet. Wird einer Komponente keine *Strategische Regel* zugeordnet, so wird sie mit Hilfe der *Default Regel* qualitativ beurteilt.

		Architekturattribute ▸		
		$AA_{1,1}$	$AA_{1,2}$	$AA_{1,3}$
Gruppe ▾				
OG_1	$P_{1,1}$	●	●	●
	$P_{1,2}$		●	
	$P_{1,3}$			●

Tabelle *7.8*: Entscheidungsmatrix der Architekturmuster und der Architekturattribute.

7.2.2.4 Modellierung der SI-Architekturmuster

Die Modellierung der Architekturmuster ist die Konkretisierung der Meta-Ebene M2 auf der Modell-Ebene M1. In Anlehnung an den in Abbildung 7.14 abgedruckten Ausschnitt des Meta-Modells setzt sich ein Architekturmuster aus komponierten SI-Komponenten zusammen, die das Ergebnis des Prozessschritts I.A.7 sind.

Um ein konkretes Architekturmuster der Modell-Ebene M1 visuell kennzeichnen zu können, wird der Stereotyp <<*Design-Muster*>> eingeführt, der in diesem Prozessschritt allen Klasseninstanzen der Meta-Klasse *Architektur-muster* zugewiesen wird. Der Entwurf eines konkreten Architekturmusters erfolgt iterativ mit Hilfe der Rangfolgenidentifikation, der Verwendungs-matrix und der Abhängigkeitsmatrix. Die in den Architekturmustern grafisch dargestellten Informationen müssen mit denen der Verwendungs- und Abhängigkeitsmatrix konsistent sein:

- Existiert innerhalb der Abhängigkeitsmatrix eine Abhängigkeit zwischen der SI-Komponente I_a der Gruppe G_x und der SI-Komponente I_b der Gruppe G_y, so muss diese Abhängigkeit auch in mindestens einem Architekturmuster P der Gruppe G_x modelliert sein.
- Wird in einem Architekturmuster eine Abhängigkeit zwischen den SI-Komponenten I_a und I_b modelliert, so muss diese Abhängigkeit auch in der Abhängigkeitsmatrix existieren.

- Existiert in der Verwendungsmatrix eine Verwendung zwischen zwei Architekturmustern, so muss diese Verwendung auch im Modell erscheinen.
- Wird in einem Architekturmuster ein anderes Architekturmuster verwendet, so muss diese Verwendung auch in der Verwendungsmatrix vermerkt sein.

Etwaige, während der Modellierungsphase auftretende Inkonsistenzen müssen durch Rücksprünge zu dem entsprechenden Prozessschritt behoben werden.

Die Verwendung eines Architekturmusters P_i in einem Architekturmuster P_j wird in der Modell-Ebene M1 nicht direkt, sondern indirekt über einen mit dem Stereotyp <<*Port-Klasse*>> gekennzeichneten Klassifikator *K* modelliert. *K* muss Teil des Architekturmusters P_i sein und wird daher in Form einer Komposition mit P_i assoziiert. *K* muss mit mindestens einer Infrastrukturkomponente I_a, die ebenfalls Teil von P_i ist, assoziiert sein. Die Verwendung von P_i in P_j wird auf Seiten von P_j durch die Assoziation zwischen einer zum Architekturmuster P_j gehörigen Infrastrukturkomponente Ib und *K* dargestellt. Die Assoziation zwischen Ia und Ib sowie die Assoziation zwischen P_i und P_j über *K* müssen mit den Einträgen in der Abhängigkeitsmatrix bzw. Verwendungsmatrix konsistent sein.

Abstraktes Beispiel

Die Modellierung der Architekturmuster wird an dem im Abschnitt 7.1.3.1 eingeführten Beispiel fortgesetzt: In Anlehnung an die ermittelten Gruppendominanzen existiert zu den SI-Komponenten der Gruppen OG3 und OG4 die größte Abhängigkeit. Daher liegt es nahe, sinnvolle Kompositionen mit SI-Komponenten dieser Gruppen zuerst zusammenzustellen. Die in Abbildung 7.15 abgedruckte Klasse *OG_4P1* ist die Konkretisierung eines Architekturmusters der Meta-Ebene M2 auf der Modell-Ebene M1. OG4P1 repräsentiert das Architekturmuster P1 der Obergruppe *OG4* und wird daher mit dieser Gruppe assoziiert. Bestandteil dieses Architekturmusters ist die SI-Komponente I11 sowie die Port-Klasse *$OG_4P1Port1$*, über die andere Architekturmuster P1 wieder verwenden können. Im Fall dieses Architekturmusters wird entschieden, dass jede Wiederverwendung von I11 immer genau 3 Instanzen von

selbiger SI-Komponente erfordert. Daher wird die Kardinalität 3 an die Assoziation zwischen der Port-Klasse und der wieder zu verwendenden Design-Klasse geschrieben.

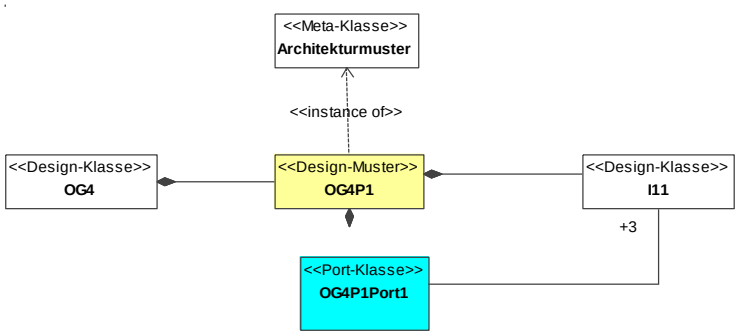


Abbildung 7.15: Architekturmuster P1 der Gruppe OG4. (Quelle: Eigene Darstellung)

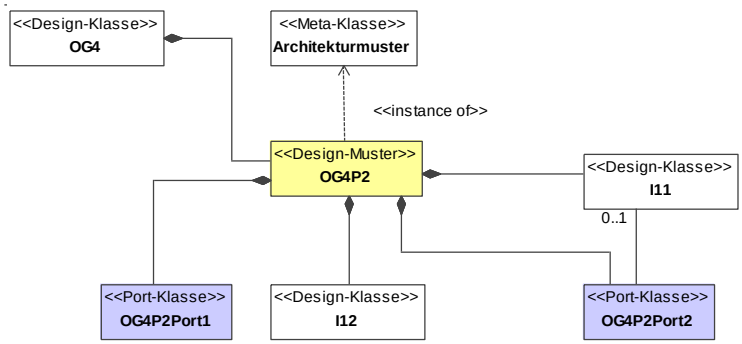


Abbildung 7.16: Architekturmuster P2 der Gruppe OG4. (Quelle: Eigene Darstellung)

Neben I11 gehört auch I12 zur Gruppe OG4. Das Architekturmuster P1 stellt jedoch lediglich die Verwendung von I11 zur Verfügung. Sofern der Infrastrukturarchitekt entscheidet, dass beim Einsatz einer Instanz von I12 optional auch ein Einsatz genau einer Instanz von II1 sinnvoll ist, kann P1 diesbezüglich nicht verwendet werden. Aus diesem Grund wird das Architekturmuster P2 benötigt, das eine derartige Komposition der SI-Komponenten I_{11} und I_{12} definiert. Wie in Abbildung 7.16 dargestellt ist, besteht keine direkte Beziehung zwischen I_{11} und I_{12}. Jede der beiden SI-Komponenten ist mit einer eigenen Port-Klasse assoziiert. Die Wiederverwendung von I_{11} bzw. I_{12} muss daher über die Port-Klasse $OG_4P2Port2$ bzw. $OG_4P2Port1$ erfolgen.

Die qualitative Beschreibung der Architekturmuster mit Hilfe von Architekturattributen kann sowohl tabellarisch als auch Modell-technisch erfolgen. Eine Modell-technische Beschreibung ist in Abbildung 7.17 am Beispiel der Gruppe OG_3 dargestellt.

Aus Tabelle 7.5 geht hervor, dass die SI-Komponente I7 eine Abhängigkeit zu SI-Komponenten der Gruppen OG3 und OG4 aufweist. Um eine Verwendung der zu OG3 und OG4 gehörenden SI-Komponenten mit I7 realisieren zu können, ist mindestens ein Architekturmuster für die Gruppe OG3 zu modellieren. Das in Abbildung 7.17 abgedruckte Architekturmuster P1 der Gruppe OG3 wurde analog zu den Architekturmustern der Gruppe OG4 entworfen.

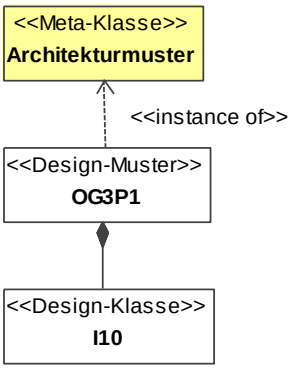


Abbildung 7.17: Architekturmuster P1 der Gruppe OG3. (Quelle: Eigene Darstellung)

Die Verwendung eines Architekturmusters in anderen Architekturmustern wird im Fall der Gruppe OG2 veranschaulicht (Abbildung 7.18): In Anlehnung an die Ergebnisse in Tabelle 7.5 ist eine Instanziierung der SI-Komponente I7 abhängig von Instanziierungen der SI-Komponenten der Gruppen OG3 und OG4. Hierzu werden in dem Modell des Architekturmusters P1 der Gruppe OG2 die entsprechenden Ports der wieder zu verwendenden Architekturmuster mit SI-Komponenten des zu modellierenden Architekturmusters assoziiert. Das Beispiel *OG_2P1* zeigt, dass die Assoziation zwischen I_7 und *$OG_4P1Port1$* bzw. zwischen I_7 und *$OG_3P1Port1$* drei Instanzen von I_{11} bzw. eine Instanz von I_{10} impliziert.

Die Modellierung der in diesem Abschnitt vorgestellten Architekturmuster repräsentiert einen Iterationsschritt. Das Ergebnis der Modellierung wird in der Verwendungsmatrix (Tabelle 7.9) festgehalten. Anlässe für weitere Iterationsschritte sind:

- Ein Architekturmuster der Gruppe OG_x benötigt SI-Komponenten der Gruppe OG_y, welche noch nicht über ein entsprechendes Architekturmuster angeboten werden.

> Ein Architekturmuster der Gruppe OG_x mit spezifischen qualitativen Eigenschaften wird benötigt. Die existierenden Architekturmuster der Gruppe OG_x weisen keine derartigen Attributausprägungen auf.

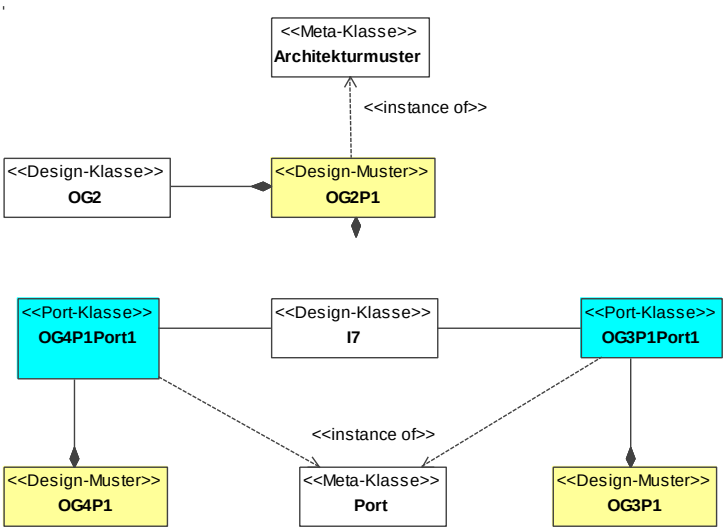


Abbildung 7.18: Architekturmuster P1 der Gruppe OG2. (Quelle: Eigene Darstellung)

		Verwendet ▸																
		OG1			OG2					OG3				OG4				
Architek-turmuster ▾		…	UG 1	UG 2	P_1	…	UG1		UG 2	P_1	…	UG 1	UG 2	P_1	P_2	…	#11	#12
			#2	#3			#4	#5	#7			#8	#10					
OG 1	…	…	…	…		…				…	…			…	…	…		
OG 2	P_1	…				…			⊙	⊙	…			⊙		…		
	…	…			…	…	…	…	…	…	…			…	…	…		
OG 3	P_1	…				…					…		⊙			…		
	…	…			…	…				…	…	…	…	…	…	…		
OG 4	P_1	…				…					…					…	⊙ {3}	
	P_2	…				…					…					…	⊙	⊙ {0..1}
	…	…			…	…				…	…			…	…	…	…	…

Tabelle *7.9*: Verwendungsmatrix nach der ersten Iteration.

8 Gestaltungskonzept – Anwendung der Methodik

Erst nachdem der SI-Katalog initial entworfen wurde, kann er innerhalb von Software Engineering Projekten eingesetzt werden. In diesem Fall wird das Ereignis E2 ausgelöst woraufhin die Phase *Anwendung des SI-Katalogs* in Kraft tritt (Abbildung 6.1). Diese Phase besteht lediglich aus einem einzigen Prozessschritt, der *Konfiguration von SI-Architekturmustern* (Abbildung 8.1), deren Subprozessschritte im Folgen erläutert werden.

Die Auswahlmöglichkeiten, die ein Anforderer innerhalb des hier beschriebenen Konfigurationsprozesses treffen kann, stellen den zum Zeitpunkt der Anwendung des SI-Katalogs jeweiligen Stand des Konfigurationswissens dar. In Anlehnung an BONGULIELMI basiert ein Konfigurationsprozesses auf folgenden Schritten: Der Selektion von einzelnen Komponenten, die Definition abstrakter und spezifischer Schnittstellen zwischen den Komponenten sowie das Testen der Abhängigkeiten und die Prüfung der Übereinstimmung zwischen den Kundenwünschen und der erzeugten Konfiguration.[497] Nach MÄNNISTÖ ET AL. werden in einem Konfigurationsprozess Kundenanforderungen einer technischen Spezifikation gegenübergestellt, welche die Bedürfnisse der jeweiligen Kunden in einer während dem Konfigurationsprozess verwendeten Sprache beschreibt.[498]

Die von BONGULIELMI beschriebenen Prozessschritte einer Konfiguration im Allgemeinen können nur bedingt bei der im weiteren Verlauf diskutierten Konfiguration von Architekturmustern eingesetzt werden: während die Selektion einzelner Komponenten (in Form der Selektion von Architekturmustern) sowie die Gegenüberstellung der spezifischen Anforderungen mit der technischen Spezifikation (in Form einer qualitativ gesteuerten Selektion) Elemente der Konfiguration innerhalb der Anwendung des SI-Katalogs sind, entfallen die Definition abstrakter Schnittstellen sowie das Testen der Abhängigkeiten. Der Grund für diesen Entfall ist, dass die zu einem Architekturmuster komponierten Ele-

[497] Vgl. Bongulielmi (2002), S. 42
[498] Vgl. Männistö et al. (1996)

mente bereits abstrakt definiert wurden und mit Hilfe abstrakter Schnittstellen verbunden sind.

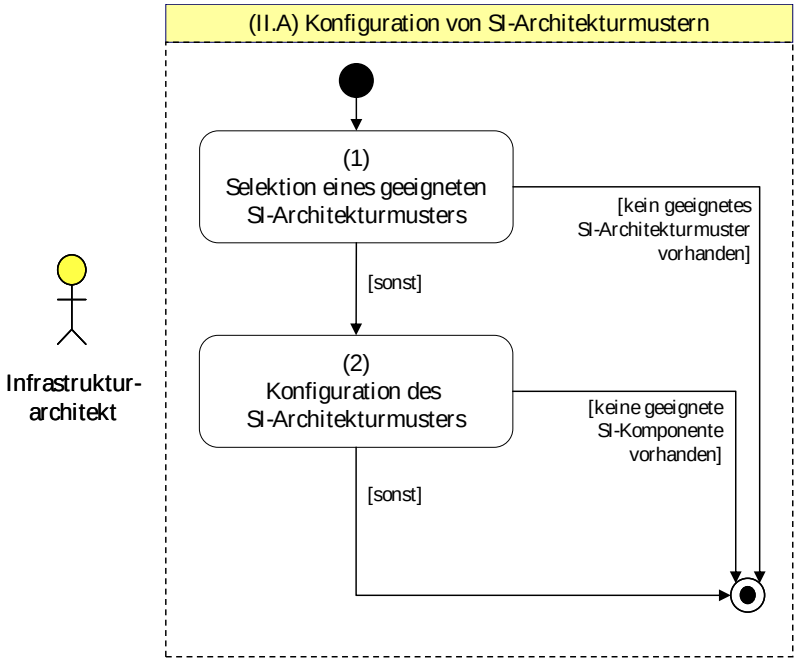


Abbildung 8.1: Der Prozessschritt II.A. (Quelle: Eigene Darstellung)

8.1 Selektion eines geeigneten SI-Architekturmusters

8.1.1 Muster als Tupel von Kontext, Problem und Lösung

Die Struktur des SI-Katalogs ist so angelegt, dass Obergruppen Architekturmuster enthalten, welche wiederum SI-Komponenten organisieren. Diese Strukturierung soll verhindern, dass SI-Komponenten direkt ausgewählt werden. Jede SI-Komponente ist Teil mindestens eines Architekturmusters. Jedes Architekturmuster besitzt wiederum spezifische Architekturattributausprägungen, die die Architekturmuster einer Obergruppe qualitativ unterscheidbar machen. Die Einhaltung dieser Vorgabe erübrigt die Vorgabe eines formalen Aufbaus des SI-Katalogs.

Muster = (Kontext, Problem, Lösung)

Die Voraussetzung der Selektierbarkeit eines geeigneten Architekturmusters ist die Existenz eines Problems, das mit Hilfe spezifischer Anforderungen an ein Infrastruktursystem bzw. an eine Infrastrukturarchitektur formuliert ist. Nach ALEXANDER ET AL. setzt der Mustereinsatz voraus, dass ein Muster als ein Tupel aus Kontext, Problem und Lösung

betrachtet wird:[499] Der *Kontext* beschreibt wiederkehrende Situationen, in denen das Muster angewendet werden kann. Das *Problem* entsteht durch Anforderungen mit Zielen und Bedingungen, die in diesem Kontext existieren. Die *Lösung* des Problems ist mit Hilfe einer kanonischen Entwurfsmethode bzw. einer Entwurfsregel beschrieben.

Diese Betrachtungsweise von Problem und Lösung ist vergleichbar mit derjenigen, die innerhalb des Generativen Domänenmodells beschrieben ist: In dem von CZARNECKI und EISENECKER definierten Paradigma zur Modellierung von Systemfamilien kann ausgehend von einer Anforderungsspezifikation mittels Konfigurationswissen aus elementaren und wiederverwendbaren Komponenten ein angepasstes Zwischen- oder Endprodukt nach Bedarf erzeugt werden.

Auch der von KERSTEN ET AL. vorgestellte Ansatz des Modular Engineering trennt inndirekt zwischen Problem und Lösung: Das Problem, dessen Lösung mit Hilfe des Modular Engineering gefunden werden soll, ist die Suche nach einer geeigneten Produktstruktur unter der Beachtung der vom Markt, den Abnehmern sowie dem produzierenden Unternehmen selber gestellten Ansprüche. Bei der Anwendung des Matrizenbasierten Verfahrens wird in Anlehnung an KERSTEN ET AL. die optimale Produktstruktur gefunden, die sowohl den Kundenanforderungen als auch der strategischen Ausrichtung des Unternehmens sowie den technischen Möglichkeiten und Grenzen gerecht wird. Dem Problem, das auf explizit formulierten Anforderungen an ein Produkt, der strategischen Unternehmensausrichtung sowie technischen Grenzen basiert, wird eine Lösung in Form einer Problemadäquaten Produktstruktur gegenübergestellt.

8.1.2 Aspekte einer Anforderung

Die Anwendung des SI-Katalogs beinhaltet eine spezifische Selektion von Architekturmustern. Die Anwendbarkeit des von ALEXANDER ET AL. definierten Mustereinsatzes setzt voraus, dass das 3-Tupel aus Kontext, Problem und Lösung an die Spezifika des SI-Katalogs angepasst werden muss (Abbildung 8.2).

[499] Vgl. Alexander et al. (1977)

Die von CZARNECKI und EISENECKER im Rahmen des Generativen Domänenmodells definierte Domäne bildet den Kontext des SI-Katalogs: die Infrastrukturarchitektur. Ein Problem, für das eine Lösung gefunden werden muss, wird in Form von expliziten Anforderungen formuliert. Diese Aufgabe wird von der Rolle des Anforderers abgedeckt. Das Ziel des Anforderers ist, dass der Infrastrukturarchitekt eine Problemadäquate Lösung in Form einer optimalen Infrastrukturarchitektur liefert, welche den Zielen und Bedingungen, die der Anforderer formuliert und die im Kontext der Infrastrukturarchitektur existieren, gerecht wird. Das Generative Domänenmodell sieht für die Problemformulierung eine Domänenspezifische Sprache (DSL) vor. Da die Anwendung des SI-Katalogs auf die Domäne der Infrastrukturarchitektur begrenzt ist, setzt der SI-Katalog voraus, dass ein vom jeweiligen Anforderer mit einer DSL formuliertes Problem folgende Aspekte einer Anforderung beinhaltet:

- Der erste Aspekt wird als *typisch* bezeichnet: Infrastrukturkomponenten werden in Ober- und Untergruppen eingeteilt. Eine Anforderung muss die gezielte Benennung einer solchen Obergruppe beinhalten. Die Gruppe repräsentiert den Typ. Beispielsweise kann eine allgemein formulierte Anforderung ein Betriebssystem fordern. Eine spezifischer formulierte Anforderung könnte ein Betriebssystem vom Typ Solaris fordern. Da die Strukturierungsebene von Architekturmustern Obergruppen sind, muss eine Anforderung im Kontext des SI-Katalogs zunächst allgemein formuliert sein. Das bedeutet, dass der Anforderer seinen Problemaspekt in diesem Beispiel in Form der Suche nach einem Betriebssystem und nicht gezielt nach einem Betriebssystem vom Typ Solaris formulieren muss.
- Der zweite Aspekt wird als *architektonisch* bezeichnet: Wie in Abschnitt 7.2.1 herausgestellt wurde, birgt die Form der Komposition von Infrastrukturkomponenten bestimmte, architektonisch-qualitative Eigenschaften in sich. Werden beispielsweise zwei Web Server mit Hilfe eines Load Balancers verbunden, und sind diese Web Server physisch unterschiedlich, so kann einer der beiden Web Server technisch ausfallen, und der verbleibende Web Server kann den Betrieb fortführen. Eine solche Architektur beschreibt die sog. *Verfügbarkeit*

dieser Web Server. Die im SI-Katalog gelisteten Architekturmuster werden mit Hilfe ihrer Architekturattribute qualitativ beschrieben. Diese Qualitätsattribute beschreiben den architektonischen Aspekt eines Problems, welchen der Anforderer zu formulieren hat.

> Der dritte Aspekt wird als *leistungstechnisch* bezeichnet: Die leistungstechnisch-qualitative Eigenschaft eines Systems kann sich sowohl als Gesamtleistung aller komponierten Infrastrukturarchitekturen als auch als Einzelleistung einer einzelnen Infrastrukturkomponente ergeben. So können beispielsweise zwei mit einem Load Balancer komponierten Web Server, die wiederum mit einem weniger leistungsfähigen Prozessor komponiert sind, mehr Clientanfragen bedienen als ein Web Server, der mit einem vergleichsweise hochleistungsfähigen Prozessor komponiert wurde. Der leistungstechnische Aspekt erfordert daher im Kontext des SI-Katalogs die Formulierung einer Einzelleistung einer an der Architektur beteiligten Infrastrukturkomponente seitens des Anforderers. Dies beinhaltet auch die gezielte Selektion einer Infrastrukturkomponente in Form von Typ und Menge auf Basis ihres spezifischen Einsatzgebietes. Beispielsweise dient eine Firewall der Unterbindung eines unerlaubten Datenaustauschs zwischen zwei Rechnern. Das Einsatzgebiet entscheidet über die Selektion einer Komponente und wurde im SI-Katalog abgebildet.

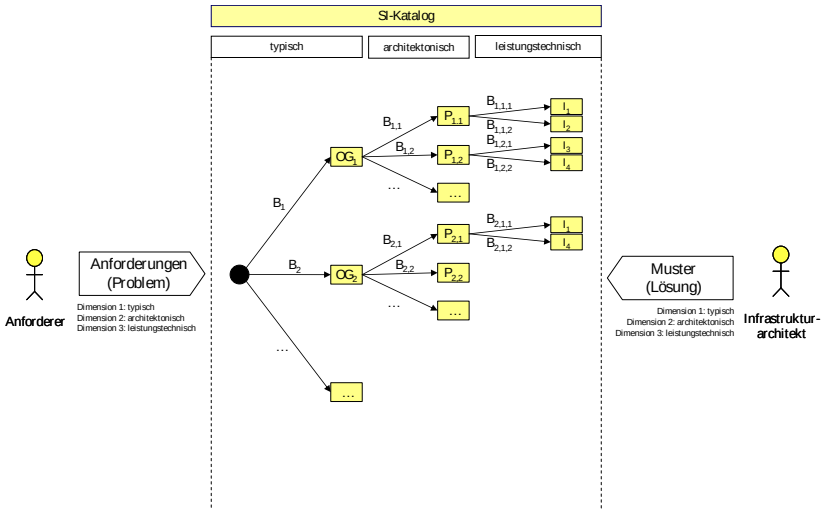


Abbildung 8.2: Der SI-Katalog zwischen Anforderung und Lösung. (Quelle: Eigene Darstellung)

8.1.3 Gezielte Mustersuche

Um dem von einem Anforderer mit den zuvor genannten Aspekten formulierten Problem eine adäquate Lösung in Form eines Musters gegenüberstellten zu können, muss das hierzu notwendige Konfigurationswissen, das im SI-Katalog enthalten ist, gemäß den drei zuvor genannten Aspekten strukturiert werden. Daher wurde während der Phase I eine ebensolche Strukturierung vorgenommen. Wie in Abbildung 8.2 dargestellt, repräsentieren die im SI-Katalog anhand der drei Aspekte typisch, architektonisch und leistungstechnisch strukturierten Architekturmuster eine Lösung für ein spezifisch formuliertes Problem. Entsprechend lässt sich der Inhalt des SI-Katalogs in Form eines Entscheidungsbaums strukturieren: Ausgehend vom Beginn entscheidet die Bedingung B_x *typisch*, in welcher Obergruppe nach einem geeigneten Architekturmuster gesucht werden soll. Wurde die Obergruppe OG_x selektiert, entscheidet der Anforderer unter der Beachtung architektonischer Aspekte welches der dieser Obergruppe zugeordneten Architekturmuster $P_{x,*}$ im Rahmen der Anforderung in Betracht kommt. Diese Entscheidung wird durch die Bedingung $B_{x,'}$ repräsentiert. Da jedes Architekturmuster eine Komposition einzelner Infrastrukturkomponenten (bzw. einzelner zu Architekturmustern komponierter Infrastrukturkomponenten) ist, entscheiden die Bedingungen $B_{x,y,*}$ welche Infrastrukturkomponente in welcher Anzahl im konkreten Fall zur Problemlösung beiträgt.

Die gewählte Strukturierung erlaubt es nun, die gezielte Mustersuche algorithmisch zu beschreiben. Aufgabe des nachfolgend abgedruckten Algorithmus F_1 ist die Selektion eines geeigneten Musters:

F_1(Anforderung):

- **Selektiere die Obergruppe OG unter Beachtung der Gruppendominanzen, die den Problemaspekt** *typisch* abdeckt.
- Falls keine geeignete Obergruppe vorhanden ist, breche ab.
- Selektiere unter Beachtung der zugeordneten Regeln das Architekturmuster P, das zu OG gehört und das den Problemaspekt *architektonisch* abdeckt.

> Falls kein geeignetes Architekturmuster vorhanden ist, breche ab.

> Führe aus: $F_2(P)$

Voraussetzung der Anwendbarkeit dieses Algorithmus ist die Existenz eines *typischen* Aspekts in der Problemformulierung des Anforderers, d. h. eine Anforderung muss die gezielte Benennung einer Obergruppe (z. B. „Datenbankserver“) beinhalten. Obergruppen besitzen Dominanzen. Die Selektion einer Obergruppe unter Beachtung der Gruppendominanzen verlangt daher die spezifische Selektion einer übergeordneten Obergruppe. Sofern das Architekturmuster P_1 einer Obergruppe OG_1 innerhalb des Architekturmusters P_2 einer Obergruppe OG_2 komponiert wird, so dass OG_2 im Kontext der Selektion dominanter ist als OG_1, wird hierdurch verhindert, dass Infrastrukturkomponenten einer weniger dominanten Gruppe vor denen der dominanteren Gruppe selektiert werden. Ein Beispiel für eine derartige Vorgehensweise ist die Selektion eines Datenbankmanagementsystems, dass im einfachsten Fall aus einem Datenbankserver besteht, an den ein Storage System angedockt ist. Die Obergruppe, der das Storage System angehört, ist im Kontext dieser Architektur weniger dominant als die Obergruppe, der der Datenbankserver angehört. Wird in diesem Fall das Storage System vor dem Datenbankserver selektiert, und weist das Storage System eine hohe Vielfalt der nicht mit dem gewünschten Datenbankserver kombinierbaren Varianten auf, besteht die Gefahr, dass der eigentlich fokussierte Datenbankserver nicht mit einem geeigneten Storage System kombiniert werden kann. Sofern sich innerhalb des SI-Katalogs keine Obergruppe findet, die dem typischen Aspekt der Problemstellung gerecht wird, muss die Suche abgebrochen werden. In diesem Fall muss ein in dieser Arbeit nicht diskutierter Prozess angestoßen werden, welcher die evtl. Beschaffung fehlender Infrastrukturkomponenten beschreibt.

Die zweite Voraussetzung der Anwendbarkeit dieses Algorithmus ist das Vorhandensein eines *architektonischen* Aspekts innerhalb der Problemformulierung. Jedes Architekturmuster besitzt qualitative Ei-

genschaften in Form der sog. *Architekturattribute*. Jedes Architekturmuster einer Obergruppe verfügt über die gleichen Architekturattribute (z. B. „Hochverfügbar“ oder „Desaster Recovery Fähig“). Der qualitative Unterschied zwischen den einzelnen Architekturmustern ergibt sich aus den jeweiligen Architekturausprägungen. Die Beachtung einer derartigen hierarchischen Suchreihenfolge lässt nicht zu, dass ein Anforderer nach einer speziellen Infrastrukturkomponente sucht. Existieren beispielsweise zwei Varianten eines Web Servers (z. B. ein Internet Information Server und ein Tomcat Web Server) kann keiner der beiden Server in diesem Prozessschritt direkt ausgewählt werden. Unterscheiden sich beide Server architektonisch, kann je nach Anforderung einer der beiden Servertypen sogar aus der Menge der selektierbaren Infrastrukturkomponenten herausfallen.

Die Selektion einzelner Infrastrukturkomponenten ist die Konfiguration eines Architekturmusters und wird im nächsten Prozessschritt thematisch behandelt.

8.2 Konfiguration des SI-Architekturmusters

Ein Architekturmuster ist eine Komposition von Infrastrukturkomponenten und anderen Architekturmustern. Sind zwei oder mehr Infrastrukturkomponenten Varianten eines Typs und können sie innerhalb ein- und desselben Architekturmusters wahlweise eingesetzt werden, so wird die Assoziation zwischen dem Architekturmuster und den Komponenten in der Regel als Spezialisierung modelliert. Die Relation zwischen den zum Architekturmuster komponierten Elementen und dem Architekturmuster selbst besitzt eine Kardinalität, die die Häufigkeit der instanziierbaren Elemente beschreibt.

Die Konfiguration des zuvor in der gezielten Mustersuche selektierten Architekturmusters hat die Aufgabe, die zu diesem Architekturmuster direkt oder indirekt[500] komponierten Infrastrukturkomponenten *leistungstechnisch* (d. h. auch mengenmäßig) gemäß der Anforderung

[500] Anmerkung: Eine indirekte Komposition entsteht, wenn ein Architekturmuster P_1 zu einem anderen Architekturmuster P_2 komponiert wird. Die zu P_1 *direkt* komponierten Infrastrukturkomponenten sind hierdurch *indirekt* zu P_2 komponiert.

auszuwählen. Die hierzu geeignete Schrittfolge ist in dem nachfolgend abgedruckten Algorithmus F_2 beschrieben:

F_2(Architekturmuster P):

Für jede SI-Komponente I, die zu P komponiert ist:

- Falls I nicht generalisiert ist: Überprüfe unter der Beachtung der zugeordneten Regeln, ob I die Leistungsanforderungen und das spezifische Einsatzgebiet abdeckt.
- Falls I qualitativ nicht geeignet ist: breche ab. Ansonsten merke dir I.
- Falls I generalisiert ist: Überprüfe unter der Beachtung der zugeordneten Regeln, welche Spezialisierung I' von I den Leistungsanforderungen und dem spezifischen Einsatzgebiet qualitativ entspricht.
- Falls I' qualitativ nicht geeignet ist: breche ab. Ansonsten merke dir I'.
- Prüfe ob die benötigte Komponentenanzahl von I bzw. I' konform mit im Architekturmuster definierten Kardinalität ist.
- Ist dies nicht der Fall: breche ab.
- Für jedes zu P komponierte Architekturmuster P': F_2(P')

Komponentenselektion

Der erste Entscheidungsweg betrifft die Frage, ob eine zum Architekturmuster *P* komponierte Infrastrukturkomponente *I* generalisiert ist oder nicht. Sofern sie es nicht ist, muss gemäß den Anforderungen an einen Konfigurationsprozess nach BONGULIELMI überprüft werden, ob zum einen das spezifische Einsatzgebiet von *I* den Anforderungen entspricht und zum anderen die Leistungsattributausprägungen von *I* die Anforderungen abdecken. Sind die Leistungsattributausprägungen von *I* qualitativ hochwertiger als die angeforderte Qualität, so ist *I* zur Abdeckung der Anforderungen dennoch qualitativ geeignet. Ist dies nicht der Fall, so muss die Konfiguration abgebrochen werden. Handelt es sich bei *I* um eine generalisierte Infrastrukturkomponente, muss jede Spezialisierung *I'* von *I* bzgl. der Abdeckung der Anforderungen überprüft

werden. Existiert beispielsweise eine Kompositionsbeziehung vom Typ 1..* zwischen P_x (z. B. „Skalierbarer Webserver“) und I_y (z. B. „Webserver“), und ist I_y eine Generalisierung der Infrastrukturkomponentenvarianten I_y' (z. B. „Internet Information Server“) und I_y'' (z. B. „Tomcat Webserver“), muss anhand des spezifischen Einsatzgebietes und anhand der Leistungsattributausprägungen überprüft werden, welche Spezialisierung von I_y innerhalb dieses Konfigurationsprozesses selektiert wird. Sofern die Leistungsattribute von I_y' und I_y'' identisch ausgeprägt sind entscheidet das spezifische Einsatzgebiet (z. B. „Für .NET-basierte Anwendungen“ bzw. „Für Java-basierte Anwendungen“) welche der beiden Infrastrukturkomponentenvarianten selektiert wird. Deckt keine Spezialisierung I’ die *leistungstechnischen* Anforderungen ab, ist *I* qualitativ ungeeignet, und die Konfiguration muss abgebrochen werden.

Wurde die Konfiguration auf Basis qualitativer Gründe nicht abgebrochen, muss in einem nächsten Schritt überprüft werden, ob die Anzahl der benötigten Komponenten von *I* (bzw. *I’*) auf Basis der Kardinalität zwischen *P* und *I* selektiert werden kann. Erfordert die Konfiguration eines Architekturmusters bspw. die Komposition eines Load Balancers und 5 Webservern, und ist zwischen dem Load Balancer und dem Webserver eine Kardinalität von 1..4 definiert, so eignet sich dieses Architekturmuster nicht um die Anforderungen mengentechnisch abzudecken. In diesem Fall muss die Konfiguration ebenfalls abgebrochen werden.

Wird die Konfiguration jedoch nicht abgebrochen, d. h. decken die zu *P* komponierten Infrastrukturkomponenten *I* qualitativ und mengentechnisch die Anforderungen ab, muss diese Konfiguration auch auf alle zu P komponierten Architekturmuster *P’* einer weniger dominanten Obergruppe angewendet werden. Ein Abbruch in der Konfiguration von *P’* hat einen Abbruch in der Konfiguration von *P* zur Folge.

Abbruch

Ein Abbruch innerhalb der Komponentenselektion gemäß Algorithmus F_2 kann folgende Gründe haben:

- Die qualitativen Eigenschaften einer zu *P* komponierten Infrastrukturkomponente sind qualitativ geringer ausgeprägt als gefordert bzw. das spezifische Einsatzgebiet einer zu *P* komponierten Infrastruktur-

komponente entspricht nicht dem geforderten Einsatzgebiet: In diesem Fall fehlt eine Infrastrukturkomponente im SI-Katalog. Es muss ein Prozess angestoßen werden, der die Beschaffung der fehlenden Infrastrukturkomponente mit anschließendem Einpflegen dieser Komponente in den SI-Katalog beschreibt. Alternativ kann zwischen dem Anforderer und dem Infrastrukturarchitekten ein Konsens erzielt werden, ob eine Änderung der Anforderungen legitim ist und die Selektion einer alternativen, im SI-Katalog gelisteten Infrastrukturkomponente in Frage kommt.

> Die zwischen *P* und *I* definierte Kardinalität entspricht nicht der geforderten Anzahl der Instanzen von I: Eine mögliche Ursache hierfür kann das fehlen eines Architekturmusters sein. Der Infrastrukturarchitekt muss daher überprüfen, ob evtl. ein neues Architekturmuster diese Anforderungen abdeckt. Ist dies der Fall, so muss ein ebensolches Muster in den SI-Katalog eingepflegt werden.

Ein Abbruch der Konfiguration kann entweder zur Folge haben, dass die Anforderungen endgültig nicht erfüllt werden können, oder der Abbruch leitet einen Lernprozess ein, der im nächsten Abschnitt diskutiert wird.

9 Gestaltungskonzept – Pflege der Methodik

Die dritte Phase, in die der SI-Katalog die innerhalb seines Lebenszyklus eintreten kann, ist die der Pflege der Methodik. Der Eintritt in diese Phase wird durch das in Abbildung 6.1 dargestellte Ereignis *E3* ausgelöst, das eine der folgenden konkreten Formen annehmen kann:

- Ein vorgegebener zeitlicher Zyklus ist beendet (Abschnitt 9.2.1).
- Neue Infrastrukturkomponenten bzw. Infrastrukturkomponentenvarianten wurden eingeführt (Abschnitt 9.2.2).
- Ein neues Architekturmuster muss eingepflegt werden (Abschnitt 9.2.3).

Die mit dem Eintritt eines dieser Ereignisse verbundenen Aktivitäten werden in den nachfolgenden Abschnitten thematisiert.

9.1 Der SI-Katalog als kognitives System

Die vom SI-Katalog erfassten Architekturmuster bilden das Domänenwissen eines Infrastrukturarchitekten symbolisch ab. NINCK und BÜSSER stellen fest, dass *Wissen* ein zentraler Erfolgsfaktor für die Lösung von komplexen Problemen ist:[501] Das auf eigenen Erfahrungen und eigenen Konstruktionen beruhende Wissen wird von Personen nicht passiv aufgenommen sondern aktiv konstruiert. Verfügt eine Person über Wissen, so ist diese Person nicht zwingend innovativ. Vielmehr muss ein Problemlösungsprozess die Innovation in Form von neuem Wissen auf Basis des vorherigen Wissensstandes schaffen.

Aus erkenntnistheoretischer Sicht existiert keine objektive Realität. Eine Person nimmt die Realität stets subjektiv war. Die übertragenen Informationen der Umgebung werden von einer Person in Form von Mustern empfangen. Diese Muster werden mit vorhandenen Denkmustern assoziiert und zu neuen Mustern zu einem sog. *Mustervorrat* hinzugefügt. Es entsteht ein Kreislauf, der als *Lernprozess* bezeichnet wird. Treten

[501] Vgl. Ninck/Büsser (2003)

zwischen den wahrgenommenen und den im Mustervorrat gespeicherten Strukturen Differenzen auf, führt dies zu einer Anpassung bzw. Neukonstruktion der mentalen Strukturen innerhalb des Mustervorrats.[502]

Betrachtet man die Methodik des SI-Katalogs als ein kognitives System, das das Wissen der Gruppe der Infrastrukturarchitekten verwaltet, bildet der Lebenszyklus des SI-Katalogs den Lernprozess innerhalb der betrachteten Domäne ab. Die initiale Erstellung des Mustervorrats erfolgt in der Phase I. In dieser Phase konstruiert der Infrastrukturarchitekt auf Basis seiner Erfahrungen einen subjektiven Mustervorrat. Dieser Mustervorrat wird innerhalb der Phase II zur Problemlösung eingesetzt. Hierbei versucht der Infrastrukturarchitekt ein vorhandenes Muster für die die vom Anforderer ausgehenden Informationen zu finden. Voraussetzung für die Anwendbarkeit des SI-Katalogs ist, dass diese Informationen gemäß den drei Aspekten *typisch*, *architektonisch* und *leistungstechnisch* strukturiert sind. Findet sich ein kein geeignetes Muster, das die Informationen der Anforderungen abdeckt, bricht der Selektionsprozess ab. Ein Selektionsabbruch kann entweder zur Folge haben, dass die Anforderungen endgültig nicht erfüllt werden können, oder der Abbruch leitet einen Lernprozess ein.

Dieser Lernprozess findet innerhalb der Phase III statt, in der ein neues Architekturmuster entweder direkt oder indirekt durch die Aufnahme einer neuen Infrastrukturkomponente angelegt wird. Die konkreten Auslöser dieses Phaseneintritts werden nachfolgend diskutiert.

9.2 Auslöser des Phaseneintritts

9.2.1 Zeitlicher Überarbeitungszyklus

Die Innovationszyklen im Bereich des Software Engineerings und insbesondere im Bereich der Infrastrukturkomponenten werden stetig kürzer.[503] Ohne eine zyklische Renovierung des vom SI-Katalog verwal-

[502] Vgl. Ninck et al. (2002)
[503] Vgl. Behrendt et al. (1998)

teten Inventars der Infrastrukturkomponenten besteht die Gefahr, dass in neuen Infrastrukturarchitekturentwürfen diejenigen Komponenten verwendet werden, deren Einsatz strategisch nicht mehr erwünscht ist. Im Prozessschritt I.A.1 wurden unterschiedliche Infrastrukturkomponenten identifiziert, die sich zum Zeitpunkt der Ist-Aufnahme im Betrieb befanden.

In Anlehnung an PENZEL sind viele dieser initial identifizierten Infrastrukturkomponenten jedoch aus strategischen Gründen nicht mehr für den Einsatz in einem Architekturentwurf geeignet. Daher wurde in Prozessschritt I.A.4 die Selektion zukunftsorientierter Infrastrukturkomponentenvarianten durchgeführt. Bei dieser Selektion wurden zum einen ein gezielter Ausschluss und zum anderen eine gezielte Bevorzugung einzelner Infrastrukturkomponenten praktiziert.

Um den Inhalt des SI-Katalogs stets aktuell zu halten, sollte sein Inhalt in zyklischen Abständen im Hinblick auf die Zukunftsorientierung einzelner Infrastrukturkomponenten überprüft werden. Hierbei müssen die strategischen Entscheidungen bzgl. der Infrastruktur, die während der initialen Erstellung bzw. der letzten Überarbeitung des SI-Katalogs getroffen wurden, beachtet werden. Diese Überarbeitung schließt den Prozessschritt I.A.4 ein, bei dem die Infrastrukturkomponenten selektiert werden, und sie analysiert, welche Auswirkung der etwaige Ausschluss einzelner Infrastrukturkomponentenvarianten auf die Architekturmuster hat. Denn, ist eine selektierte Infrastrukturkomponente zu einem Architekturmuster komponiert, muss dieses Architekturmuster überarbeitet (d. h. gepflegt) werden.

Beim Eintritt dieses Ereignisses werden die Prozessschritte III.A (Abbildung 9.1) und III.B (Abbildung 9.2) durchlaufen.

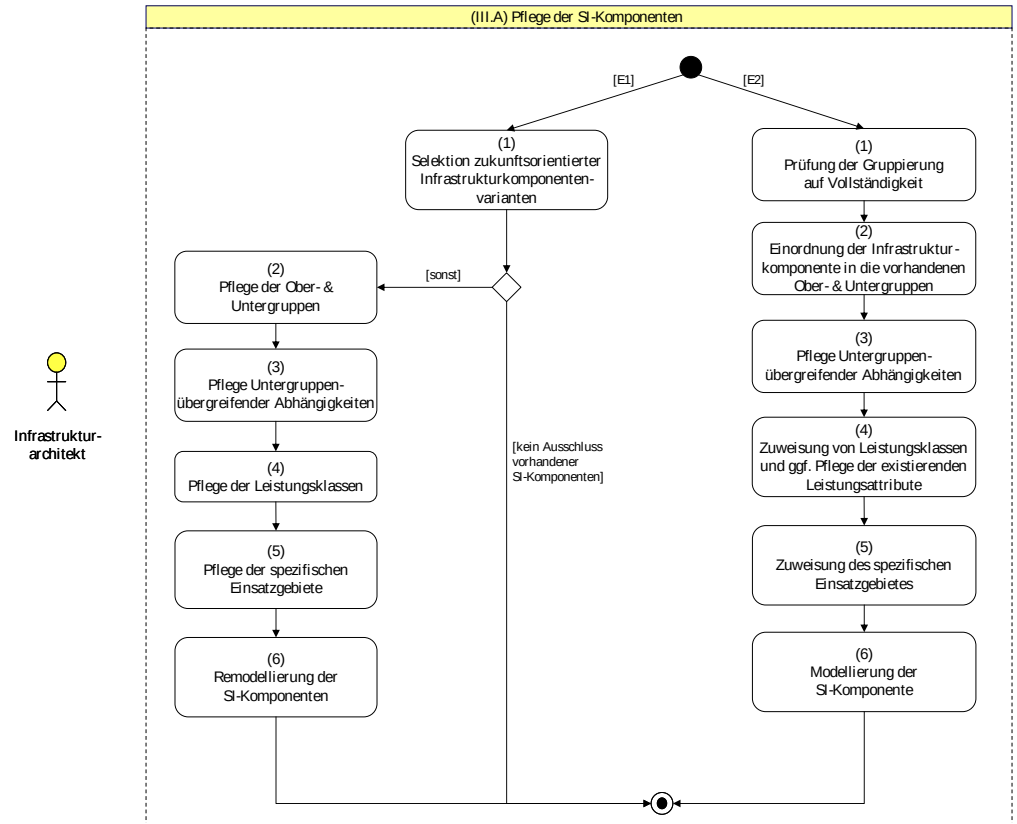


Abbildung 9.1: Der Subprozess Pflege der SI-Komponenten ist ereignisgesteuert. (Quelle: Eigene Darstellung)

9.2.2 Einführung neuer Infrastrukturkomponenten

Ebenso wie nicht mehr zukunftsorientierte Infrastrukturkomponenten innerhalb des Lebenszyklus des SI-Katalogs selektiert und ausgeschlossen werden, erfordert die Anwendbarkeit des SI-Katalogs auch die Einführung neuer Infrastrukturkomponenten. Die Einführung ebensolcher Komponenten erfordert einen akuten Bedarf, der sich aus konkreten, *typischen* Anforderungsaspekten ergibt. Der Bedarf kann zum einen auf das Fehlen eines als Obergruppe definierten Infrastrukturkomponententyps (z. B. „Webserver“) zurückgeführt werden, oder er kann sich aus der Forderung nach einer spezifischen Infrastrukturkomponentenvariante (z. B. „Microsoft IIS 5.0“) ergeben.

Generell sollte der Einführung neuer Infrastrukturkomponenten eine Untersuchung vorausgehen, die sowohl den akuten Bedarf sowie die Sinnhaftigkeit einer solchen Einführung analysiert hat. Der SI-Katalog ist in diesem Zusammenhang lediglich ein passives Strukturierungsinstrument, welches das Domänenwissen des Bereichs Infrastrukturarchi-

tektur als Mustervorrat verwaltet. Wird beispielsweise während der Durchführung der Phase II festgestellt, dass die spezifische Infrastrukturkomponentenvariante „Mircosoft IIS 5.0“ gefordert wird, welche jedoch nicht Teil des SI-Katalogs ist, muss die Gruppe der Infrastrukturarchitekten einen Prozess einleiten, der die Untersuchung der Sinnhaftigkeit einer etwaigen Beschaffung dieser spezifischen Infrastrukturkomponentenvariante beinhaltet. In Anlehnung an KELLER und WENDT ist der Architekt ein Mediator, der eine Kommunikation mit denjenigen Akteuren anstrebt, die die Rollen des Kunden, des Managements und des Entwicklers innehaben.[504] In Abschnitt 6.3.2 wurde diese Rolle im Kontext der Finanzdienstleistungsinformatik insofern konkretisiert, dass der Infrastrukturarchitekt ein Mediator zwischen Kunde, Betrieb und Management ist. Über die Sinnhaftigkeit der Einführung einer spezifischen Infrastrukturkomponentenvariante muss daher mit allen beteiligten Akteuren ein Konsens unter wirtschaftlichen und strategischen Gesichtspunkten herbeigeführt werden.

Sofern beschlossen wurde, dass eine neue Infrastrukturkomponente eingeführt wird, müssen die bestehenden Matrizen und Architekturbilder, die initial in der Phase I erstellt wurden und ggf. in der Phase III gepflegt wurden, überarbeitet werden. Dies schließt sowohl die Pflege der SI-Komponenten als auch die Pflege (d. h. Überarbeitung und ggf. Neuerstellung) der SI-Architekturmuster ein.

Auch beim Eintritt dieses Ereignisses werden die Prozessschritte III.A (Abbildung 9.1) und III.B (Abbildung 9.2) durchlaufen.

9.2.3 Einführung eines neuen Architekturmusters

Der Bedarf bzgl. der Einführung eines neuen Architekturmusters resultiert aus der Anwendung des SI-Katalogs. Eine in der Phase II betrachtete Anforderung muss die drei Aspekte typisch, architektonisch und leistungstechnisch behandeln. Sofern bei der Anwendung des SI-Katalogs im Prozessschritt II.A.2 kein Architekturmuster gefunden wurde, welches die *architektonischen* Aspekte der Anforderung abdeckt, ist

[504] Vgl. Keller/Wendt (2003); siehe Abschnitt 6.3.3

es die Aufgabe des Infrastrukturarchitekten einen Prozess zur Erstellung eines neuen Architekturmusters einzuleiten.

In Prozessschritt I.B.2 wurde ein Rang-bezogener, inkrementeller Entwurf von SI-Architekturmustern durchgeführt, der unter Beachtung der identifizierten Abhängigkeiten zwischen den einzelnen Infrastrukturkomponenten sowie unter Berücksichtigung der Gruppendominanzen Architekturmuster auf Basis von Erfahrungswerten eines Infrastrukturarchitekten hervorbrachte. Wird bei der Anwendung des SI-Katalogs in Phase II festgestellt, dass der architektonische Aspekt einer Anforderung ein Architekturmuster erfordert, welches nicht im SI-Katalog aufgelistet ist, muss der Infrastrukturarchitekt ein neues Architekturmuster auf Basis der vorliegenden konkreten Anforderungen entwerfen.

Im Unterschied zum initialen Entwurf der Architekturmuster in Prozessschritt I.B.2, bei dem lediglich Erfahrungswerte zum Musterentwurf vorhanden sind, kann der Infrastrukturarchitekt in Prozessschritt III.B auf konkrete Anforderungen zurückgreifen. Auch in diesem Fall muss er als Mediator zwischen Kunde, dem Management und dem Betrieb auftreten und den Musterentwurf unter wirtschaftlichen Gesichtspunkten praktizieren. Der Prozessschritt III.B entspricht daher dem Prozessschritt II.B und unterscheidet sich lediglich darin, dass konkrete Anforderungen statt Erfahrungswerte den Musterentwurf leiten.

Beim Eintritt dieses Ereignisses wird der Prozessschritt III.B (Abbildung 9.2) durchlaufen.

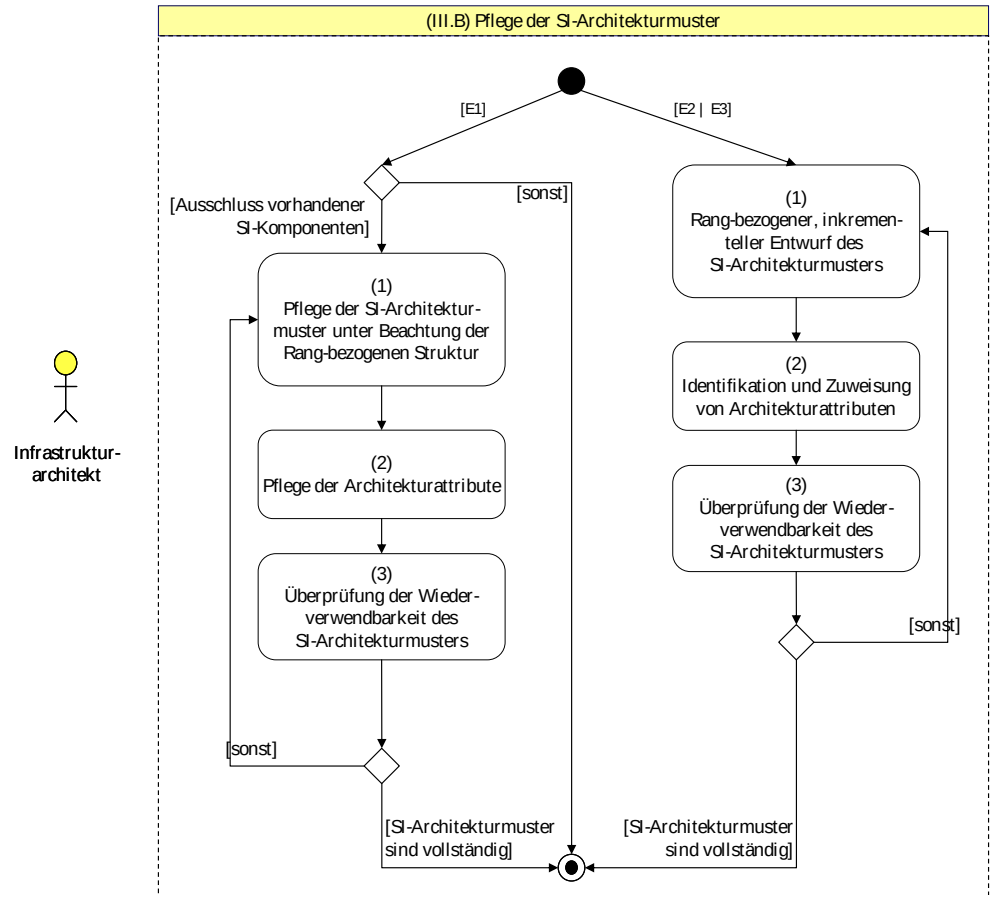


Abbildung 9.2: Der Subprozess Pflege der SI-Architekturmuster ist ereignisgesteuert. (Quelle: Eigene Darstellung)

10 Einführung, Anwendung & Pflege der Methodik am praktischen Beispiel

Ziel dieses Kapitels ist die praktische Einführung, Anwendung und Pflege der in den Kapiteln 6 bis 9 theoretisch definierten und als SI-Katalog bezeichneten Methodik. Die Umsetzung der Methodik wird im Folgenden bewertungsneutral beschrieben.

10.1 Allgemeines

10.1.1 Anmerkungen zur Umsetzung der Methodik

Die Umsetzung des abstrakt definierten SI-Katalogkonzepts erfolgte bei der Postbank Systems AG im Rahmen des Aktionsforschungsprojektes. Die Umsetzung selber ist Teil des Aktionsforschungsprojektes und repräsentiert den letzten Iterationsschritt dieses Projektes. Die Umsetzung der Methodik hat mindestens[505] 9 Monate in Anspruch genommen. Die Phasen I und II wurden im Dezember 2005 abgeschlossen. Die im Folgenden präsentierten Ergebnisse stellen keine vollständige Erfassung der bei der PBS existierenden Infrastrukturkomponenten bzw. der bei der PBS einsetzbaren Architekturmuster dar. Vielmehr handelt es sich um Teilergebnisse, da die Vielzahl und Varianz der tatsächlich vorhandenen Infrastrukturkomponenten sowie die tatsächlich benötigten Architekturmuster weitaus größer sind als diejenigen, die im Rahmen des Aktionsforschungsprojektes erfasst bzw. erstellt wurden.

10.1.2 Tools

Zur praktischen Umsetzung der Martizen-basierten Darstellungsform des SI-Katalos wurde Microsoft Excel eingesetzt. Zur UML-basierten Darstellungsform wurde anfangs mit dem Tool GoAgile von GoAgile

[505] Der Terminus „mindestens“ wird an dieser Stelle gewählt, da Einzelergebnisse, die bereits in früheren Iterationsschritten zur Erfassung und Analyse der prasisrealten Zusammenhänge erstellt wurden, nicht erneuert werden mussten, da sie qualitativ ausreichend für die Umsetzung der Methodik waren.

gearbeitet. Auf Grund vieler Unzulänglichkeiten dieses Produktes wurde letztendlich MagicDraw UML von No Magic verwendet.

10.1.3 Rollen

Der Lebenszyklus des SI-Katalogs wird von drei Phasen geprägt. Der Eintritt in eine jeweilige Phase ist als als Reaktion auf ein ausgelöstes Ereignis definiert. Die Umsetzung des SI-Katalogs in den einzelnen Phasen wird im Folgenden dokumentiert.
Bevor jedoch die eigentliche Umsetzung des SI-Katalogs erfolgen kann, muss das im SI-Katalog abstrakt definierte Rollenkonzept auf den Fall der PBS übertragen werden:

- Die Rolle *Kunde* wird bei der PBS von den Fachbereichen der PB wahrgenommen.
- Das *Management* setzt sich im Fall der PBS sowohl aus den einzelnen Ressortvorständen als auch aus dem für strategische Richtungsentscheidungen zuständigen Chief Information Officer (CIO) zusammen. Letzterer ist Vorstandsmitglied der PB.
- Die Rolle *Infrastrukturarchitekt* wird von den Mitarbeitern der Domäne Infrastruktur wahrgenommen.
- Die Domänenarchitekten nehmen die Rolle *Applikationsarchitekt* wahr.
- Die Rolle *Entwickler* wird bei der PBS nur selten von eigenen Mitarbeitern ausgefüllt. Oftmals wird diese Rolle von Subunternehmen wahrgenommen, die sich auf die Softwareentwicklung spezialisiert haben.
- Der *Betreiber* ist das jeweilige Pendant des Infrastrukturarchitekten in Form des Plattformverantwortlichen innerhalb des Ressorts Betrieb.

Neben den im SI-Katalog definierten Rollen existieren bei der PBS noch weitere Rollen: Diese Rollen werden u. a. deshalb benötigt, da jede als Projekt identifizierte Maßnahme durch einen *Projektleiter* gesteuert werden muss. Ein Projektleiter ist dafür zuständig, dass alle Projektbeteiligten (incl. der Architekten) Ergebnisse innerhalb eines vordefinierten

zeitlichen Rahmens und in einer vordefinierten Qualität erzielen. Neben der Rolle des Projektleiters existiert u. a. noch die Rolle *Business Analyst*. Dieser bildet die Schnittstelle zwischen *Kunde* und *Architekt*, indem er die Anforderungen des Kunden erfasst, hinterfragt und für die Projektbeteiligten aufbereitet.

10.2 Entwurf des SI-Katalogs

10.2.1 Identifikation der SI-Komponenten

10.2.1.1 Identifikation der Infrastrukturkomponenten des Infrastrukturbereichs

Im Rahmen einer Identifikation der zum Zeitpunkt der Untersuchung im Unternehmen eingesetzten Infrastrukturkomponenten wurde der Aufwand für die durchzuführende Ist-Aufnahme der betriebenen Hard- und Softwarekomponenen auf 3 Monate begrenzt. In Zusammenarbeit mit den Plattformverantwortlichen des Ressorts Betrieb sowie den entsprechenden Infrastrukturarchitekten wurden die bestehenden Komponenten identifiziert und in Tabelle 10.1 festgehalten.

Nach zweieinhalb Monaten erfasste die Ist-Aufnahme insgesamt 278 verschiedene Infrastrukturkomponenten. Ein nach Beendigung der Ist-Aufnahme durchgeführte Analyse ergab, dass die Liste der erfassten Komponenten zwar nicht vollständig ist, sie aber eine durchaus ausreichende Basis für den weiteren Untersuchungsverlauf darstellt.

Wie die auszugsweise in Tabelle 10.1 abgedruckten Ergebnisse zeigen, sind gleichzeitig mehrere Varianten eines Infrastrukturkomponententyps im Einsatz. Im Fall der Betriebssysteme hat sich beispielsweise als Grund herausgestellt, dass Softwarehersteller versuchen ihre Produkte mit der zum Zeitpunkt der Veröffentlichung aktuellsten Betriebssystemversion auszuliefern: Liefert ein Hersteller H_1 beispielsweise sein nur auf AIX 4 lauffähiges Softwareprodukt P_1 zum Zeitpunkt t_1 aus, so muss der Betrieb den entsprechenden Server über den Zeitpunkt t_1 hinaus betreiben. Veröffentlicht der Hersteller des Betriebssystems eine neue Version desselben, und kauft das Unternehmen zum Zeitpunkt t_2 das

Softwareprodukt P_2 des Herstellers H_2 (welches nur auf AIX 5.1 lauffähig ist) so muss der Betrieb einen weiteren Server mit diesem Betriebssystem betreiben. Somit werden in diesem Beispiel zum Zeitpunkt t_2 zwei unterschiedliche Varianten ein- und derselben Infrastrukturkomponente betrieben. In diesem Fall existieren vier Produkte von drei verschiedenen Herstellern, die innerhalb eines Unternehmens betrieben werden müssen. Nach der Veröffentlichung einer neuen Betriebssystemversion ist eine Abwärtskompatibilität nicht gewährleistet.

Eine weitere Beobachtung ist, dass im Fall von Hardwareprodukten Varianz bewusst in Kauf genommen wird. Im Fall der Storage Systeme können sich beispielsweise zwei Produkte qualitativ und funktionstechnisch gleichen. Um jedoch nicht in Abhängigkeit eines Herstellers zu geraten und um bei den Preisverhandlungen für neu einzukaufende Komponenten der einzelnen Hersteller eine bessere Verhandlungsposition einnehmen zu können, werden Produktvarianten eingekauft und später betrieben.

Folgendes Beispiel zeigt, wie Varianz von Softwareprodukten in einem solch großen Unternehmen entstehen kann: Eine Abteilung benötigt zur Bewertung von Kursdaten eine entsprechende Applikation. Auf dem Markt existieren mehrere Hersteller, die ein solches Produkt anbieten. Die Lösung L_1 des Herstellers H_1 ist in .NET programmiert und benötigt zur Laufzeit das Betriebssystem Microsoft Windows. Zur Zwischenspeicherung der Kursdaten wird eine Datenbank vom Typ Microsoft SQL Server benötigt. Die Lösung L_2 des Herstellers H_2 ist in Java programmiert und kann auf jedem beliebigen Betriebssystem und mit nahezu jeder verfügbaren relationalen Datenbank verwendet werden. Entscheidet sich die Abteilung zum Kauf von L_1 so werden zwangsweise das Betriebssystem Windows sowie ein Microsoft SQL Server in den Betrieb eingeführt, auch wenn dieser aus strategischen Gründen den Betrieb auf Oracle und DB2 Datenbanken beschränken möchte.

#ID	Hersteller	Produktname	Version	Typ[506]
1	BEA	WebLogic	8.1	SW
2	BEA	WebLogic	7.0	SW

[506] Anmerkung: HW = Hardwarekomponente, SW = systemnahe Softwarekomponente

3	JBoss	JBoss AS	3.2	SW
4	JBoss	JBoss AS	4.0	SW
…	…	…	…	…
43	EMC	Symmetrie		HW
44	EMC	DMX		HW
45	EMC	FC		HW
46	Hitachi	Thunder		HW
47	Hitachi	Lightning		HW
…	…	…	…	…
128	IBM	AIX	4	SW
129	IBM	AIX	5.1	SW
130	IBM	AIX	5.2	SW
131	IBM	AIX	5.3	SW
…	…	…	…	…

Tabelle *10.1*: Identifizierte Infrastrukturkomponenten im Rahmen des Prozessschritts I.A.1.

10.2.1.2 Einteilung der Infrastrukturkomponenten in Ober- & Untergruppen

Während der Ist-Aufnahme wurden die erfassten Infrastrukturkomponenten auf Basis ihrer Plattformzugehörigkeit bereits grob in Obergruppen eingeteilt. In einem nächsten Schritt wurde die grobe Strukturierung der Infrastrukturkomponenten durchgeführt. Hierzu wurde die Tabelle, die der Erfassung der einzelnen Infrastrukturkomponenten im vorherigen Prozessschritt diente, um zwei Spalten erweitert: *Obergruppe* und *Untergruppe*. Mit Hilfe eines Brainstormings wurden zunächst grob die Obergruppen festgelegt um ihnen nachfolgende Untergruppen zuweisen zu können. Anschließend wurde die erdachte Strukturierung überarbeitet. Die Vorgehensweise ist in Tabelle 10.2 angedeutet.

Nachdem die *Gruppenzugehörigkeit* über die beiden zusätzlichen Tabellen festgelegt wurde, konnte eine abschließende Sortierung das in Tabelle 10.3 dargestellte Ergebnis hervorbringen.[507] Da die Einteilung in Gruppen umfangreicher ist als die organisatorische Einteilung der PBS in Plattformen, wurden die an der Untersuchung teilnehmenden Architekten im Rahmen dieser Arbeit als Ansprechpartner und Verantwortliche für die identifizierten Gruppen themenbezogen aufgeteilt.

[507] Anmerkung: Die geklammerten Zahlen repräsentieren den Rang der jeweiligen Gruppe.

#ID	…	…	…	Gruppenzugehörigkeit	
				Obergruppe	Untergruppe
1	…	…	…	OG1	OG1/UG1
2	…	…	…	OG1	OG1/UG2
3	…	…	…	OG1	OG1/UG1
4	…	…	…	OG2	-
5	…	…	…	OG3/UG1	OG3/UG1/UG1
…				…	…

Tabelle *10.2*: Die Tabelle aus Prozessschritt A.1 wird im Prozessschritt A.2 um die Gruppenzugehörigkeit erweitert.

Gruppenzugehörigkeit		
J2EE Applikationsserver (1)	JBOSS	
	BEA	
	IBM	
	SAP (1)	
Webserver (2)	Java Webserver (2)	
	Microsoft Webserver (0)	
	Andere (0)	
Betriebssysteme (38)	Unix (27)	Linux (16)
		Solaris (5)
		AIX (6)
	Windows (11)	
	z/OS (4)	
	OS/390 (0)	
	BS2000 (0)	
Maschinen/Prozessoren (45)	SPARC (6)	
	PSeries (7)	
	Pentium-kompatibel (29)	
	Multiprise (1)	
	ZSeries (1)	
	BS2000-S1x0 (1)	
Datenbanken (9)	Oracle (3)	
	Microsoft (3)	
	IBM (3)	
Middleware (0)	Message Broker (0)	
	Application Broker (0)	
Monitoring Tools (0)	Applikationsebene (0)	
	Netzwerkebene (0)	
Security (0)	Firewall (0)	
	Intrusion Detection (0)	
Netzkomponenten (126)	Router (41)	
	Switch (41)	
	Proxy (3)	

	Load Balancer (41)
Storage Systeme (7)	
Hardware Management Tools (5)	
Virtualisierungstools (0)	
SAP R/3 (0)	

Tabelle *10.3*: Ermittelte Ober- & Untergruppen während der Identifikation der SI-Komponenten

10.2.1.3 Identifikation Untergruppenübergreifender Abhängigkeiten

Nachdem die Infrastrukturkomponenten in einer ersten Erhebung in Ober- und Untergruppen eingeteilt wurden, sieht das Vorgehensmodell im Prozessschritt I.A.3 die Identifikation Untergruppenübergreifender Abhängigkeiten vor. Zu diesem Zweck werden innerhalb des Prozessschritts die Infrastrukturkomponenten in einer Tabelle gegenübergestellt. Die Beziehung zwischen den Infrastrukturkomponenten wird mittels von Muss-/Kann- und XOR-Abhängigkeiten gekennzeichnet.

Angesichts der Tatsache, dass die Identifikation der Infrastrukturkomponenten insgesamt 278 Varianten hervorgebracht hat, erfordert die exakte Befolgung des Vorgehensmodells in diesem Fall die Handhabung einer 278x278 Zellen zählenden Matrix. Bereits der erste Durchführungsversuch zeigte, dass eine solche Matrix schier zu komplex für die adäquate Gegenüberstellung der einzelnen Infrastrukturkomponenten im Rahmen eines Workshops ist. Aus diesem Grund wurde die Komplexität der Abhängigkeitsmatrix in einem ersten Durchlauf dahingehend reduziert, dass die Abhängigkeiten zunächst auf Gruppenebene identifiziert wurden. Diese Vorgehensweise widerspricht nicht dem im Vorgehensmodell definierten Verfahren, da die Abhängigkeit einer Infrastrukturkomponente I_a einer Gruppe G_x von einer Infrastrukturkomponente I_b einer Gruppe G_y sowohl als Gruppendominanz $G_x < G_y$ als auch als Muss- bzw. Kann-Abhängigkeit $G_x \circ G_y$ bzw. $G_x \odot G_y$ definiert ist.

Anhand der in Tabelle 10.4 erfassten Abhängigkeiten wurden die in Tabelle 10.3 festgehaltenen Gruppendominanzen ermittelt. Im Anschluss an den Workshop wurden die Teilnehmer in Form einer Hausaufgabe gebeten, die Abhängigkeiten der in Ihr Aufgabengebiet fallenden Infrastrukturkomponenten von den Infrastrukturkomponenten der dominierten Untergruppen innerhalb der Abhängigkeitsmatrix zu detaillieren.

Nach der Fertigstellung der Teilmatrizen wurden die jeweils enthaltenen Ergebnisse zu einer Gesamtmatrix zusammengefasst. Zur Erstellung der Matrix wurde Microsoft Excel verwendet.

			J2EE Applikations-server				Webserver			Betriebssysteme							Maschinen / Prozessoren						Datenbanken			Middleware		Monitoring Tools		Security		Netzkomponenten				Storage Systeme	Hardware Management Tools	Virtualisierungs-tools	SAP R/3
										Unix																													
			JBOSS	BEA	IBM	SAP	Java Web-server	Microsoft Webserver	Andere	Linux	So-la-ris	AIX	Windows	z/OS	OS/390	BS2000	SPARC	pSeries	Pentium-kompatibel	Multiprise	zSeries	BS2000-S1x0	Oracle	Microsoft	IBM	Message Broker	Application Broker	Applikations-ebene	Netzwerk-ebene	Firewall	Intrusion Detection	Router	Switch	Proxy	Load Balan-cer				
J2EE Applika-tions-server	JBoss						○			[○	○]												[○	○	○]							○	○		○				
	BEA																																					
	IBM																																					
	SAP																																					
Web-server	Java Webserver									⊙																						○	○	○	○				
	Microsoft Webserver												⊙																			○	○	○	○				
	Andere									⊙													[○	○	○]							○	○	○	○				
Betriebssysteme	Unix	Linux																	⊙													○	○		○				
		Solaris															⊙															○	○		○				
		AIX																⊙														○	○		○				
	Windows																		⊙													○	○		○				
	z/OS																				⊙											○	○		○		⊙		
	OS/390																			⊙												○	○		○		⊙		
	BS2000																					⊙										○	○		○		⊙		
Maschinen / Prozesso-ren	SPARC																																					
	PSeries																																					
	Pentium-kompatibel																																					
	Multiprise																																					
	Zseries																																					
	BS2000-S1x0																																					

Tabelle *10.4*: Abhängigkeitsmatrix auf Gruppenebene.

		J2EE Applikations-server				Webserver			Betriebssysteme							Maschinen / Prozessoren						Datenbanken			Middleware		Monitoring Tools		Security		Netzkomponenten				Storage Systeme	Hardware Management Tools	Virtualisierungstools	SAP R/3
									Unix																													
		JBOSS	BEA	IBM	SAP	Java Webserver	Microsoft Webserver	Andere	Linux	Solaris	AIX	Windows	z/OS	OS/390	BS2000	SPARC	pSeries	Pentium-kompatibel	Multiprise	zSeries	BS2000-S1x0	Oracle	Microsoft	IBM	Message Broker	Application Broker	Applikations-ebene	Netzwerk-ebene	Firewall	Intrusion Detection	Router	Switch	Proxy	Load Balancer	Storage Systeme	Hardware Management Tools	Virtualisierungstools	SAP R/3
Datenbanken	Oracle								[○			○]																			○	○		○	⊙			
	Microsoft											⊙																			○	○		○	⊙			
	IBM												⊙																		○	○		○	⊙			
Middleware	Message Broker										⊙																				○	○		○				
	Application Broker								[○			○]																				○	○		○				
Monitoring Tools	Applikationsebene								[○	○	○	○]																										
	Netzwerkebene								[○	○	○	○]																										
Security	Firewall								[○	○	○	○]																										
	Intrusion Detection								[○	○	○	○]																										
Netzkomponenten	Router																																				
	Switch																																				
	Proxy																																				
	Load Balancer																																				
Storage Systeme																															○	○		○				
Hardware Management Tools																																						
Virtualisierungstools																																						
SAP R/3					○				[○		○]											[○	○	○]							○	○		○				

Tabelle *10.5*: Abhängigkeitsmatrix auf Gruppenebene (Fortsetzung).

10.2.1.4 Selektion zukunftsorientierter Infrastrukturkomponentenvarianten

Im Anschluss an Prozessschritt I.A.3 wurden diejenigen Infrastrukturkomponenten der Abhängigkeitsmatrix identifiziert, die strategisch in zukünftigen Architekturentwürfen verwendet werden sollen. Da viele der Verantwortlichen bereits im Vorfeld Überlegungen angestellt haben, welche Infrastrukturkomponenten veraltet sind und welche nicht, entfiel ein Großteil der Zeit auf die Abstimmung der Verantwortlichen untereinander, welche Auswirkungen die Erklärung der Nichtverwendbarkeit einer spezifischen Komponentenvariante auf von ihr abhängige Infrastrukturkomponenten anderer Gruppen hat.

Bei der Überführung der Abhängigkeitsmatrix in die Selektionsmatrix hat sich herausgestellt, dass die Darstellung der identifizierten Abhängigkeiten im Rahmen einer gezielten Selektion zukünftig verwendbarer Infrastrukturkomponenten die direkten Auswirkungen auf andere Komponentengruppen deutlich macht: Denn je größer der Rang einer Gruppe ist, desto häufiger sind andere Komponenten von den Komponenten dieser Gruppe abhängig. Wird eine Komponente einer ranghohen Gruppe als veraltet gekennzeichnet, desto größer sind die Auswirkungen auf andere Komponenten, sofern diese über eine Muss-Abhängigkeit (oder über eine XOR-Abhängigkeit aus der eine Muss-Abhängigkeit resultiert) mit dieser verbunden sind.

10.2.1.5 Spezifikation von Leistungsklassen je Komponentenvariante

Eine Leistungsklasse wurde als ein Instrument zur qualitativen Unterscheidung grundsätzlich vergleichbarer Infrastrukturkomponenten definiert. Als grundsätzlich vergleichbar werden die Infrastrukturkomponenten einer Gruppe angenommen.

Zur Einteilung der im letzten Prozessschritt selektierten SI-Komponenten in Leistungsklassen wurden in Einzelgesprächen zusammen mit den für die jeweiligen Gruppen zuständigen Architekten qualitative Unterscheidungsmerkmale der SI-Komponentenvarianten erarbeitet. Auf Grund der eingeschränkten Literaturbasis musste in je-

dem einzelnen Fall auf das Erfahrungswissen des jeweiligen Architekten sowie auf die von Herstellerseite vorgegebenen Leistungsindikatoren zurückgegriffen werden.

Die bspw. von der Gruppe *SAP* direkt oder indirekt abhängigen Gruppen sind (aufsteigend sortiert nach Rängen): *Netzkomponenten*, *Maschinen/*

Prozessoren, *Betriebssysteme*, *Storage Systeme*, *Datenbanken*, *Webserver* und *J2EE Applikationsserver*. Festzustellen ist, dass sich vergleichbare Infrastrukturkomponenten zweier Hersteller qualitativ nur marginal unterscheiden. Oftmals ist der Preis das entscheidende Kriterium. Da der Preis jedoch kein qualitatives Entscheidungskriterium ist, wurde er nicht mit in die nachfolgenden Untersuchungen aufgenommen. Wirtschaftliche Aspekte der Auswahl spezifischer Infrastrukturkomponenten werden bereits indirekt durch die strategische Selektion derselben behandelt. Insgesamt konnten folgende Leistungsattribute identifiziert werden:

- *Stabilität*: Die Stabilität einer Infrastrukturkomponente beschreibt die relative Ausfallwahrscheinlickeit derselben. Je unwahrscheinlicher ihr Ausfall ist, desto stabiler ist die Komponente.
- *Sicherheit*: Die Sicherheit einer Infrastrukturkomponente legt die relative Wahrscheinlichkeit fest, dass Unbefugte Zugriff auf die Funktionalitäten dieser Komponente erlangen. Je unwahrscheinlicher ein unerlaubter Zugriff ist, desto sicherer ist die Komponente.
- *Performance*: Die Performance einer Infrastrukturkomponente beschreibt die relative Geschwindigkeit mit der sie ihre Funktionen im Vergleich zu Varianten dieser Komponente anbietet. Je höher diese relative Geschwindigkeit ist, desto performanter ist die Komponente.
- *Kapazität*: Die Kapazität beschreibt die relative Fähigkeit einer einzelnen Komponente, Daten verwalten zu können. Je mehr Daten im Verhältnis zu Komponentenvarianten verwaltet werden können, desto höher ist die Kapazität einer Infrastrukturkomponente.
- *Änderbarkeit*: Die Änderbarkeit beschreibt Möglichkeit, die Infrastrukturkomponente durch eine neuere Version derselben Komponente zu ersetzen, ohne dass die bestehende Komponente ihre laufende Tä-

tigkeit einstellt. Je geringer die Wahrscheinlichkeit ist, dass die Komponente für ein sog. *Update* ihre laufende Arbeit einstellt, desto höher ist der Grad ihrer relativen Änderbarkeit.

In Bezug auf die betrachteten Gruppen wurden die Leistungsattribute im Sinne einer qualitativen Unterscheidbarkeit wie folgt zugeordnet:

- *Netzkomponenten*: Sicherheit, Stabilität, Performance
- *Maschinen/Prozessoren*: Performance, Stabilität
- *Betriebssysteme*: Performance, Sicherheit, Stabilität, Änderbarkeit
- *Storage Systeme*: Performance, Kapazität, Stabilität, Änderbarkeit
- *Datenbanken*: Performance, Kapazität, Stabilität, Änderbarkeit
- *Webserver*: Performance, Sicherheit, Stabilität, Änderbarkeit
- *J2EE Applikationsserver*: Performance, Sicherheit, Stabilität, Änderbarkeit
- *SAP*: Performance, Stabilität

Mit Hilfe von Priorisierungsmatrizen wurden die SI-Komponenten der einzelnen Gruppen aus Sicht der Architekten bzgl. der Erfüllung der zugewiesenen Leistungsattribute qualitativ bewertet. Der Erfüllungsgrad der SI-Komponenten beruht hierbei auf den persönlichen Erfahrungen und Einschätzungen der jeweiligen Architekten.

		Leistungsattribute ▸					
		Performance	**Kapazität**	**Stabilität**	**Änderbarkeit**	Aw	rw
	Gewicht	0,5	0,2	0,2	0,1	1	
Gruppe ▾							
Storage Systeme	EMC DMX X000 (FC Disks/10K)	9	9	9	3	8,4	0,19
	EMC CX (FC Disks)	3	9	3	3	4,2	0,1
	EMC CX (ATA Disks)	1	9	1	3	2,8	0,06
	IBM DS8XXX (FC DISKS/10K)	9	9	9	3	8,4	0,19
	IBM DS4/6x (FC Disks)	3	9	3	3	4,2	0,1

	Hitachi Thunder (ATA Disks)	3	9	3	3	4,2	0,1
	Hitachi Thunder (ATA Disks)	1	9	1	3	2,8	0,06
	Hitachi Lightning (FC Disks / 10K)	9	9	9	3	8,4	0,19
					Σ	44,8	1

Tabelle *10.6*: Priorisierungmatrix der Gruppe der Storage Systeme.

Ein Beispiel für eine solche Priorisierungsmatrix ist das in Tabelle 10.6 abgedruckte Ergebnis: In der Finanzdienstleistungsinformatik ist die qualitative Leistungsausprägung *Performance* besonders wichtig. Auf Grund der sehr großen Datenmengen, die in der Finanzdienstleistungsinformatik verarbeitet werden müssen, ist die Geschwindigkeit der Lese- und Schreiboperationen, die in einem Storage System ausgeführt werden, entscheidend für das Antwortzeitverhalten der gesamten Applikation. Insbesondere zeitkritische Systeme sind auf äußerst schnelle Datenzugriffe angewiesen. Die Leistungsattribute *Kapazität* und *Stabilität* sind ebenfalls sehr wichtige Faktoren, jedoch sind sie im Vergleich zur Performance von geringerer Bedeutung. Die Fähigkeit, Daten zu verarbeiten, d. h. im Fall der Storage Systeme zu speichern, ist gleich wichtig wie die Stabilität des Systems, die die Wahrscheinlichkeit repräsentiert, dass die Daten auf einer Infrastrukturkomponente nicht verloren gehen. Von untergeordneter Rolle ist das Leistungsattribut *Änderbarkeit*.

Eine Datenauswertung der in der Priorisierungsmatrix vergebenen Punkte zeigt, dass die SI-Komponenten teils qualitativ gleiche Ausprägungen besitzen. Daher wurden folgende Leistungsklassen gebildet:

- *Storage High End* = {EMC DMX X000 (FC Disks/10K); IBM DS8XXX (FC Disks/10K); Hitachi Lightning (FC Disks / 10K)}
- *Storage Mid Range* = {EMC CX (FC Disks); IBM DS4/6x (FC Disks); Hitachi Thunder (FC Disks)}
- *Storage Low End* = {EMC CX (ATA Disks); Hitachi Thunder (ATA Disks)}

10.2.1.6 Zuweisung spezifischer Einsatzgebiete und Regeln

Das spezifische Einsatzgebiet einer SI-Komponente ist ein Platzhalter, mit dem komponentenspezifische Parameter in Prosa festgehalten werden können. Primär soll dies einer Beschreibung des Einsatzgebietes dienen, die den Anwendern des SI-Katalogs zusätzliche Informationen im Sinne einer Entscheidungsunterstützung und Wissensvermittlung liefert. Eine formale Vorgabe der Beschreibungsform eines spezifischen Einsatzgebiets ist nicht Teil des Meta-Modells. Strategische Entscheidungen werden in Form der Regeln festgehalten.

Spezifisches Einsatzgebiet

Bei der Umsetzung des Meta-Modells bei der PBS wurde für die betrachteten SI-Komponenten das Feld „Spezifisches Einsatzgebiet“ als allgemeines Beschreibungsfeld genutzt. Auf Grund der Wissenskomplexität und des Wissensvolumens, mit denen jeder einzelne Architekt in seinem eigenen Themengebiet konfrontiert ist, konnte beispielsweise beobachtet werden, dass einem Domänenarchitekt, der sich hauptsächlich mit den fachlichen Anforderungen der Gesamtbanksteuerungssysteme auseinandersetzt, u. a. der Verwendungszweck von Storage Systemen unbekannt war bzw. die Aufgabe von Betriebssystemen nicht bewußt war. Ein Beispiel für das spezifische Einsatzgebiet eines Storage Systems vom Typ *EMC DMX X000 (FC DISKS/10K)* ist: *Das Storage System der Firma EMC dient der Speicherung von Daten außerhalb von Servern. StorageSysteme bestehen im Wesentlichen aus Front-End Adapter, Cache, Back-End Adapter und (FC und/oder ATA) Festplatten. Das integrierte Betriebssystem (Micro Code) bietet nach außen hin eine Administrationsschnittstelle um Design und Aufteilung der Kapazitäten sowie Bereitstellung dieser an die Systeme vornehmen zu können. Intern stellt das Betriebssystem sicher, dass Einzelausfälle (Front-/Back-End, Cache, Festplatten, Lüfter, Stromversorgungen) nicht zu Produktionsunterbrechungen führen, und dass durch intelligente Zugriffsalgorithmen eine ausbalancierte Ressourcenauslastung stattfindet. Die Leistungsfähigkeit des Storage Systems wird über die interne Dimensionierung und Konfiguration der Ressourcen erzielt.*

Regeln

Während das spezifische Einsatzgebiet allgemein verfügbare Informationen enthält, bietet die Zuweisung von Regeln die Möglichkeit, einer SI-Komponente unternehmensspezifische Entschei-

dungen auf Basis der IT-Strategie zuzuweisen. Die Zuweisung derartiger, strategischer Informationen zu einzelnen SI-Komponenten stellte sich jedoch anfangs als sehr problematisch heraus, da zum Zeitpunkt der Untersuchung im gesamten Unternehmen keine schriftlich fixierte IT-Strategie verfügbar war. Daher haben die beteiligten Infrastrukturarchitekten die Entscheidungen der zuständigen Entscheidungsträger anhand abgeschlossener Projekte analysiert und in Form von Regeln abgebildet. Eine Regelformulierung erfolgte nur dann, wenn einer SI-Komponente eindeutige, sich wiederholende und nicht widersprüchliche Entscheidungen zugewiesen werden konnten.[508]

Ein Beispiel für die Formulierung eines derartigen Regelwerks findet sich bei den SI-Komponenten, die der Gruppe der J2EE Applikationsserver zugeordnet sind. Der Komponente *BEA WebLogic 8.1* wurden folgende Regeln zugeordnet:

- Ein Komponenteneinsatz erfolgt, wenn das Wiederanfahren der Komponente innerhalb von 4 Stunden gewährleistet werden muss.
- Ein Komponenteneinsatz erfolgt, wenn die Reaktionszeit der Systemadministratoren im Fehlerfall unter einer Stunde betragen muss.
- Ein Komponenteneinsatz erfolgt, wenn die Reaktion der Systemadministration zu einem beliebigen Zeitpunkt erfolgen muss.

Im Vergleich dazu konnten der SI-Komponente *JBoss AS 4.0.0* folgende Regeln zugeordnet werden:

- Ein Komponenteneinsatz erfolgt, wenn das Wiederanfahren der Komponente innerhalb von 2 Tagen gewährleistet werden muss.
- Ein Komponenteneinsatz erfolgt, wenn die Reaktionszeit der Systemadministratoren im Fehlerfall unter 8 Stunden betragen muss.
- Ein Komponenteneinsatz erfolgt, wenn die Reaktion der Systemadministration nur zu Geschäftszeiten erfolgen erfolgen kann sowie eine direkte Reaktion nicht gewährleistet werden muss.

Die Details dieses Regelwerks zeigen, dass neben technischen Gründen auch organisatorische Gründe die Selektion einer spezifischen SI-

[508] Anmerkung: Zur Sicherstellung der Korrektheit dieser Regeln wurden selbige (für jede SI-Komponente einzeln) in Sitzungen des sog. *Architektur-Boards* unter Teilnahme des Vorstands als gültig erklärt.

Komponente beeinflussen. Die organisatorischen Gründe für die beiden SI-Komponenten beruhen darauf, dass der Betrieb der PBS die zuständigen Mitarbeiter hauptsächlich für die Administration des Applikationsservers der Firma *BEA* geschult hat. Somit stehen mehr Administratoren zur Verfügung, die eine 24-stündige Systembetreuung gewährleisten können.

Wie aus Tabelle 10.7 hervorgeht, wurde zwischen beiden Applikationsservern Leistungsunterschiede festgestellt. *BEA WebLogic 8.1* ist auf Grund seiner Attributausprägungen Teil der Leistungsklasse *J2EE High End*. Im Gegensatz dazu ist *JBoss AS 4.0.0* Bestandteil der Leistungsklasse *J2EE Low End*. Das zuvor beschriebene Regelwerk kann nun zu dem Fall führen, dass in einem Architekturentwurf, in dem die leistungstechnischen Anforderungen lediglich eine SI-Komponente der Leistungsklasse *J2EE Low End* beschreiben, ein Architekt trotz der geringeren Anforderungen eine SI-Komponente der Leistungsklasse *J2EE High End* auswählt. Dieser Fall tritt genau dann ein, wenn der Architekt in Anlehnung an das Regelwerk mit Hilfe einer gezielten Befragung des Anforderers die Existenz einer mit der SI-Komponente *BEA WebLogic 8.1* verknüpften Regel feststellt. Verlangt der Anforderer beispielsweise eine Reaktionszeit der Systemadministratoren im Fehlerfall innerhalb einer Stunde, und beschreibt er gleichzeitig eine leistungstechnische Anforderung, die derjenigen eines *JBoss AS 4.0.0* entspricht, so wird der Architekt die SI-Komponente *BEA WebLogic 8.1* in den Architekturentwurf aufnehmen, obwohl diese aus leistungstechnischer Sicht überqualifiziert ist.

		Leistungsattribute ▸						
		Performance	**Sicherheit**	**Stabilität**	**Änderbarkeit**	**Herstellersupport**	Aw	Rw
	Gewicht	0,15	0,2	0,15	0,1	0,4	1	
Gruppe ▾								
J2EE Applikationsserver	BEA Web Logic 8.1	9	9	9	3	9	8,4	0,71

	JBoss AS 4.0.0	9	3	9	1	0	3,4	0,29
						Σ	11,8	1

Tabelle *10.7*: Priorisierungmatrix der Gruppe der J2EE Applikationsserver.

10.2.1.6.1.1 Modellierung der SI-Komponenten

Der Prozessschritt dient der Modellierung aller relevanten SI-Komponenten. In Anlehnung an das Meta-Modell werden die SI-Komponenten in Form von Klassen auf der Modellebene M1 entworfen. Sie sind Instanzen der Meta-Klassen der Modellebene M2.

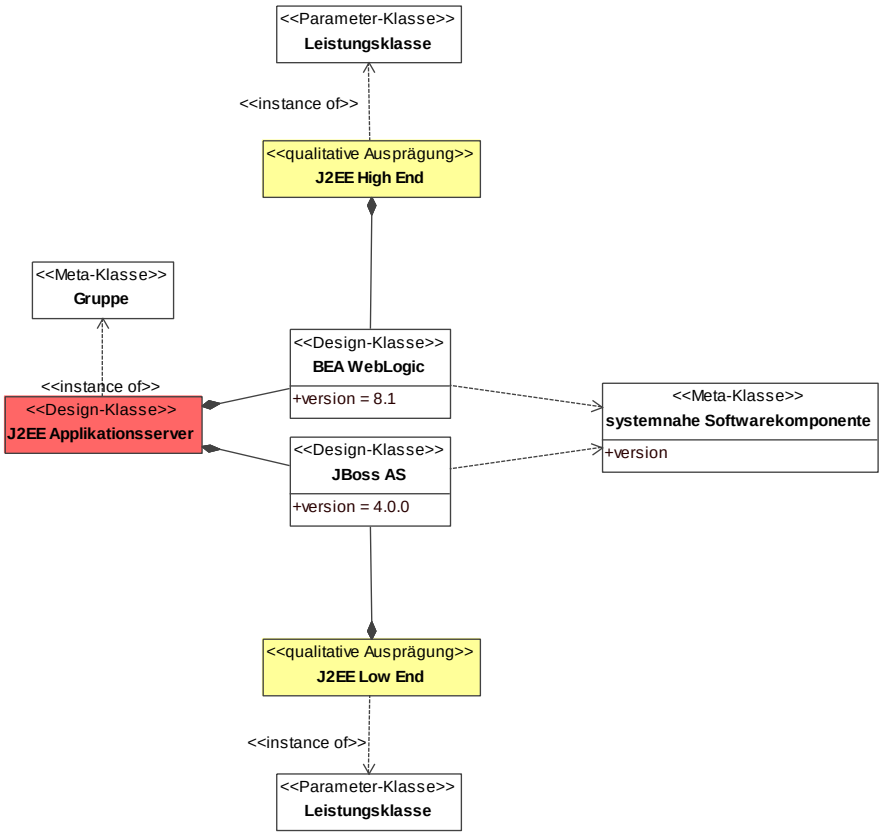


Abbildung 10.1: Modellierung am Beispiel der SI-Komponenten der Gruppe J2EE Applikationsserver. (Quelle: Eigene Darstellung)

Abbildung 10.1 zeigt die Modellierung der SI-Komponenten der Gruppe *J2EE Applikationsserver*: Als Instanzen der Meta-Klasse *systemnahe Softwarekomponente* auf der Modell-Ebene M1 werden den Klassen *BEA WebLogic* und *JBoss AS* mit dem Attribut *version* gekennzeichnet. Neben der grafischen Zuweisung zur Gruppe der *J2EE Applikations-server* werden auch die qualitativen Ausprägungen in diesem Modell

festgehalten. Anhand der Ergebnisse aus Tabelle 10.7 wurden zwei Leistungsklassen gebildet – *J2EE High End* und *J2EE Low End*. Die Komposition zwischen *BEA WebLogic* und *J2EE High End* bzw. zwischen *JBoss AS* und *J2EE Low End* weist den beiden SI-Komponenten eine eindeutige Zugehörigkeit zu. Analog wurde bei der in Abbildung 10.2 abgedruckten Modellierung der SI-Komponenten der Gruppe *Storage Systeme* verfahren.

Wie aus den beiden Abbildungen hervorgeht, wurden sowohl die Leistungsattribute als auch die Leistungsattributausprägungen nicht modelliert. Der Grund ist, dass die Modellierung einer Gruppe mit mehr als zwei SI-Komponenten bereits für eine Unübersichtlichkeit der einzelnen UML-Diagramme sorgt: In dem Diagramm der *Storage Systeme* müssten beispielsweise 4 Leistungsattribute modelliert werden, die jeweils mit allen 8 Klassen assoziiert werden müssten. Jede dieser 32 Assoziationen müsste wiederum mit einer von 32 Leistungsattributausprägungen assoziiert werden. Die Modellierung der Leistungsattribute und deren Attributausprägungen erhöht demnach die Komplexität und Unübersichtlichkeit eines einzelnen Diagramms. Durch das Festhalten der Leistungsattributausprägungen in Matrizenform wird jedoch die Modellkonsistenz gewährleistet.

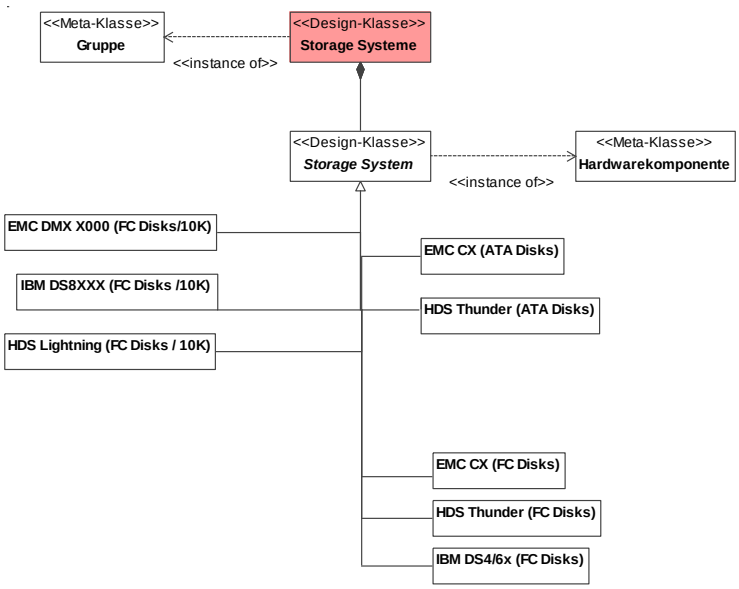


Abbildung *10.2*: Modellierung am Beispiel der SI-Komponenten der Gruppe Storage Systeme. (Quelle: Eigene Darstellung)

Da das Konzept des SI-Katalogs jedoch einen hybriden Ansatz verfolgt, bei dem die Modellinformationen sowohl grafisch als auch Matrix-basiert festgehalten werden, wurde im Rahmen der Untersuchung auf die Modellierung dieser Modellinhalte zu Gunsten der Modellbezogenen Komplexität verzichtet. Ebenso wurde auf die Abbildung des spezifischen Einsatzgebiets innerhalb des grafischen Modells verzichtet.

10.2.1.7 Abstimmungsprozess innerhalb der Organisation

Das betrachtete Unternehmen arbeitet Prozessorientiert. Dies macht es erforderlich, dass sämtliche entscheidungsrelevanten Handlungsschritte definiert und durch das höhere Management verabschiedet werden müssen.

Das Konzept des SI-Katalogs beinhaltet die Selektion spezifischer Infrastrukturkomponenten sowie deren Komposition zu Architekturmustern. Auf Grund der organisatorischen Aufteilung des Unternehmens in unterschiedliche Ressorts kann es bei einzelnen Entscheidungen zu Spannungen zwischen den beteiligten Personen kommen. Ein Grund für derartige Spannungen liegt in der Natur der organisatorischen Aufspaltung begründet: Das Aufgabengebiet des Ressorts *Technologiemanagement* ist die Pflege und Weiterentwicklung der einzelnen in den Informationssystemen eingesetzten Technologien. Demgegenüber steht das Aufgabengebiet des Ressorts *Betrieb*, das den technischen Anteil eines Informationssystems überwacht, administriert und steuert. Das Ressort Betrieb hat mehr Mitarbeiter und verantwortet ein viel größeres Budget als das Ressort Technologiemanagement. Hierarchich sind die Mitarbeiter beider Ressorts inetwa gleich positioniert.

Spannungen entstehen beispielsweise durch die unterschiedlichen Aufgabengebiete beider Ressorts: Während sich die einen die Kompetenz zusprechen, durch architektonische Überlegungen und Vorgaben die Entwicklungen der Informationssysteme des Finanzdienstleisters unter wirtschaftlichen Gesichtspunkten zu überwachen und zu steuern, sprechen die anderen dieser Gruppe jene Kompetenz ab. Auslöser hierfür ist der Anspruch der Mitarbeiter des Ressorts Betrieb, durch ihren direk-

ten Kontakt zu den „betriebenen“ Informationssystemen kompetentere Entscheidungen bzgl. der Informationssystementwicklungen unter wirtschaftlichen Gesichtspunkten treffen zu können. Das elitäre Flair, das der Rolle des Architekten u. a. in der Literatur zugesprochen wird, in Kombination mit der Personifizierung dieser Rolle in einer großen Organisation erschwert die Zusammenarbeit in dem betrachteten Unternehmen.

Aus diesem Grund musste der in Abbildung 10.3 dargestellte Abstimmungsprozess geschaffen werden. Der Prozess beschreibt die Vorgehensweise bei der Verabschiedung von SI-Komponenten zwischen den beiden Rollen des Architekten und relevanter Stakeholder (= Betrieb). Sowohl der Abstimmungsprozess als auch der Prozess innerhalb des SI-Katalogs wurden abgestimmt und organisatorisch verankert.

Da der Abstimmungsprozess organisatorisch begründet ist und als Spezifikum des betrachteten Unternehmens angenommen wird, ist er nicht expliziter Bestandteil des SI-Katalogkonzepts.

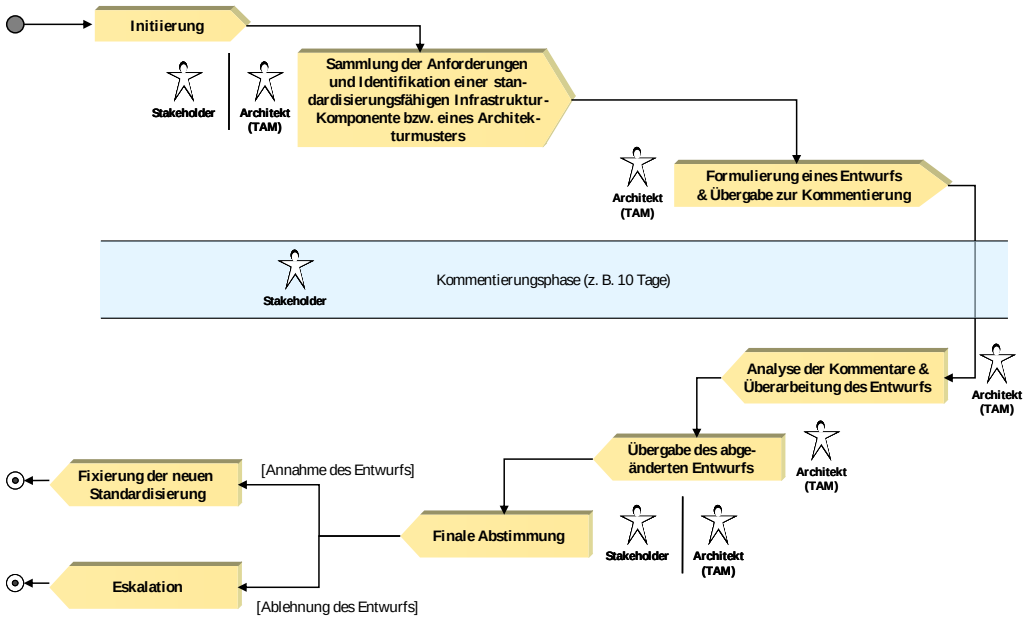


Abbildung 10.3: Prozess zur Verabschiedung von Architekturmustern. (Quelle: Eigene Darstellung)

10.2.1.8 Entwurf der SI-Architekturmuster

Die Entwurf der Architekturmuster ist ein inkrementeller und kreativer Prozess: Skizzen werden angefertigt und auf ihre Tauglichkeit über-

prüft. Architekturattribute werden ermittelt und Architekturmustern zugewiesen.

In diesem „Findungsprozess“ werden Architekturmuster Rang-bezogen und inkrementell entworfen. Wird ein Architekturmuster als tauglich befunden, wird es aus der Skizze in ein Modell überführt. Da dieser Prozess zum Teil Architekturmuster hervorbringt, die entweder nicht vollständig sind oder sogar nicht verwendbar sind, werden die im Meta-Modell aufgelisteten Prozessschritte im Folgenden nicht explizit aufgeführt. Eine wesentliche Hilfe für diesen Entwurfsprozess stellen die Gruppendominanzen dar: Ausgehend von den am wenigsten dominanten Gruppen werden einzelne Muster entworfen, qualitativ beurteilt, modifiziert und zur Wiederverwendung in anderen Mustern angepasst.

Während der Entwurfsphase wurde festgestellt, dass die ein- oder andere Lösung bereits in früheren Projekten skizziert wurde. Die Tatsache, dass viele dieser früheren Lösungen mit größtenteils unterschiedlichen Notationen in Power Point entworfen wurden und stets nur Teil einer Gesamtpräsentation der Phasen Vorstudie bzw. Grobentwurf waren, erschwerte jedoch die Suche. In diesen Fällen änderte sich die Suche von einem „Welche Lösungen für Datenbankarchitekturen existieren?“ zu einem „In welchen Projekten wurden Datenbankarchitekturen benötigt?“

Skizzierung vor Modellierung

Da der Entwurf eines Architekturmusters auf dieser Meta-Ebene sehr abstrakt ist, und weil Architekturmuster sehr komplex werden können, hat sich herausgestellt, dass die Vorab-Skizzierung der späteren Instanz eines solchen Musters (wie in Abbildung 10.4 dargestellt) den Entwurf des eigentlichen Architekturmusters enorm unterstützt. Auf Grund früherer Erfahrungen und mit Hilfe der identifizierten Abhängigkeiten zwischen den einzelnen Gruppen des SI-Katalogs wurden die Skizzen als Kommunikationsinstrument zwischen den Architekten verwendet. Bevor ein Architekt jedoch innerhalb seiner Wissensdomäne Architekturen skizziert ist es sinnvoll zuvor die Architekturattribute zu spezifizieren. Die von Architekturattributen gesteuerte Skizzierung hat im Gegensatz zur situativ getriebenen Skizzierung den Vorteil, die qualitativen Anforderungen an Architekturmuster in den Vordergrund zu stellen. Bei der anfangs situativ getriebenen Skizzierung fiel auf, dass

die Architekten zwei oder mehr unterschiedliche Skizzen mit qualitativ gleich ausgeprägten Architekturattributen entworfen hatten. Anschließend musste der entsprechende Architekt eine der Skizzen auswählen, um diese in ein Architekturmuster übersetzen zu können.

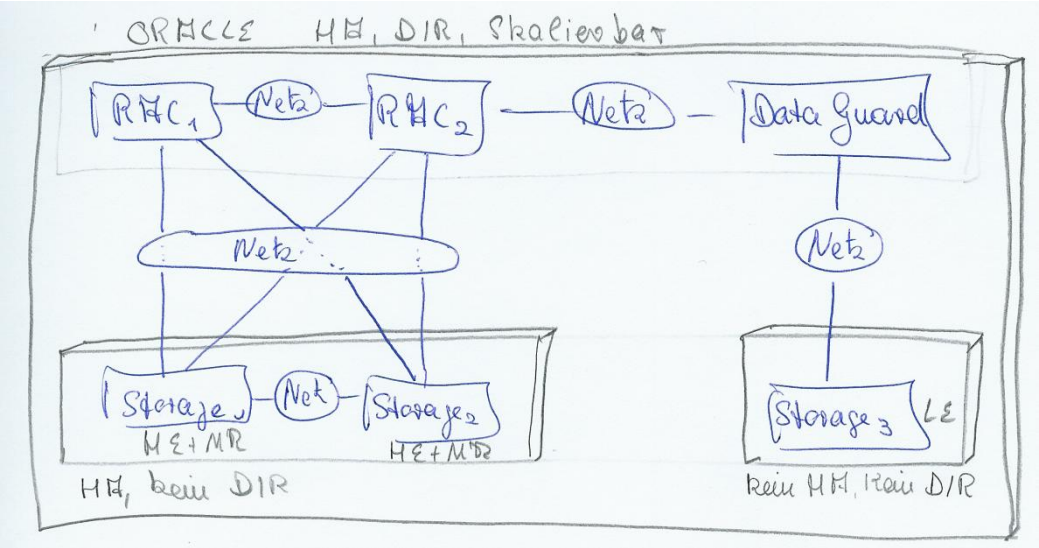


Abbildung 10.4: Skizzierung eines Architekturmusters für ein hochverfügbares, D/R-fähiges und skalierbares Datenbanksystem mit Oracle-Komponenten. (Quelle: Eigene Darstellung)

Architekturattribute

Die von Architekturattributen gesteuerte Skizzierung erforderte initiale Workshops, auf denen die Anforderungen an Infrastrukturarchitekturen in der Finanzdienstleistungsinformatik gesammelt wurden. In einer ersten Workshoprunde wurden die Teilnehmer gebeten, die Anforderungen an die Infrastrukturarchitektur im Allgemeinen mittels Architekturattributen zu charakterisieren. Diese Vorgehensweise führte jedoch zu folgender Erkenntnis:

- Das Ergebnis ist eine Menge M_{aA} allgemeingültiger Architekturattribute AA_i (i = 1, 2, …) die eine Informationssystemübergreifende Anwendbarkeit suggeriert.
- Die Identifikation der Architekturattribute basiert jedoch auf den Erfahrungen einzelner Personen innerhalb ihrer Wissensdomäne. Und diese ist zum größten Teil Informationssystemspezifisch.
- Wird nun versucht M_{aA} mit einem Architekturmuster zu verknüpfen, so gibt es eine Vielzahl einzelner Architekturattribute, die nicht auf die Infrastrukturkomponentenkompositionen innerhalb des Musters passen. Beispielsweise besitzen die Architekturattribute „Long Dis-

tance Unterstützung“ bzw. „Short Distance Unterstützung“, die die Qualitätsausprägungen eines Storage Systems beschreiben, keine Anwendbarkeit auf die Architektur eines J2EE Applikationsservers. Während der Workshopdurchführungen sorgte die Ignoranz dieser Zusammenhänge zu teils heftigen Kontroversen zwischen den Workshopteilnehmern.

Auf Basis dieser Erkenntnis wurden in Einzelsitzungen mit dem jeweils für eine Obergruppe verantwortlichen Architekten Obergruppenspezifische Architekturattribute identifiziert. Wird beispielsweise in einem Architekturmuster der Obergruppe *Datenbanken* ein Architekturmuster der Obergruppe *Storage Systeme* verwendet, so sind die für die *Storage Systeme* definierten Architekturattribute implizit auch für die Architekturmuster der *Datenbanken* gültig.

Eine weitere Erkenntnis der Attributierung von Architekturmustern ist, dass die Architekturattribute im Gegensatz zu den Leistungsattributen einzelner Infrastrukturkomponenten härter ausgeprägt sind. Während die Ausprägung eines Leistungsattributs im Vergleich zu vergleichbaren Infrastrukturkomponenten mit den Werten 0, 1, 3 und 9 charakterisiert wurde, stellte sich bei der Anwendung derselben Verfahrensweise heraus, dass die Korrelation zwischen Architekturmustern und Architekturattributen stets mit den Werten 0 und 9 bewertet wurde. Ebenso konnte festgestellt werden, dass keine Gewichtung zwischen Architekturattributen verlangt wurde. Es lässt sich daher vermuten, dass es sich bei den Architekturattributen um gleichwichtige Qualitätsanforderungen handelt, die von den Architekturmustern entweder erfüllt („ja“) oder nicht erfüllt („nein“) werden. Diese Beobachtung führt dazu, dass anstatt einer Priorisierungsmatrix eine Bool’sche, ungewichtete Entscheidungsmatrix vorgeschlagen wird. Beispiele für derartige Entscheidungsmatrizen sind die in Tabelle 10.8 und Tabelle 10.9 abgedruckten Korrelationen zwischen Architekturmustern und Architekturattributen der Obergruppen *Datenbanken* und *Storage Systeme*.

		Architekturattribute ▸		
		HA-Fähigkeit	**D/R-Fähigkeit**	**Skalierbarkeit**
Gruppe ▾				
Datenbanken	DB (Skalierbar, D/R, HA) Pattern	●	●	●
	DB (D/R, HA) Pattern		●	
	DB (Skalierbar, D/R) Pattern			●
	DB (D/R) Pattern		●	
	DB (Skalierbar) Pattern			●

Tabelle *10.8*: Entscheidungsmatrix der Architekturmuster und der Architekturattribute der Gruppe Datenbanken.

Regeln

Während des Architekturmusterentwurfs wurde festgestellt, dass die Entscheidungsmatrizen keine strategischen Entscheidungen abbilden können. So kann es durchaus vorkommen, dass eine Anforderung in einem konkreten Projekt mit einem Architekturmuster der Qualität *Q* erfüllt werden kann, eine strategische Entscheidung seitens des Managements jedoch eine wesentlich höhere Qualität als *Q* verlangt. Konkret handelt es sich bei den strategischen Vorgaben um Regeln, die bei der Selektion eines Architekturmusters beachtet werden müssen. Da diese Regeln jedoch nicht für jedes Architekturmuster vorhanden sind, besitzt ein Architekturmuster ex lege eine *Default Regel*, die qualitativen Eigenschaften des Musters betrachtet und die dann in Kraft tritt, wenn dem Muster keine *Strategische Regel* zugeordnet ist.

		Architekturattribute ▸							
		HA-Fähigkeit	**D/R-Fähigkeit**	**RPO = 0**	**RPO > 0**	**RTO = 0**	**RTO > 0**	**Short Distance Unterstützung**	**Long Distance Unterstützung**
Gruppe ▾									
Storage Systeme	Storage (Einfach) Pattern			●			●	●	
	Storage (Dual) Pattern	●		●		●		●	
	Storage (Synchron) Pattern	●		●			●	●	

	Storage (Asynchron) Pattern		●		●		●	●	●
	Storage (Synchron + Asynchron) Pattern	●	●	●			●	●	●

Tabelle *10.9*: Entscheidungsmatrix der Architekturmuster und der Architekturattribute der Gruppe Storage Systeme.

Ein Beispiel für eine Strategische Regel der Gruppe *Storage Systeme* ist:

> *Wenn Infrastrukturkomponenten der Leistungsklasse „Storage Low End" konsolidiert werden sollen oder können, sind für diese Konsolidierung Infrastrukturkomponenten der Leistungsklasse „Storage Mid Range" zu wählen.*

Zwei Beispiele für Strategische Regeln der Gruppe *SAP R/3* sind:

> *Wenn die Anforderung Geschäftsprozesse beschreibt, die auf die sog. Asset-Datenmodelle (Geschäftspartner, Konten, Depots, Kredite, Hypotheken etc.) zurückgreifen, sind Infrastrukturkomponenten der Leistungsklasse „SAP High End" zu wählen.*

> *Wenn die Anforderung entweder geschäftskritische bzw. Kundengerichtete Geschäftsprozesse beschreibt, oder wenn das zu entwerfende System ein Abwicklungssystem bzw. ein die Bank unterstützendes System (Recherche, Zahlungsverkehr, Rating, Reporting etc.) ist, sind Infrastrukturkomponenten mindestens der Leistungsklasse „SAP Mid Range" zu wählen.*

Verwendungsmatrix

Mit der Erstellung der Vewendungsmatrix, in der die Wiederverwendungen der einzelnen Architekturmuster festgehalten werden, wurde erst nach Abschluss der gesamten Entwurfsphase begonnen. Ein Grund hierfür ist, dass der Entwurf der Architekturmuster ein kreativer und inkrementeller Prozess ist, in dem Entwürfe häufig revidiert werden. Ein Ausschnitt der Verwendungsmatrix ist in Tabelle 10.10 abgedruckt.

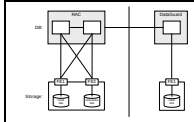
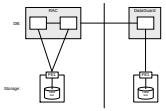
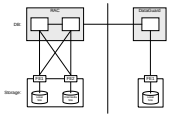
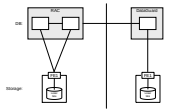
			Verwendet ▸									
			Datenbanken							Storage Systeme		
												
Architekturmuster ▾			DB (Skalierbar, D/R, HA) Pattern	DB (Skalierbar) Pattern	...	Oracle RAC 9i	Oracle Single Instance 9i	Oracle Data Guard	...	Storage (Einfach) Pattern	Storage (Dual) Pattern	...
Datenbanken		DB (Skalierbar, D/R, HA) Pattern			...	⊙ {2..n}		⊙	...	⊙	⊙	...
		DB (Skalierbar, D/R) Pattern			…	⊙ {2..n}		⊙	…	⊙ {2}		…
...	...	...	…	…	…	…	…	…	…	…	…	…

Tabelle *10.10*: Ausschnitt aus der Verwendungsmatrix der Architekturmuster und SI-Komponenten.

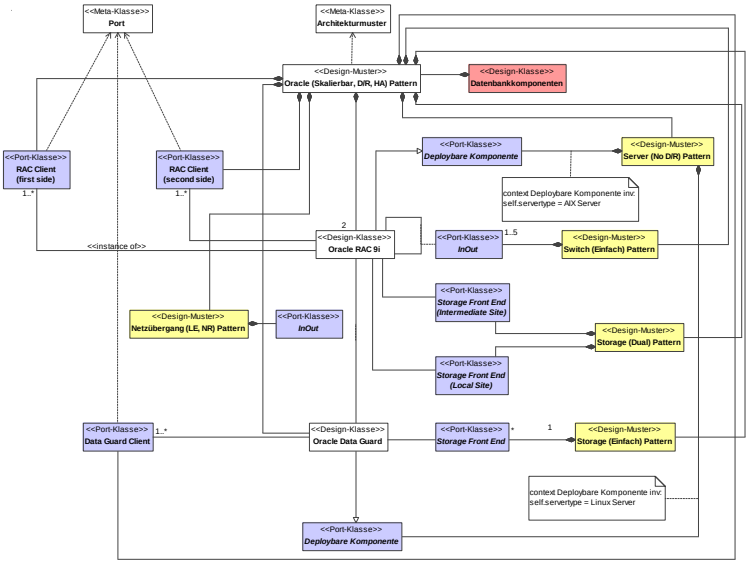


Abbildung 10.5: Modellierung eines Architekturmusters der Gruppe Datenbanken. (Quelle: Eigene Darstellung)

10.3 Anwendung des SI-Katalogs

Die Anwendung des SI-Katalogs wird durch das Ereignis E2 ausgelöst. Das Ziel dieses Prozessschritts ist die Selektion geeigneter Architekturmuster und die Konfiguration desselben. Da ein zu einem Architekturmuster aggregiertes Element ebenfalls ein Architekturmuster sein kann, ist auch die Anwendung des SI-Katalogs ein iterativer Prozess. Zur Demonstration des Prozesses wird folgende Anforderung zu Grunde gelegt:

Beispielanforderung:

Das zu entwerfende System erfordert eine hochperformante und ausfallsichere Datenbank für die ein administrativer 24x7 Support gewährleistet werden muss. Da das System Kundendaten verarbeitet darf kein Datenverlust auftreten. Zudem muss gewährleistet sein, dass ein Systemausfall praktisch nicht auftritt. Auf dem System werden ca. 500 User operieren, die hohe bis sehr hohe Transaktionslasten produzieren. Bei einer Transaktion werden zwischen 5kB und 8kB Daten gelesen bzw. weniger als 1kB Daten geschrieben. Pro User werden bis zu 10 Transaktionen zu Spitzenzeiten parallel ausgeführt.

Implizit werden zur beispielhaften Anwendung des SI-Katalogs die im Meta-Modell beschriebenen Prozessschritte *Selektion eines geeigneten SI-Architekturmusters* sowie *Konfiguration des SI-Architekturmusters* in mehreren Iterationsschritten ausgeführt. Um den folgenden Ausführungen eine möglichst überschaubare Struktur zu verleihen, werden die Ausführungen der beiden Prozessschritte nicht in separaten Abschnitten erläutert, sondern im Abschnitt *Iterative Selektion und Konfiguration* zusammengefasst.

10.3.1 Identifikation der typischen, architektonischen und leistungstechnischen Aspekte

Voraussetzung für die Anwendbarkeit des SI-Katalogs ist die Zerlegbarkeit einer Anforderung in die drei Aspekte: typisch, architektonisch und leistungstechnisch.[509] Bevor der SI-Katalog auf die Beispielanforderung angewendet werden kann, muss diese in die drei beschriebenen Aspekte zerlegt werden. Die Zerlegung wurde mit Hilfe einer Nominalphrasenanalyse durchgeführt:

- Der *typische Aspekt* der Anforderung leitet sich aus der Benennung der zu verwendenden Obergruppe ab. Gesucht wird eine Datenbank.
- Die Identifikation des *architektonischen Aspekts* erfordert die Zerlegung der Anforderung in architektonisch-qualitative Eigenschaften. Als solche können folgende Anforderungen ausgemacht werden: Vorbeugung eines Datenverlustes (= D/R-Fähigkeit) sowie Vorbeugung eines Systemausfalls (= HA-Fähigkeit).
- Der dritte, *leistungstechnische Aspekt*, beschreibt die leistungstechnisch-qualitativen Eigenschaften des zu entwerfenden Systems. Indikatoren dieser Aspektbeschreibung sind die Anforderungen bzgl. einer hochperformanten Datenbank, welche hohe Transaktionslasten (= 500 User je 10 Transaktionen je 9kB Datenvolumina) verarbeiten kann und zudem ausfallsicher (= stabil) ist. Die Änderbarkeit und die

[509] Vgl. Abschnitt 8.1.2

Kapazität wurden nicht explizit erwähnt und spielen daher eine untergeordnete Rolle.

10.3.2 Iterative Selektion und Konfiguration

Mit Hilfe der im letzten Abschnitt identifizierten Aspekte der Anforderung kann der Algorithmus zur Selektion und Konfiguration der SI-Architekturmuster angewendet werden:

F_1: Selektion der OG

Die den Problemaspekt *typisch* abdeckende Obergruppe ist gemäß Tabelle 10.3 die Gruppe der *Datenbanken*. Nach der Selektion der Obergruppe wird aus der dieser Obergruppe zugeordneten Menge der Architekturmuster M_{DB} dasjenige Architekturmuster ausgewählt, das die architektonisch-qualitativen Anforderungen am geeignetsten abdeckt.

		Leistungsattribute ▸					
		Performance	**Kapazität**	**Stabilität**	**Änderbarkeit**	Aw	Rw
	Gewicht	0,4	0,2	0,39	0,01	1	
Gruppe ▾							
Datenbanken	Oracle RAC 9i	9	9	9	3	8,9	0,26
	Oracle Single Instance 9i	9	3	9	3	7,7	0,23
	Oracle DataGuard	3	3	9	3	5,3	0,16
	Microsoft SQL Server 2000	3	3	3	3	3	0,09
	IBM DB2 8	9	9	9	3	8,9	0,26
					Σ	33,9	1

Tabelle *10.11*: Priorisierungmatrix der Gruppe der Datenbanken.

F_1: Selektion des Architekturmusters *P*

Ein Blick auf Tabelle 10.8 zeigt die MDB zugeordneten Architekturattribute *HA-Fähigkeit*, *D/R-Fähigkeit* und *Skalierbarkeit*. Der zuvor identifizierte architektonische Aspekt der Anforderung verlangt nach einem Architekturmuster dass sowohl *HA*- als auch *D/R-Fähigkeit* aufweist. Das MDB ebenfalls zugeordnete Architekturattribut *Skalierbarkeit* ist

nicht expliziter Bestandteil der Anforderung. Würde es nun zwei Architekturmuster geben, deren Architekturattribute bis auf das der *Skalierbarkeit* identisch ausgeprägt sind, müsste der Architekt in diesem Fall den Anforderer explizit nach der Ausprägung der Skalierbarkeit der gesuchten Datenbank befragen. Da die architektonisch-qualitative Anforderung nur ein Architekturmuster *P* der Menge MDB mit *P = DB (Skalierbar, D/R, HA) Pattern* zulässt, erübrigt sich die erneute Befragung. Das selektierte Architekturmuster P ist Inputparameter für den rekursiven Algorithmus F_2.

F2:
Konfiguration von ***P***

Sowohl aus dem in Abbildung 10.5 abgedruckten Architekturmuster für P als auch aus der in Tabelle 10.10 abgedruckten Verwendungsmatrix geht hervor, dass direkt zwei SI-Komponenten zu P komponiert sind: *Oracle RAC 9i* und *Oracle Data Guard*. Bei den übrigen zu P komponierten Elementen handelt es sich um Architekturmuster.

Da die zu P komponierten SI-Komponenten *Oracle RAC 9i* und *Oracle Data Guard* nicht generalisiert sind, müssen sie zunächst daraufhin überprüft werden, ob sie den leistungstechnischen Aspekt der Anforderung abdecken. Die in Tabelle 10.11 abgedruckte Priorisierungmatrix zeigt die qualitative Beurteilung der Gruppe der Datenbanken aus Sicht der zuständigen Infrastrukturarchitekten. Explizit wird in der Anforderung eine hochperformante Datenbank gesucht, die hohe Transaktionslasten verarbeiten kann. Der Indikator *Performance* ist expliziter Bestandteil der Leistungsattribute der Gruppe der Datenbanken. Die allgemeingehaltene Formulierung „hoch“ wird durch eine spezifische Metrik der Transaktionslasten manifestiert. Im Gegensatz zur Performance ist der Indikator *Transaktionslast* impliziter Bestandteil der Menge der Leistungsattribute. Im betrachteten Fallbeispiel entscheidet der zuständige Infrastrukturarchitekt, dass Transaktionlast eine qualitative Komposition der Leistungsattribute Performance, Kapazität und Stabilität ist. Alle Attributausprägungen der SI-Komponente *Oracle RAC 9i* deuten auf eine Eignung derselbigen zur problemadäquaten Abdeckung des leistungstechnischen Aspekts hin.

Die SI-Komponente *Oracle DataGuard* ist eine sog. *Standby-Datenbank*, die eine transaktionskonsistente Kopie der produktiven Primärdatenbank Oracle RAC auf einem zweiten Rechnersystem rep-

räsentiert. Da diese SI-Komponente keine direkten Anfragen eines Clients entgegennimmt ist lediglich ihre Stabilität von großer Bedeutung. Auch diese qualitative Eigenschaft wird von der Komponente erfüllt.

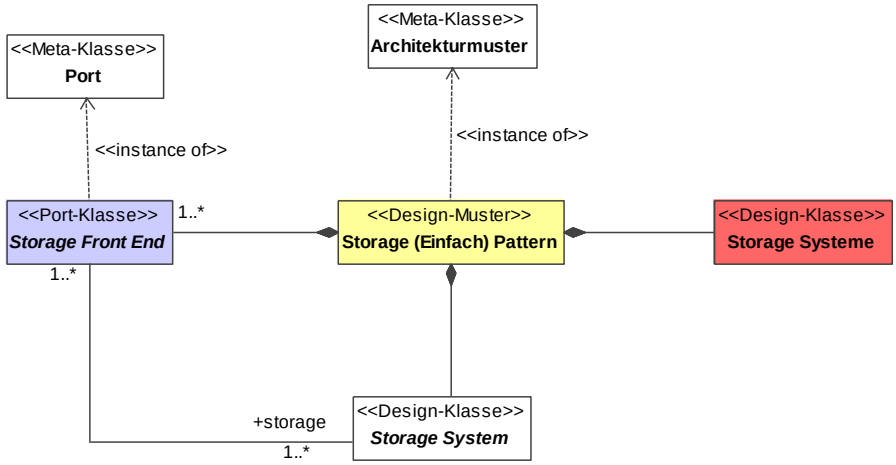


Abbildung 10.6: Modellierung der SI-Komponenten der Gruppe Storage Systeme. (Quelle: Eigene Darstellung)

Bzgl. der Kardinalitäten sieht das Architekturmuster mindestens 2 Instanzen der SI-Komponente *Oracle RAC 9i* vor. Die SI-Komponente *Oracle DataGuard* muss genau ein Mal instanziiert werden. In Bezug auf den leistungstechnischen Aspekt der Anforderung ist dies problemadäquat.

Der Algorithmus F_2 ist rekursiv aufgebaut indem er für jedes zu P aggregierte Architekturmuster P' aufgerufen wird. Im betrachteten Fall sind P insgesamt 5 Architekturmuster zugeordnet, denen einzeln wiederum Architekturmuster zugeordnet sind. Zur Verdeutlichung der rekursiven Ausführung wird daher die Rekursion anhand des zu P komponierten Architekturmusters *P'* der Gruppe *Storage Systeme* vorgeführt.

F2: Konfiguration von ***P'***

Sei P' das zu P komponierte Architekturmuster *Storage (Einfach) Pattern*. Innerhalb dieses Iterationsschrittes wird F2 analog zur vorherigen Anwendung ausgeführt. Wie aus Abbildung 10.5 hervorgeht, besteht eine 1:1 Assoziation zwischen der SI-Komponente *Oracle DataGuard* und dem Porttyp *Storage Front End* des Architekturmusters *Storage (Einfach) Pattern*, welches in Abbildung 10.6 abgedruckt ist. Der Port *Storage Front End* wiederum ist mit der abstrakten Klasse *Storage System* assoziiert.

Aus Abbildung 10.2 geht hervor, dass der abstrakte Typ *Storage System* eine Generalisierung aller SI-Komponenten der Gruppe *Storage Systeme* darstellt. Gemäß F2 erfordert eine generalisierte, zu P' komponierte Komponente, eine Selektion der problemadäquatesten Spezialisierung dieser Komponente unter Beachtung möglicher zugeordneter Regeln und mit Berücksichtigung des Leistungstechnischen Aspekts: Da der Gruppe der *Storage Systeme* nur eine Regel[510] zugeordnet ist, die die in dem betrachteten Anwendungsfall nicht notwendige Konsolidierung von Infrastrukturkomponenten betrifft, verbleibt eine Selektion der SI-Komponente nach leistungstechnischen Aspekten gemäß Tabelle 10.6. Um den in der Beispielanforderung geforderten qualitativen Ansprüchen gerecht zu werden sind *-viele Instanzen einer SI-Komponente der Leistungsklasse *Storage Mid Range* erforderlich.
Somit stehen drei mögliche SI-Komponenten zur Auswahl. Eine spezifische Selektion kann zu diesem Zeitpunkt entfallen, sofern die Kompatibilität zwischen der SI-Komponente *Oracle DataGuard* und den in Frage kommenden Storage Systemen uneingeschränkt ist. Weiterhin kann eine Selektion auf einen späteren Zeitpunkt verschoben werden, sofern die SI-Komponenten preislich auf gleichem Niveau liegen. Ohne die Vorgabe strategischer Regeln zur spezifischen Selektion ist eine Selektion zu diesem Zeitpunkt unnötig.

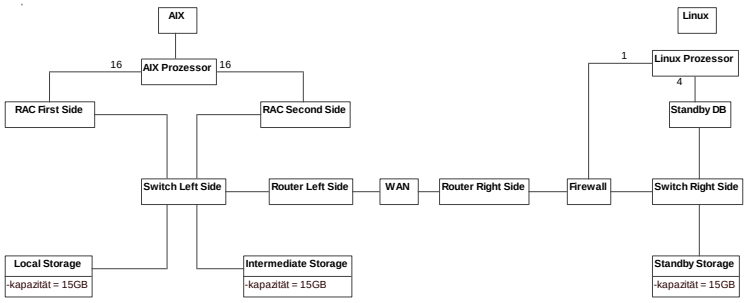


Abbildung 10.7: Konfiguration der instanziierten Modellelemente auf Meta-Ebene M0. (Quelle: Eigene Darstellung)

Instanzierung der Konfiguration auf Meta-Ebene M0

Gemäß Abschnitt 6.2 werden im Meta-Modell des SI-Katalogs die abstrakten Zusammenhänge derjenigen Meta-Elemente beschrieben, die innerhalb des Layers M1 die Architekturmuster komponieren. Wird ein

[510] Vgl. Abschnitt 10.2.1.8

Architekturmuster innerhalb eines Softwareentwicklungsprojekts instanziiert, so wird diese Laufzeitinstanz auf Layer M0 konkretisiert.

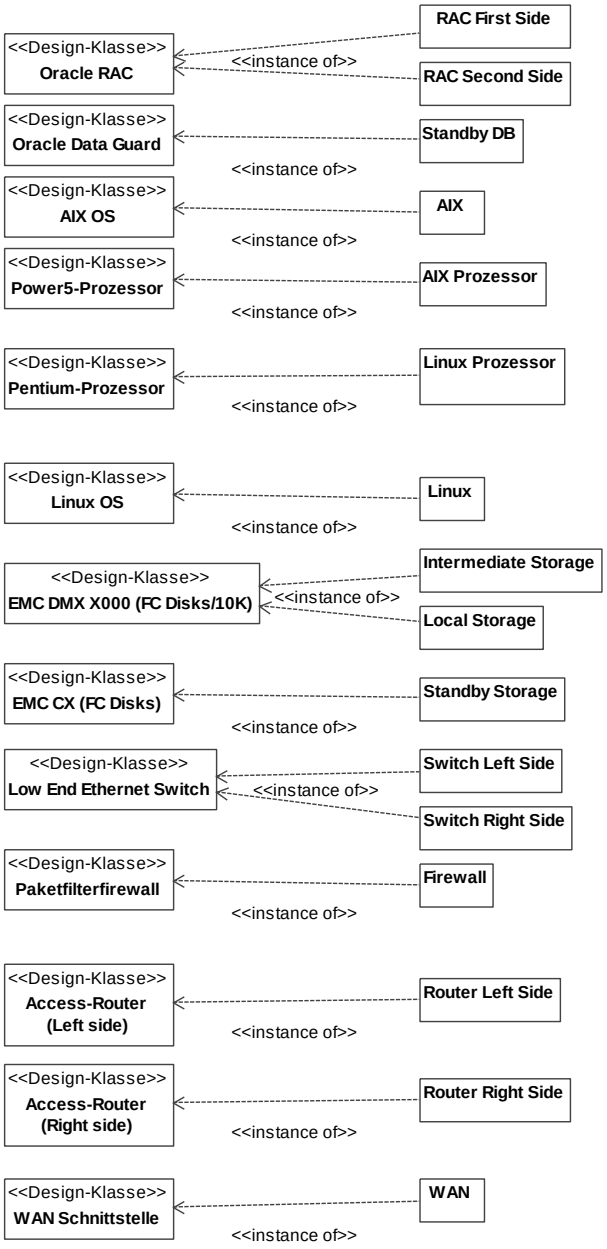


Abbildung 10.8: Instanziierung der Modellelemente auf Meta-Ebene M0. (Quelle: Eigene Darstellung)

Das Ergebnis der iterativen Selektion und Konfiguration innerhalb des betrachteten Prozessschritts ist eine Menge von SI-Komponenten, deren Zusammenhang durch die konfigurierten Architekturmuster beschrieben wird. Ihre Instanziierung ergibt eine Anwendungsfallbezogene Assoziation unterschiedlicher Instanzen der SI-Komponenten. Die Instanziierung der Konfiguration des in diesem Abschnitt betrachteten Anwendungsfalls ist in Abbildung 10.7 dargestellt. Die UML-technische

Beschreibung der in Abbildung 10.7 verwendeten Elemente ist in Abbildung 10.8 abgedruckt.

10.4 Pflege des SI-Katalogs

Die *Pflege des SI-Katalogs* ist die dritte Phase in die der SI-Katalog im Rahmen seines Lebenszyklus eintreten kann. Der Phaseneintritt wird durch das Ereignis *E3* ausgelöst, welches entweder das Ende eines zeitlichen Zyklus repräsentiert, die Einführung neuer Infrastrukturkomponenten bedeutet oder das Einpflegen eines neuen Architekturmusters bedeutet.

10.4.1 Verankerung der Zuständigkeiten innerhalb der Organisation

Im Fall des untersuchten Unternehmens muss die Pflege des SI-Katalogs prozesstechnisch definiert und vorgeschrieben werden. Andernfalls würde sie in dem betrachteten Unternehmen als optional eingestuft werden. Das Resultat wäre die einmalige Erstellung der SI-Komponenten und Architekturmuster ohne jegliche Pflege und Weiterentwicklung.

Das Mittel zur Selbstverpflichtung der Wartung und Pflege des SI-Katalogs innerhalb des betrachteten Unternehmens ist eine sog. *Richtlinie*. Eine durch den Vorstand verabschiedete Richtlinie schreibt den betreffenden Mitarbeitern vor, personelle, materielle und monetäre Ressourcen zur Umsetzung der Richtlinie bereitzustellen und die Umsetzung derselbigen zu kontrollieren. Im Detail werden der Geltungsbereich sowie die betreffenden Personen genannt, für die diese Richtlinie verbindlich sind. Der Richtlinie ist ein innerbetrieblicher Abstimmungsprozess zur Konsensherstellung zugeordnet. Der Abstimmungsprozess (Abbildung 10.3) beschreibt die Vorgehensweise und das Zusammenspiel der einzelnen Akteure und sieht neben einem Konsens auch eine Eskalation vor.

Im Einzelnen formuliert die Richtlinie:

- Die im SI-Katalog beschriebenen SI-Komponenten und SI-Architekturmuster werden nach Vorschlag der Architekten im Konsens mit dem Betreiber definiert.
- Die definierten Standards sind verbindlich in Projekten einzusetzen.
- Jegliche Abweichungen von diesen definierten Standards müssen eigens durch das jeweils zuständige Gremium (A-Board, Fachvorstand etc.) genehmigt werden.
- Der SI-Katalog ist zyklisch (= jährlich) auf seine Aktualität zu prüfen.
- Diese Richtlinie gilt für die gesamte Postbank Systems AG.

10.4.2 Praktische Umsetzung der Phase

Im Rahmen der Umsetzung des SI-Katalogs bei dem untersuchten Unternehmen wurde festgesetzt, dass die Pflege des SI-Katalogs erst nach Abschluss der Phase Anwendung des SI-Katalogs initiiert wird. Die zyklische Aktualisierung wurde auf ein Jahr festgelegt.
Auslöser für den Phaseneintritt können entweder der zyklische Prüfturnus, die Einführung neuer Infrastrukturkomponenten oder die Einführung neuer Architekturmuster sein. Da das Forschungsprojekt auf zwei Jahre begrenzt wurde, konnte lediglich die Phase der Einführung neuer Architekturmuster praktisch begleitet werden. Eine Beobachtung der beiden anderen Ereignisauslöser erfordert demnach ein weiteres Forschungsprojekt.

11 Bewertung des Konzepts

Zu Beginn dieser Arbeit wurde die Finanzdienstleistungsinformatik als ein Spezialisierungsgebiet der Wirtschaftsinformatik angenommen. Zu den Tätigkeitsfeldern der Wirtschaftsinformatik gehört ökonomisches Handeln.[511] Eine Bewertung des in dieser Arbeit vorgestellten Konzepts wird daher unter betriebswirtschaftlichen Gesichtspunkten herbeigeführt.

Die Bewertung an sich kann über zweierlei Wege erzielt werden: der allgemeine Weg führt über die Literatur indem die theoretischen Eckpfeiler der Arbeit (z. B. Plattform-Konzept, Architekturmanagement, Musterbildung etc.) unter ökonomischen Gesichtspunkten betrachtet werden. Hingegen führt der spezielle Weg über die Evaluierung des Konzepts im Rahmen der zwei Fallstudien.

11.1 Bewertung im Allgemeinen

Um eine Bewertung im Allgemeinen durchführen zu können müssen die Eckpfeiler des in der Arbeit entwickelten Modells hervorgehoben werden: Ziel dieser Arbeit war es, ein begründetes Gestaltungskonzept in Form eines Modells bereitzustellen, das die Komplexitätsbedingten Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik erfasst, um zu einer Lösung beizutragen. Hierzu wurden drei Problemdimensionen erfasst, die die Infrastrukturarchitektur charakterisieren:

- Der Komplexität der Infrastrukturarchitektur wird zunächst mit Hilfe einer Dekomposition der Infrastrukturkomponenten begegnet. Nach der Identifikation der einzelnen Komponenten und Komponentenvarianten wird eine strategische Selektion durchgeführt. Hierdurch werden die ausselektierten Komponentenvarianten für zukünftige Architekturentwürfe ausgeschlossen. Eine Reduktion der Komponentenzahl impliziert eine Reduktion der Komplexität. Auf die Dekomposition folgt eine erneute Komposition der selektierten SI-Komponenten.

[511] Vgl. Kapitel 1

- Ein Infrastrukturarchitekturentwurf kann entweder ein reiner Neuentwurf oder die Wiederverwendung bereits Entworfenens sein. Unterschied zwischen beiden Prozessen ist, dass der Lösungsraum im Fall des Neuentwurfs offen ist. Die Wiederverwendung bereits entworfener Architekturen ist eine Konfiguration dieser Architekturen in dem durch die Konfigurationsmöglichkeiten bestimmten Variantenraum.[512] Die Wissenskomplexität und das Wissensvolumen werden demnach durch die Einschränkung des Variantenraums reduziert. Vergleichbar mit den in der Industrie eingesetzten Plattformen reduzieren die Architekturmuster den offenen Lösungsraum auf einen eingeschränkten Konfigurationsraum.
- Die dritte Problemdimension mit der sich diese Arbeit auseinandersetzt, ist die Finanzdienstleistungsinformatik, die, bedingt durch ihre historische Entwicklung, einer der Komplexitätstreiber ist. Zudem stellt die Finanzdienstleistungsinformatik eigene qualitative, teils gesetzlich vorgeschriebene Ansprüche an die Informationssysteme. Diese qualitativen Eigenschaften vergrößern den Variantenraum und steigern die Komplexität konfigurierbarer Architekturmuster. Der erhöhte Wettbewerb unter den Finanzdienstleistern erhöht den Druck ökonomisch zu Handeln. Je geringer die Ausgaben, desto höher die Einnahmen. Daher wird der Finanzdienstleistungsinformatik abverlangt, die Kosten für die Bereitstellung der informationstechnischen Dienste zu senken.

11.1.1 Wertbeitrag des Architekturmanagements

"Cost-Justify" Architecture

Für ein Modell, das ein Gestaltungskonzept für die Praxis der Finanzdienstleistungsinformatik darstellt und den Kostendruck als Modellgestaltende Problemdimension erklärt, muss ein öknonomischer Nutzen nachgewiesen werden. Unter rein betriebswirtschaftlichen Gesichtspunkten stellt sich die Frage nach der Rentabilität des Modells.
Konkreter formuliert ZACHMAN diese Rentabilitätsfrage in Bezug auf das der Infrastrukturarchitektur übergeordnete Thema Enterprise Architek-

[512] Vgl. Bongulielmi (2002), S. 48ff

tur: „Wherever I go and talk about Enterprise Architecture, the most frequently asked question I get is, 'Well, how do you cost-justify architecture?' [...] The common perception is, it takes too long and it costs too much to do Architecture and […] you have to do too much work before you can deliver some end results."[513]

Ähnlich stellen auch SCHWINN und WINTER fest, dass die Rentabilität jeglicher Architekturmanagementaktiviäten kurzfristig nur Aufwände produziert. Daher empfehlen sie nicht die Rentabilität in Einzelprojekten zu suchen, sondern den ökonomischen Nutzen der durchgeführten Maßnahmen Projektübergreifend über einen längeren Zeitraum zu ermitteln.[514]

Betrachtet man lediglich den Aufwand, der betrieben wurde, um die Infrastrukturkomponenten im Rahmen der Fallstudie 1 in Gänze zu identifizieren, und addiert man den Aufwand hinzu, der notwendig war um die Menge der Infrastrukturkomponenten auf SI-Komponenten zu reduzieren und letztendlich zu Architekturmustern zu komponieren, so liegt dieser Betrachtungszeitraum weit über zwei Jahren. Es bleibt zu vermuten, dass mindestens 4 bis 8 Jahre kontinuierlicher Modellevaluierung notwendig wären, um eine exakte Aussage bzgl. der Rentabilität des Modells treffen zu können. ZACHMAN empfiehlt daher Architektur im Allgemeinen als einen immateriellen Vermögensgegenstand aufzufassen: „You invest in assets in order to enable you to do something you otherwise are unable to do."[515] Die Rentabilität der Architektur als immateriellem Vermögensgegenstand hängt von der Häufigkeit ihres Einsatzes ab. ZACHMAN sieht die Architektur als Investition eines Unternehmens in die Zukunft, die die folgenden vier Dinge ermöglicht:[516]

- *Alignment*: Die Informationssysteme sind an der Unternehmensstrategie ausgerichtet.[517]
- *Integration*: Systeme können über Betriebssystemgrenzen hinweg kommunizieren und Daten sind überall verfügbar.

[513] Zachman (2001)
[514] Vgl. Schwinn/Winter (2005)
[515] Zachman (2001)
[516] Vgl. Zachman (2001)
[517] Vgl. Krcmar (2005), S. 316ff

- *Change*: Sofern Änderungen an den bestehenden Informationssystemen vorgenommen werden müssen, kann ein Architekt die Architekturpläne analysieren und auf Basis dieser Erkenntnisse die zu verändernde Architektur vorschlagen. Fehlen diese Architekturpläne bleiben dem Architekten drei Möglichkeiten: Änderungen mittels „Trial and Error“ Prinzip, Reverse Engineering oder kompletter Neubau des Informationssystems.
- *Reduced „Time to Market”*: Architektur hilft die Zeit zu verkürzen, die zwischen Auftragseingang und Fertigstellung des Architekturentwurfes vergeht.

Architekturmanagement basiert in erster Linie auf dokumentiertem Wissen bezüglich der ganzheitlichen Architektur der Informationssysteme. Zudem leitstet Architekturmanagement nach ZACHMAN einen Beitrag zum Total Quality Management.[518] Es bleibt zu vermuten, dass die Messung des monetären Wertbeitrags, den das Architekturmanagement im Allgemeinen leistet, einen ähnlich langen Zeitraum erfordert wie die Disziplinen Wissensmanagement und TQM.

Der Wertbeitrag des Architekturmanagements ist demnach eher strategisch anzusehen: in Anlehnung an HEINRICH schreiben HAFNER ET AL. dem Architekturmanagement eine theoretische und monetär nicht nachzuweisende Unterstützungsfunktion zur wirtschaftlichen Erreichung strategischer Unternehmensziele zu.[519]

Ein weiterer Hinweis auf die strategische Wirkung des Architekturmanagements als Teil einer IT-Strategie ist der Umstand, dass die konkrete Ausgestaltung der IT von Analysten zu den strategischen Wettbewerbsfaktoren eines Unternehmens gezählt wird: „Für Moody’s schlägt sich die IT-Strategie v. a. in der Bewertung von Unternehmen nieder, zu deren Kernkompetenzen sie zählt“[520].

[518] Vgl. Zachman (2001)
[519] Vgl. Hafner et al. (2004); vgl. Heinrich (2002)
[520] o. V. (2001), entnommen aus Scheller (2006), S. 8; in Bezug auf strategische Relevanz siehe auch Rohloff (2005)

11.1.2 Wertbeitrag aus Investitionssicht

Die Umsetzung des SI-Katalogkonzepts erfordert zunächst einen Aufwand. Obwohl das SI-Katalogkonzepts rein methodischer Natur ist, werden Mitarbeiter des umsetzenden Unternehmens benötigt, die die innerhalb des Konzepts definierten Rollen ausfüllen. Unternehmensinterne Leistungsverrechnungsmodelle implizieren einen monetären Aufwand. Daher stellt sich für Entscheider im Fall des SI-Katalogkonzepts wie bei jedem anderen Projekt die Frage, welche Kosten es verursacht und welchen Ertrag es einbringt. Ergo ist die Entscheidung zur Umsetzung des SI-Katalogkonzepts eine Investitionsentscheidung mit hoher Unsicherheit.

Call-Option nach Black/Scholes

Eine Reihe von Autoren setzt sich mit dem Thema der Investitionsrechnung im Allgemeinen auseinander.[521] Im Speziellen vergleicht HOFER den die Investition in eine Plattform mit dem Erwerb einer Call-Option nach Black-Scholes:[522] Eine Option bedeutet das Recht und nicht die Verpflichtung des Optionsscheininhabers einen Kauf oder Verkauf zukünftig zu tätigen. HOFER folgert aus der Bewertung von Optionen für das Plattformkonzept, dass das Wahlrecht des Optionsscheininhabers zur Ausübung der Option mit zunehmender Unsicherheit ansteigt. Somit kann sich der Einsatz des Plattformkonzepts in einer von unternehmerischer Unsicherheit geprägten Situation lohnen.

Die Bewertung mit Hilfe des Black-Scholes-Modells ermöglicht zwei verschiedene Bewertungsarten des Plattformkonzepts. Zum einen kann der Investition in eine Plattform ein Wert zugeordnet werden. Zum anderen kann der maximal in eine Plattform zu investierende Betrag ermittelt werden. Durch eine aktive Veränderung der Variablen ergeben sich nach HOFER folgende drei Strategien, die zu einer Erhöhung des Werts einer Plattform führen:[523]

> Die Erhöhung des Projektwerts bedeutet eine Erhöhung des Erlöses und ist wichtiger als die Senkung der Investitionskosten.

[521] u. a. Strunz (1998), Kruschwitz (2005), Wöhe (1996)
[522] Vgl. Hofer (2001), S. 171ff bzw. Black/Scholes (1973)
[523] Vgl. Hofer (2001), S. 176

- Der Optionswert kann durch Unsicherheit erhöht werden indem die Unsicherheit durch riskantere Plattformenentwicklungen vergrößert wird. Eine Wiederverwendung einer Plattform in unterschiedlichen Segmenten erhöht gleichermaßen die Chancen wie die Risiken. Die negativen Auswirkungen sind jedoch begrenzt.
- Wird die Laufzeit der Option verlängert oder der Wertverlust gesenkt, steigt der Wert der Option.

Die Amortisation einer Plattform kann sowohl über eine Kostensenkung als auch über eine Umsatzerhöhung vorangetrieben werden. Je günstiger eine Plattform (incl. der in ihr eingesetzten Elemente) ist, desto eher rentiert sie sich. Je häufiger eine Plattform wiederverwendet wird, desto eher rentiert sich diese Plattform. BLACK und SCHOLES beschreiben diese Mechanismen mit Hilfe der nachfolgend abgedruckten Formel:[524]

$$F = N\{d_1\}Se^{-\delta T} - N\{d_2\}Xe^{-rT}$$

mit $d_1 = \frac{\ln(\frac{S}{X}) + T(r - \delta + 0{,}5\sigma^2)}{\sigma\sqrt{T}}$, $d_2 = d_1 - \sigma\sqrt{T}$

Seien F der Optionswert der Call-Option, S der Projektwert, X die Investitionskosten, T die Laufzeit, σ die Unsicherheit der Option, r der risikolose Zinssatz sowie δ die Dividende, so können die Entwicklung und Pflege einer Plattform entsprechend in das Modell eingepflegt werden.[525] Je größer die Dividende im Verlauf der Zeit und je länger der Verlauf der Zeit, desto geringer wird der Wert der Option. Die Dividende erfasst sowohl alle anfallenden Kosten für die Pflege der Plattform als auch sämtliche Opportunitätskosten, die mit dieser Plattform verbunden sind. Ein Beispiel für derartige Opportunitätskosten ist die Fokussierung auf eine Plattform und der daraus resultierende eingeschränkte Handlungsspielraum in Bezug auf die Abdeckung spezifischer Anforderungen durch die Plattform selber. Angenommen das Architekturteam entwirft ein Architekturmuster für J2EE Applikationsserver, das lediglich eine Applikationsservervariante (z. B. JBoss) mit einer Betriebssystemvariante (z. B. Linux) komponiert. In diesem Fall entstünden Opportunitätskosten, wenn eine konkrete Anforderung eine andere Applikations-

[524] Vgl. Benaroch/Kaufmann (1999)
[525] Vgl. Hofer (2001), S. 177

servariante (z. B. Bea WebLogic) oder eine andere Betriebssystemvariante (z. B. Solaris) erfordert.

Die Anwendung des Black-Scholes-Modells zur Bewertung von IT-Projekten findet et al. bei BENAROCH und KAUFMANN, SULLIVAN ET AL. sowie KAZMAN ET AL. statt.[526] Der Vergleich des Plattformkonzepts mit dem Erwerb einer Call-Option ist auf das SI-Konzept anwendbar, da sämtliche Prozesse innerhalb des Konzepts Aufwände verursachen: sowohl einen initialen Aufwand als auch einen laufenden Aufwand für die Pflege des Katalogs. Der Finanzdienstleister investiert mit dem Einsatz des SI-Katalogkonzepts in ein Projekt, von dem er sich im Verlauf der Zeit einen Nutzen erhofft. STAHLKNECHT und HASENKAMP unterscheiden zwischen quantifizierbarem Nutzen und nicht monetär quantifizierbarem Nutzen.[527] Quantifizierbarer Nutzen lässt sich nicht immer monetär bewerten. Nicht quantifizierbarer Nutzen ist qualitativer Natur. Beispiele sind höhere Termintreue, verbesserte Qualität, kürzere Entwicklungszeiten oder ein verbessertes Wissensmanagement.

Dennoch muss an dieser Stelle hervorgehoben werden, dass die bei HOFER oder auch PIECH erwähnten Plattformen in wesentlich größerer Stückzahl produziert werden als dies etwa im Fall der Architekturmuster bei Finanzdienstleistern zutrifft. Die Folge einer geringeren Stückzahl ist, dass sich der Rentabilitätszeitpunkt nach hinten verschiebt. Je seltener eine Plattform bzw. ein Architekturmuster wiederverwendet wird, desto höher ist die Wahrscheinlichkeit, dass die Projektkosten und die Dividende die Einsparungen im Lauf der Zeit überwiegen.

Dass Plattformen aber auch in kleinerer Stückzahl angewendet werden können, zeigen FRANKE ET AL. für die Kleinserienfertigung im sog. *Spezialmaschinenbau*. Der Nachweis eines quantifizierbaren Nutzens der Plattformen bleibt bei FRANKE ET AL. aus. Allerdings verweisen FRANKE ET AL. auf Hersteller, die das Plattformkonzept in der Kleinserienfertigung erfolgreich einsetzen. Der beschriebene Nutzen ist nicht quantifizierbarer Natur. [528]

---

[526] Vgl. Benaroch/Kaufmann (1999), Sullivan et al. (1999) sowie Kazman et al. (2001)
[527] Vgl. Stahlknecht/Hasenkamp (2005), S. 251ff
[528] Vgl. Franke et al. (2002)

Allerdings ist fraglich, ob eine derartig mathematischkomplexe Bewertungsmethodik den Entscheidern innerhalb eines Finanzdienstleisters gerecht wird. Wahrscheinlicher ist das Überwiegen des Pragma-tismus gegenüber theoretischen Werten. Daher wird im nachfolgenden Abschnitt die Bewertung des SI-Katalogkonzepts unter praktischen Gesichtspunkten durchgeführt.

11.2 Bewertung im Speziellen

Ziel dieses Abschnitts soll die Bewertung des SI-Katalogkonzepts im Speziellen sein. Während bisher lediglich ein theoretischer Wertbeitrag auf Basis der Literatur ermittelt werden konnte, wird in diesem Abschnitt der Wertbeitrag anhand der beiden Fallstudien ermittelt.
Die beiden Fallstudien sind unterschiedlicher Natur. Während bei der einen Fallstudie das SI-Katalogkonzept entwickelt und angewendet werden konnte, reichten die Ressourcen bei der zweiten Fallstudie lediglich für eine theoretische Auseinandersetzung mit dem Modell. Daher werden die beiden Fallstudien im Folgenden getrennt ausgewertet.

11.2.1 Fallbeispiel 1

Zur Erhebung der nachfolgend beschriebenen Érgebnisse wurden Einzelinterviews durchgeführt.

11.2.1.1 Bewertung

11.2.1.1.1 Kritikpunkte

Die Auswertung der Interviewergebnisse ergab, dass sich die geäußerte Kritik im Wesentlichen auf das Thema UML konzentriert. Im Einzelnen konnten nachfolgend aufgelistete Punkte identifiziert werden:

Modellkomplexität

Die Infrastrukturarchitektur ist sehr komplex. Das SI-Katalogkonzept erfasst nicht nur die Eigenschaften und Assoziationen der Soft- und Hardwarekomponenten in einem Meta-Modell, es enthält auch ein Vorgehensmodell, das die Handlungsschritte von der Erfassung der SI-Komponenten bis hin zur Pflege der SI-Architekturmuster beschreibt.

Hierdurch wird ein komplexes System mit Hilfe eines anderen komplexen Systems beschrieben. Ziel des SI-Katalogs ist es, die bestehende Komplexität der Infrastrukturarchitektur auf Dauer zu reduzieren. Da die Anwendung des SI-Katalogkonzepts voraussetzt, zum einen die Modellelemente zu kennen und zum anderen das Modell in Gänze zu verstehen, verlangt dieser Umstand ein enormes Zusatzwissen von einem Architekten ab. Ebenfalls kann davon ausgegangen werden, dass einzelne Modellelemente der Meta-Ebene im Laufe der Zeit an die jeweils vorherrschenden Organisations- und Technologiebedingten Gegebenheiten angepasst werden müssen. Dies macht die Rolle des Meta-Modellinhabers erforderlich. Da das Modell auf die Infrastrukturarchitektur begrenzt ist, muss der Meta-Modellinhaber in persona ein Architekt sein, dessen Aufgabengebiet um die Pflege des Meta-Modells erweitert wird. Hierzu zählt vor allem auch eine temporäre Befreiung dieses Architekten von alltäglichen Architekturprojekten.

Gefahr der Pauschalisierung

Mehrere Umstände sprechen für eine Pauschalisierungsgefahr. Zum einen birgt der Einsatz der SI-Architekturmuster die Gefahr, dass die komplexen Eigenschaften jedes innerhalb des Meta-Modells als Obergruppe bezeichneten Themengebiets von denjenigen Personen, die sich nicht auf dieses Themengebiet spezialisiert haben, pauschalisiert werden. Die reine Orientierung an Leistungs- und Architekturattributen sowie spezifischen Einsatzgebieten und Regeln verhindert nicht, dass Laien ohne Konsultierung der Spezialisten Architekturen entwerfen, deren qualitative Ausprägung nicht der qualitativen Anforderung entspricht. Ein Grund hierfür ist die Schwierigkeit, exakte Qualitätsattribute zu formulieren. Der Algorithmus der Anwendung des SI-Katalogs erfordert, dass genügend Qualitätsattribute den Anforderungen gegenüberstehen. Die Qualitätsattribute müssen sämtliche typischen, architektonischen und leistungstechnischen Anforderungen eines neuen Architekturprojekts abdecken. Treffen (un)genau formulierte Anforderungen auf nicht exakt formulierte Qualitätsattribute, so führt dies leicht zu einem inadäquaten Architekturentwurf.

Eine exakte und vollständige Formulierung der Architekturattribute ist in den meisten Fällen sehr schwer bis unmöglich. Ein Beispiel hierfür sind die zur Verfügung stehenden Möglichkeiten der Leistungsattributaus-

prägung *Performance* im Fall der Obergruppe *Datenbanken*. Eine exakte Formulierung des Qualitätsattributs *Performance* wird primär dadurch erschwert, dass keine allgemein gültige und Herstellerübergreifend verwendete Definition dieses Attributs existiert. Prägend für die Qualitätsausprägung der Eigenschaft *Performance* ist beispielsweise die Art und Weise des SQL-Befehls. Je nach Abfragetyp (z. B. Natural Join, Inner Join etc.) kombiniert mit einer variiernden Datensatzgröße kann eine Datenbankvariante performanter sein als eine andere. Da jedoch nicht alle Einflussfaktoren der Qualitätsattribute im Detail bekannt sind und Variantenübergreifend variieren, sind Näherhungswerte unabdingbar. Das Meta-Modell bietet zwar die theoretische Möglichkeit jeden Einflussfaktor in Form eines Leistungs- oder Architekturattributs exakt zu formulieren, jedoch behindern die real-praktischen Umstände den Gebrauch dieser Möglichkeit. Daher ist es umso wichtiger, dass die Architekten untereinander kommunizieren.

Sofern die Architekten den SI-Katalog als rein Architekturinternes Beratungsinstrument einsetzen, relativiert sich diese Pauschalisierungsgefahr. Ein weiterer genannter Punkt ist die Sorge, dass die Existenz des SI-Katalogs den Eindruck beim höheren Management erwecken könnte, dass die Spezialisten ihr Wissen auf Dauer verewigt hätten und sich somit selbst ad absurdum geführt hätten.

Inadäquanz der UML

Zur Modellierung der Architekturmuster innerhalb des SI-Katalogs wurde ein hybrider Ansatz gewählt, der sowohl eine Matrizen-basierte als auch eine UML-basierte Dokumentation zulässt. Da UML als de facto Standard der Modellierung von IT-Architekturen identifiziert wurde, wurde der Fokus der Modellierung im Rahmen der Fallstudie auf diese Sprache gelegt. Erst bei initialen Modellierungsworkshops wurde festgestellt, dass im Schnitt 5 von 10 Architekten dieser Sprache nicht mächtig sind. Bei den Infrastrukturarchitekten war das Verhältnis noch höher (9 von 10).

Dem Objektorientierten Paradigma folgend erfordert die UML abstraktes Denken. HERZ ET AL. stellen fest, dass abstraktes Denken erlernt werden muss.[529] Möglicherweise ist dies der Grund für die anfängliche

[529] Vgl. Herz et al. (2001)

Skepsis und Ablehnung der Mehrzahl der Infrastrukturarchitekten gegenüber der UML. Auslöser ist der über Jahre gewohnte Umgang mit „Freistil“-Werkzeugen[530] wie Visio oder PowerPoint, die dem Architekten beispielsweise erlauben, ClipArts von Servern, Netzwerkroutern, Datenbanken etc. in die Grafiken mit einzubinden. Die Diagramme der UML erfordern ein abstraktes Denken und Vorstellungsvermögen, das eine gewisse Zeit benötigt um akzeptiert zu werden. Hinzu kommt, dass die an sich bereits abstrakte UML durch das Meta-Modell des SI-Katalogs mit jeder Ebene stärker abstrahiert wird. Somit ergibt sich eine Meta-Meta-Ebene, die für UML-Neulinge die Hemmnisschwelle enorm erhöht.

Die bei KELLER und WENDT beschriebene Rolle des Architekten, bei der selbiger als Mediator zwischen Kunde, Management und Entwicklung agiert,[531] wird bei dem untersuchten Unternehmen in der Art gelebt, dass der Architekt Mediator zwischen Kunde, Management, Entwicklung und Betrieb ist. Der typische Arbeitsablauf eines Architekten sieht vor, dass ein Architekturentwurf letztendlich vom Vorstand freigegeben werden muss, nachdem der Entwurf mehrere Gremien durchlaufen hat. Die Freigabe erfolgt im untersuchten Unternehmen durch Präsentation und Sichtung des Architekturentwurfs. Die Präsentation erster Architekturentwürfe, die mit der UML beschrieben wurden, stellte sich jedoch als vorstandsuntauglich heraus, da schlichtweg die gewohnten ClipArts von Servern, Netzwerkroutern, Datenbanken etc. fehlten. Gleiches konnte auch bei Architekturdiskussionen mit Vertretern des Betriebs festgestellt werden. Hauptgrund ist der Umstand, dass diese Zielgruppe des Architekten dessen „Sprache“ nicht verstanden hat. Als Folge der Ablehnung musste der Architekturentwurf „Zielgruppengerecht“ gestaltet werden, wobei der Architekt zwei Versionen seines Architekturentwurfs erstellte: eine interne, UML-basierte Variante, die innerhalb des Architekturteams verwendet wurde, und eine externe, PowerPoint- bzw. Visio-basierte Variante, die als Diskussionsgrundlage für Nicht-Architekten diente.

---

[530] Anmerkung: Dieser Begriff erscheint dem Autor nicht wissenschaftlich genug. Jedoch drückt er eine gewisse Unbefangenheit aus, mit der Visio- oder PowerPoint-Nutzer Notationen nach Belieben variieren können.

[531] Vgl. Keller/Wendt (2003)

Hoher Pflegeaufwand

Der Umstand, dass der Architekt zwei Versionen eines Architekturentwurfs erstellen muss, kombiniert mit dem Aufwand, der für die Pflege der SI-Architekturmuster notwendig ist, bedeutet einen hohen Zeitaufwand für den Architekten. Einige Interviewpartner äußerten daher, dass die Erstellung, Anwendung und Pflege des SI-Katalogs das Alltagsgeschäft stark bürokratisiert. Zudem werden UML-Diagramme auf Grund ihrer Notation sehr schnell sehr komplex. Wird ein UML-Diagramm zu groß, kann es in der Regel auf mehrere Einzeldiagramme aufgeteilt werden. Letzteres erschwert jedoch den Überblick. Auch dies treibt den Aufwand in die Höhe.

Werden die Matrizen wie im Fallbeispiel parallel zu den UML-Diagrammen mitgeführt, erleichtern diese den Gesamtüberblick in Bezug auf die Verflechtungen einzelner Architekturmuster. Da jedoch kein Werkzeug zur Verfügung stand, welches die UML-Diagramme und Matrizen automatisch konsistent gehalten hat, mussten die Matrizen mit Excel nachgepflegt werden. Auch dies trieb den Pflegeaufwand für den SI-Katalog enorm in die Höhe.

Meta-Modell

Die bis zu diesem Punkt geäußerten Kritikpunkte bezogen sich in erster Linie auf die Anwendung und Pflege des SI-Katalogs. Im Folgenden wird nun die Kritik auf das Meta-Modell des SI-Katalogs konzentriert.

Das Meta-Modell schreibt vor, dass ein Architekturmuster zu genau einer Obergruppe gehört. Ferner wird vorgeschrieben, dass Architekturmuster einer Obergruppe mit gleichen Architekturattributausprägungen zu einem Architekturmuster zusammengeführt werden müssen. Die negativen Auswirkungen dieser Regel werden am Beispiel des Designs des Architekturmusters „Server (No D/R) Pattern“ deutlich, welches eine nicht Desaster-Recovery fähige Clusterkomposition beschreibt (Abbildung 11.1): Das Architekturmuster „OS (Simple) Pattern“ generalisiert die SI-Komponenten *Linux*, *Solaris*, *AIX* und *Windows*. Da sich die Betriebssysteme rein architektonisch nicht unterscheiden (d. h. sie haben gleiche Architekturattributausprägungen) sondern lediglich leistungstechnische Unterscheidungsmerkmale aufweisen, müssen sie auf Grund der Prämisse in dem Architekturmuster „OS (Simple) Pattern“ generalisiert werden. Der abstrakte Typ *Server* ist eine Generalisierung der SI-Komponenten *Pentium kompatibel*, *SPARC* und *PSeries*. Er

kann immer nur mit genau einem Betriebssystemtyp verbunden werden. Da der Betriebssystemtyp auf Grund der o. g. Regel in einem eigenen Architekturmuster definiert wird, kann keine direkte Assoziation zwischen den beiden Typen hergestellt werden. Vielmehr muss die vom UML-Standard erfasste Object Constraint Language (OCL) für solche Fälle verwendet werden.[532] Die Folge ist eine weitere Komplexitätssteigerung der UML-Diagramme.

Eine andere Auswirkung der konsequenten Verbannung Obergruppenfremder SI-Komponenten aus den der jeweiligen Obergruppe zugeordneten Architekturmustern wird am Beispiel der Instanziierung der Architekturmuster deutlich: je größer die Anzahl der Obergruppen ist und je häufiger die Wiederverwendung von Architekturmustern in anderen Architekturmustern angewendet werden kann, desto komplexer ist das Instanz-Diagramm der Meta-Ebene M0 und desto höher ist der Aufwand eines Architekten bei der Instanziierung. Wird beispielsweise in einem Architekturentwurf, der mit Hilfe von Visio oder PowerPoint erstellt wird, eine Firewall als einfaches Feuer-Symbol dargestellt, erfordert die konsequente Verfolgung der UML je nach Architekturmuster die Assoziation zwischen einer Klasse vom Typ Firewall, einem Server und einem Betriebssystem.

[532] Anmerkung: Eine alternative Lösung wäre ein bewusstes Verzichten auf die Dekomposition der Obergruppen *Server* und *Betriebssysteme*.

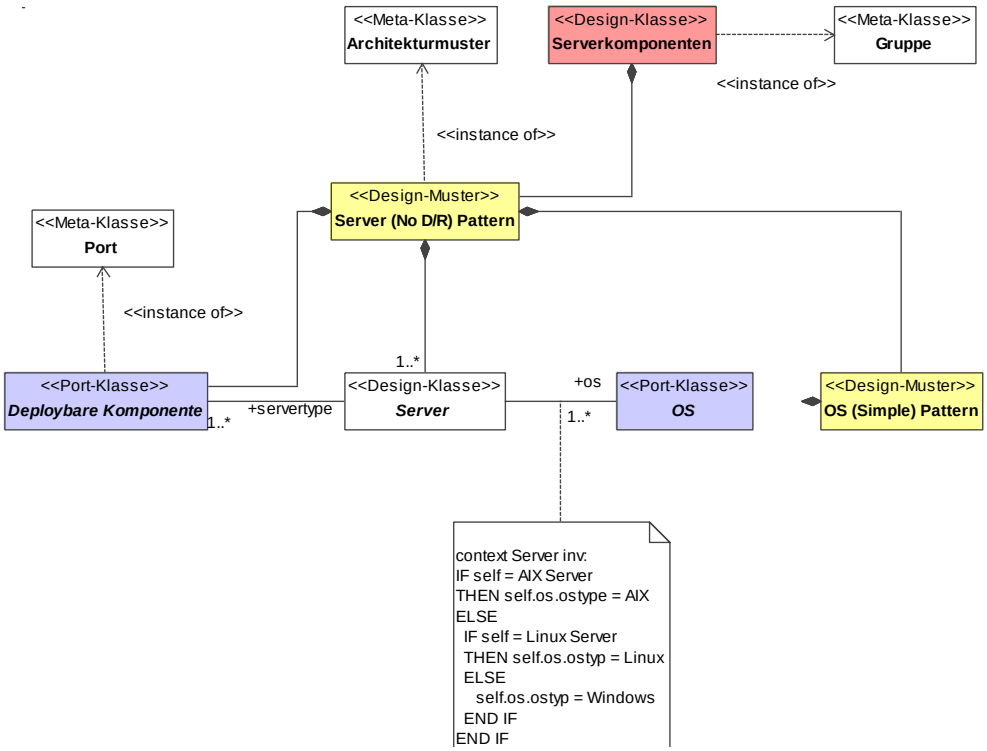


Abbildung 11.1: Das Architekturmuster „Server (No D/R) Pattern“. (Quelle: Eigene Darstellung)

11.2.1.1.2 Potentiale

Sensibilisierung

Zu dem nicht quantifizierbaren Nutzen des SI-Katalogs muss in erster Linie die Sensibilisierung derjenigen Personen für das Metier der Infrastrukturarchitektur genannt werden, die sich nicht täglich mit der Komplexität der Infrastruktur auseinandersetzen. Zu Beginn der Untersuchung des betrachteten Unternehmens musste sich das Ressort *Technologiemanagement* als Folge einer Reorganisation neu finden. Die Aufspaltung der Architekten in Domänen- und Infrastrukturarchitekten erfolgte auf Basis früherer Organisationsstrukturen. Die Infrastrukturarchitekten wurden größtenteils aus den Reihen des Ressorts *Betrieb* gestellt, wohingegen die Domänenarchitekten zum Teil von anderen Unternehmen (bspw. T-Systems) abgeworben wurden bzw. aus den Reihen der Softwareentwicklung entstammten. Die räumliche und organisatorische Trennung der beiden Gruppen kombiniert mit der Unzugänglichkeit einzelner Gruppenmitglieder führte zu einer Teilung der Architektur in zwei Lager: die „Anzugträger“ (Domänenarchitekten) und die „Blaumänner“ (Infrastrukturarchitekten). Während einzelne Infrastrukturarchitekten der Meinung waren, den Domänenarchitekten fehle es an nötigem Sachverstand, wurde umgekehrt vereinzelt eine Klassifizie-

rung der Infrastrukturarchitekten als Architekten zweiter Klasse festgestellt.

Die Einführung des SI-Katalogs eröffnete eine initiierende Kommunikation zwischen beiden Gruppen: auf der einen Seite konnten die Domänenarchitekten mit Hilfe der Architekturmuster und der Leistungsklassen die Architekturentwürfe nachvollziehen. Auf der anderen Seite gelang es den Infrastrukturarchitekten ein Verständnis für die abstraktere Denkweise der Domänenarchitekten sowie deren Reduktion komplexer Sachverhalte auf einfachere Leistungs- und Architekturattribute zu entwickeln. Die ineinander übergreifenden Architekturmuster bewirken sowohl eine Zusammenarbeit und Abstimmung der Infrastrukturarchitekten untereinander als auch eine Kommunikation aller Architekten. Die scheinbare Reduktion der Komplexität der Infrastrukturarchitektur auf Architekturmuster und Qualitätsattribute bietet einen Einstieg für fremde Wissendomänen.

IST/SOLL-Analyse

Grundlage der entworfenen Architekturmuster sind sowohl existierende als auch neue Architekturen. Jedes Architekturmuster ist eine Komposition unterschiedlicher SI-Komponenten, die sich durch unterschiedliche Leistungsattributausprägungen sowie spezifische Einsatzgebiete voneinander unterscheiden. Die individuelle Komposition der SI-Komponenten zu einem Architekturmuster wird im SI-Katalog qualitativ durch die Architekturattribute beschrieben.

Durch die Zuordnung der entworfen Architekturmuster zu den bestehenden bzw. in Planung befindlichen Systemen wird ein IST/SOLL-Vergleich ermöglicht, der das Potential der Abänderung der IST-Komposition nach qualitativen und wirtschaftlichen Gesichtspunkten hervorhebt. Insgesamt konnten bei einem Vergleich vier unterschiedliche Szenarien identifiziert werden (Tabelle 11.1):

- Informationssystem *A*: Sind die beiden Varianten *S'* (z. B. *Oracle 9i Single Instance*) und *S''* (z. B. *Microsoft SQL Server 2000*) als Generalisierung *S* mit einem Architekturmuster assoziiert, kann ein Wechsel von *S'* zu *S''* die laufenden Kosten langfristig senken, sofern die TCO für *S''* günstiger sind als die TCO für *S'*. Weisen *S'* und *S''* unterschiedliche Leistungsausprägungen auf, müssen diese bei einer

Verschlechterung der Qualität im Fall eines Wechsels berücksichtigt werden.

> Informationssystem *B*: Die Qualität des Systems *B* ist höher als nötig. Eine mögliche Ursache ist, dass qualitative Anforderungen an ein Informationssystem zu Beginn eines Projektes unter hoher Unsicherheit überschätzt wurden. Der Wechsel von der jetzigen Architektur auf eine andere kann die TCO des Systems drastisch senken. Gleichzeitig ist ein Qualitätsverlust obligatorisch.

> Informationssystem *C*: Hierbei handelt es sich um den umgekehrten Fall. Die Qualität des Systems *C* ist geringer als nötig. Grund für den IST/SOLL-Unterschied kann eine gesetzliche Änderung sein (z. B. eine nicht vorhandene Desaster-Recovery Fähigkeit). Durch eine zunehmende Qualität können die Kosten drastisch steigen.

> Informationssystem *D*: Das System ist zum jetzigen Zeitpunkt problemadäquat.

Als Notation hat sich im Rahmen der Fallstudie bewährt: $\rightarrow$ (neutral), $\downarrow$ (Reduktion), $\downarrow\downarrow$ (starke Reduktion), $\uparrow$ (Erhöhung), $\uparrow\uparrow$ (starke Erhöhung), Q (Qualität), K (Kosten).

Obergruppe: *SAP-Systeme*

	IST-Architektur			**SOLL-Architektur**			**Potential**	
System	AM_1	AM_2	…	AM_1	AM_2	…	**Q**	**K**
A	•			•			$\rightarrow$	$\downarrow$
B		•		•			$\downarrow\downarrow$	$\downarrow\downarrow$
C	•				•		$\uparrow$	$\uparrow\uparrow$
D	•			•				
…								

Tabelle *11.1*: Qualitative Betrachtung bestehender Systeme.

Kosten

Informationssysteme verursachen Kosten. Beispiele für Kostentreiber sind die Betriebskosten (Strom, Wartung, Personal etc.) bzw. Softwarelizenzen. Ein Architekturmuster ist eine Komposition einzelner Soft- und Hardwarekomponenten. Jede einzelne Infrastrukturkomponente verursacht individuelle Kosten.

Im Fall des betrachteten Unternehmens kann der Betrieb jeder einzelnen Infrastrukturkomponente sog. *Betriebskosten* zuweisen. Diese Be-

triebskosten enthalten diejenigen Kosten, die der Betrieb der Bank für die Zurverfügungstellung der Komponente jährlich in Rechnung stellt. Die Betriebskosten werden jährlich überarbeitet und je nach Obergruppe der einzelnen Infrastrukturkomponente von einem anderen Plattformverantwortlichen berechnet. Durch die Zuweisung dieser Betriebskosten zu den einzelnen Infrastrukturkomponenten wird einem Architekten ermöglicht, die ungefähren Kosten seines Architekturentwurfs zu ermitteln, ohne jede einzelne Infrastrukturkomponente bei dem zuständigen Plattformverantwortlichen des Betriebs kostentechnisch zu erfragen.

Die Kosten für den Betrieb einer einzelnen Infrastrukturkomponente beinhalten neben den laufenden Kosten auch die einmaligen Kosten, die beim Kauf der Komponente anfallen. Je größer die bei einem Hersteller abgenommene Menge ist, desto größer ist der Rabatt, den ein Einkäufer aushandeln kann. Je häufiger eine Infrastrukturkomponente in einem Architekturentwurf verwendet wird, desto größer ist die Wahrscheinlichkeit, dass ein Einkäufer im Zuge einer neuen Bestellung den Preis für diese Infrastrukturkomponente senken kann.

Eine Infrastrukturkomponente eines Herstellers wird in der Regel von Konkurrenten dieses Herstellers angeboten. Die Wahlmöglichkeit eines Architekten innerhalb der Architekturentwürfe kann den Einkauf insofern unterstützen, dass dieser in den Verhandlungen mit dem präferierten Hersteller die Option des Wechsels zu dessen Konkurrenten im Sinne einer Preissenkung zu seinen Gunsten nutzen kann.

Der Einkäufer ist eher betriebswirtschaftlich orientiert als technisch. Daher benötigt er die Unterstützung technisch versierterer Kollegen. Um sich von der Konkurrenz abzuheben versuchen die Hersteller die individuellen Funktionalitäten des eigenen Produkts hervorzuheben. Hierbei verwenden sie technische Spezialbegriffe, die dem Einkäufer meist unbekannt sind.

Innerhalb des SI-Katalogs werden die einzelnen SI-Komponenten zunächst Obergruppen zugeordnet, um anschließend in Leistungsklassen eingeteilt zu werden. Tritt ein Einkäufer nun beispielsweise in die Verhandlungen mit einem Hersteller für eine Storagekomponente, kann diese Komponente (sofern noch nicht vom SI-Katalog erfasst) von ei-

nem Architekten mit Hilfe der Leistungsattribute klassifiziert werden. Die Zuordnung dieser Komponente zu einer Leistungsklasse (z. B. *Mid Range*) ist eine Reduktion der Wissenskomplexität in Bezug auf die betrachtete Komponete auf eine Leistungsklasse. Eine solche Reduktion ermöglicht dem Einkäufer ein Gegenangebot eines Konkurrenten innerhalb dieser Leistungsklasse einzuholen. Das Resultat ist die Stärkung des Einkäufers bei Preisverhandlungen.

Neben diesen Verhandlungsrelevanten Vorteilen begünstigt die Zuweisung der Kosten zu den einzelnen SI-Komponenten den direkten IST/SOLL-Vergleich. Wird beispielsweise in einer Architektur eine Infrastrukturkomponete vom Typ *S'* (z. B. *Oracle 9i Single Instance*) eingesetzt, und kann diese Komponente durch eine Variante vom Typ *S''* (z. B. *Microsoft SQL Server 2000*) ersetzt werden, trägt ein Vergleich der Kosten, die für den Betrieb der beiden Komponententypen anfallen, zur Entscheidungsfindung bei.

Anpassung an die Clientel

Die Kommunikation und Abstimmung der Architekten untereinander bewirkt ein geschlosseneres Bild, das die Architektur ihrer Clientel gegenüber präsentiert. Als besonders wichtig hat sich die Beratung des Managements herausgestellt. Die von der Bank in Auftrag gegebenen Projekte müssen mit dem auf ein Maximum begrenztes Budget der Finanzdienstleistungsinformatik umgesetzt werden. Aus diesem Grund müssen alle Architekturentwürfe vom Management innerhalb eines sog. *Architekturboards* diskutiert und abgestimmt werden.[533]

Primäres Entscheidungskriterium des Managements sind die mit den Architekturentwürfen verbundenen Kosten. Die Zuweisung der Kosten zu den einzelnen SI-Komponenten ermöglicht zum einen die Abschätzung der Investitionssumme. Kombiniert mit der Reduktion der Beschreibung der einzelnen Architekturkomponenten auf ihre Leistungsattribute bzw. Leistungsklassen werden selbst komplexe Architekturentwürfe in einer für den Kunden nachvollziehbaren Art vorgestellt.

Die Wiederverwendung der nicht-UML Grafiken, die jedem Architekturmuster zugeordnet wurden, unterstützt diesen Effekt. Ein Beispiel für

[533] In Anlehnung an Niemann (2005), S. 171ff, kommt Architekturboards zudem die Aufgabe zu, die vorgestellten Architekturentwürfe auf ihre Eignung für die strategischen Ziele des Unterenhmens zu überprüfen.

die visuelle Unterstützung ist in Abbildung 11.2 dargestellt: Die Abbildung zeigt das Architekturmuster „DB (Skalierbar, D/R, HA) Pattern“. Mit Hilfe der Komponentenbauweise kann dem Management plastisch erläutert werden, weshalb ein Datenbankentwurf Kosten im 7-stelligen Euro Bereich verursacht. Der Entwurf hebt beispielsweise hervor, dass drei Storage Systeme benötigt werden. Werden auf Grund der Anforderungsspezifikation Komponenten der Leistungsklasse „Storage High End“ benötigt, muss der 3-fache Preis für ein Storage System angesetzt werden. Gleiches gilt für die Netzwerkübergänge zwischen den einzelnen Datenbank- und Storage-Komponenten.

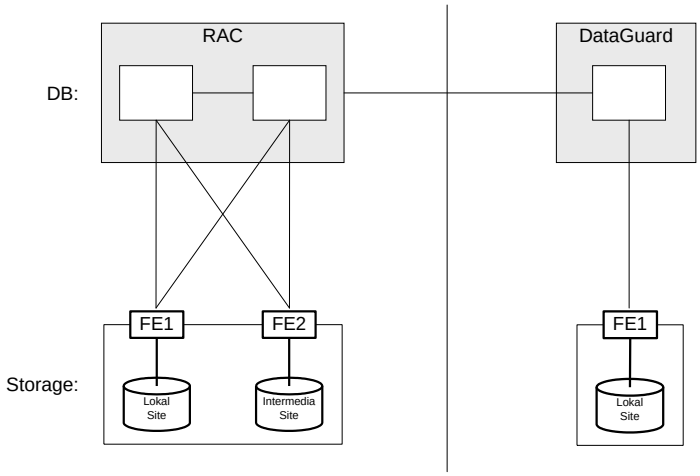


Abbildung 11.2: Nicht-UML Darstellung des Architekturmusters „DB (Skalierbar, D/R, HA) Pattern“. (Quelle: Eigene Darstellung)

11.2.1.2 Auswertung der spezifischen Befragung

Die Zielgerichtete Befragung der Interviewpartner ergab die im Folgenden abgedruckten und in Problemdimensionen getrennten Ergebnisse.

Problemdimension 1

Die in Tabelle 11.2 abgedruckten Ergebnisse der Befragung bzgl. der Komplexität der Infrastrukturarchitektur zeigen einen durchweg positiven Einfluss des SI-Katalogkonzepts auf diese Problemdimension. Auffällig ist die überwiegende Aussage, dass das SI-KK keine Auswirkung auf den Grad der Komplexität der Infrastrukturarchitektur hat. Diese Konzentration ist auf die Diskussion zurückzuführen, dass die konsequente Anwendung des Plattformkonzepts zum einen die Varianz der Infrastrukturkomponenten eingrenzt und sich somit senkend auf die Komplexität im Allgemeinen aus-wirkt, zum anderen aber die Komplexität des Meta-Modells bzw. die Komplexität der auf Basis des Meta-

Modells erstellten Modelle die Komplexität in anderer Form wieder erhöhen. Der Reduktion der Komplexität der realtechnischen[534] Infrastrukturarchitektur steht ergo eine Erhöhung der Komplexität der modelltechnischen Infrastruktur gegenüber. Diese Bewertung ist konsistent mit der Bewertung der Auswirkung des SI-KKs auf den Grad der Dynamik (Frage 2). Die konsequente fest-zyklische Anwendung des Plattformkonzepts verspricht die die Varianz im Rahmen neuer Projekte einzugrenzen, so dass der unkontrollierten Einführung neuer Infrastrukturkomponenten auf der einen Seite vorgebeugt wird, und dass auf der anderen Seite neu einzuführende Infrastrukturkomponenten in zyklisch festgelegten Abständen auf ihre Tauglichkeit hin überprüft werden.

#	**Frage**	**Ergebnis**
1	Welche Auswirkung hat das SI-KK auf den Grad der Komplexität der Infrastrukturarchitektur?	senkend: 6x ohne Auswirkung: 9x steigernd: 0x
2	Welche Auswirkung hat das SI-KK auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betreffen?	senkend: 8x ohne Auswirkung: 7x steigernd: 0x
3	Welche Auswirkung hat das SI-KK auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten?	senkend: 12x ohne Auswirkung: 3x steigernd: 0x
4	Inwieweit hilft das SI-KK dabei, die Komplexität der Infrastrukturarchitektur im Allgemeinen zu beherrschen?	überhaupt nicht: 0x wenig: 2x ausreichend: 12x sehr: 1x
5	Inwieweit hilft das SI-KK dabei, die Komplexität der Infrastrukturarchitektur speziell in Ihrem Unternehmen zu beherrschen?	überhaupt nicht: 0x wenig: 2x ausreichend: 12x sehr: 1x

Tabelle *11.2*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 1 (Fallstudie 1).

Problemdimension 2

Ebenfalls positiv wirkt sich die Anwendung des SI-KKs auf die Problemdimension 2 (Wissenskomplexität und das Wissensvolumen) aus: Durch die Zusammenfassung einzelner Infrastrukturkomponenten zu konfigurierbaren Architekturmustern werden die Zusammenhänge der Komponenten dokumentiert. Die qualitative Bewertung einzelner Kom-

[534] Anmerkung: Unter real-technischer Infrastrukturarchitektur werden die realen Infrastrukturkomponentenkompositionen als betriebsfähige Systeme zusammengefasst.

ponenten und ganzer Komponentenkompositionen abstrahiert die Detailkomplexität und vereinfacht die Entscheidungsfindung.

#	Frage	Ergebnis
1	Inwieweit hilft das SI-KK dabei, die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur zu beherrschen?	überhaupt nicht: 0x wenig: 0x ausreichend: 6x sehr: 9x
2	Inwieweit trägt das SI-KK dazu bei die sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgemein zugänglich zu dokumentieren?	überhaupt nicht: 0x wenig: 3x ausreichend: 6x sehr: 6x
3	Inwieweit trägt das SI-KK dazu bei die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass Sie in mehr als 60% aller Fälle nachvollziehen können weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	überhaupt nicht: 0x wenig: 7x ausreichend: 8x sehr: 0x
4	Inwieweit trägt das SI-KK dazu bei, dass eine standardisierte Notation zur Dokumentation der Infrastrukturarchitektur verwendet wird?	überhaupt nicht: 2x wenig: 4x ausreichend: 7x sehr: 2x
5	Inwieweit trägt das SI-KK dazu bei, die Wissenskomplexität bzw. das Wissensvolumens der Infrastrukturarchitektur Ihrem Unternehmen zu beherrschen?	überhaupt nicht: 0x wenig: 3x ausreichend: 8x sehr: 4x

Tabelle *11.3*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 2 (Fallstudie 1).

Dennoch wurde Skepsis gegenüber der generellen Beseitigung der Kopfmonopole geäußert, die das Wissen auf einzelne Personen konzentrieren. Ein Grund hierfür ist die Tatsache, dass die UML auf Grund ihrer Flexibilität ein- und dasselbe System unterschiedlich beschreiben lässt. So kann ein eher simples System durch viele Abstrahierungen und Generalisierungen durch ein komplexes Modell beschrieben werden, während gleichzeitig ein komplexes System durch Abstrahierungen und Generalisierungen durch ein unzureichend beschreibendes Modell repräsentiert werden kann. Diese Bedenken, die sich in der Ausprägung „wenig“ der Frage 2 widerspiegeln, beruhen im Wesentlichen auf der Annahme, dass einzelne Personen vorsätzlich die Modelle zu komplex bzw. unzureichend beschreiben, um einen gewissen Grad des Systemspezifischen Wissens nicht preiszugeben. Auch sehr allge-

mein formulierte Qualitätsattributausprägungen wie beispielsweise „schnell“ oder „langsam“ tragen hierzu bei.

Im Fall der Frage 3 wurde angemerkt, dass der SI-Katalog keinen Architekturauswahlprozess vorsieht, in den eine nachvollziehbar dokumentierte Entscheidungsfindung integriert ist. Der SI-Katalog ist so aufgebaut, dass er einem Architekten unterschiedliche Architekturmuster zur Selektion und anschließenden Konfiguration anbietet. Muss sich der Architekt beispielsweise zwischen den beiden Architekturmustern *P'* und *P''* entscheiden, so garantiert das SI-Katalogkonzept nicht, dass die Selektion nachhaltig dokumentiert wird.

Die eher negative Bewertung des SI-KKs im Rahmen der Frage 4 beruht auf der bereits hervorgehobenen Tatsache, dass die mit der UML dokumentierten Architekturmuster von der Clientel eines Architekten in der Regel nicht verstanden werden. Um Kundenorientierter zu agieren ist der Architekt daher gezwungen, eine Doppeldokumentation in Form eines exakten UML-Modells und einer weniger exakten PowerPoint- bzw. Visio-Grafik durchzuführen. Der in der Praxis herrschende Zeitdruck kann dazu führen, dass der Fokus eines Architekten auf die weniger exakte PowerPoint- bzw. Viso-Grafik gelegt wird, und dass das entsprechend exaktere UML-Modell bei hoher Unwahrscheinlichkeit „bei Gelegenheit“ nachdokumentiert wird.

#	**Frage**	**Ergebnis**
1	Inwieweit trägt das SI-KK dazu bei, sich der Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in Ihrem Unternehmen auswirkt, anzupassen?	überhaupt nicht: 6x wenig: 5x ausreichend: 5x sehr: 0x
2	Inwieweit trägt das SI-KK dazu bei, sich dem Kostendruck, der auf Ihr Unternehmen ausgeübt wird, anzupassen?	überhaupt nicht: 3x wenig: 5x ausreichend: 7x sehr: 0x

Tabelle *11.4*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 3 (Fallstudie 1).

Problemdimension 3

Eine eher negative Tendenz zeichnet sich im Fall der Problemdimension 3 ab: Der mit der Pflege des SI-Katalogs verbundene Aufwand lässt vermuten, dass er der Dynamik der Veränderungen im Infrastrukturbereich nicht standhält. Nur ein entsprechend langer Pflegezyklus kann

dem entgegenwirken, birgt aber die Gefahr, Innovation zu verhindern (Frage 1). Ebenfalls negativ wirkt sich dieser Verwaltungsaufwand auf den Kostendruck aus. Kosteneinsparpotentiale sind zwar durch das Plattformkonzept theoretisch gegeben, dennoch steigert der erforderliche Verwaltungsmehraufwand die Pflegekosten des Gesamtsystems.

11.2.2 Fallbeispiel 2

Im Rahmen des zweiten Fallbeispiels konnte das SI-KK auf Grund der zur Verfügung gestellten Ressourcen nur theoretisch evaluiert werden. Auch fällt die Befragungsbasis mit 5 Interviewpartnern viel kleiner als im ersten Fallbeispiel aus. Eine qualitativ hochwertige Evaluierung des SI-KK erfordert jedoch ein intensives Auseinandersetzen mit dem Konzept. Dennoch kann eine Befragung auf theoretischer Basis eine Tendenz bzgl. der Akzeptanz hervorbringen. Die Identifikation eines quantifizierbaren Nutzens wird durch die theoretische Befragung ausgeschlossen.

11.2.2.1 Bewertung

11.2.2.1.1 Kritikpunkte

Modellkomplexität

Die Interviewpartner stellten fest, dass das vorgestellte Konzept eine hohe Komplexität aufweist. Obwohl es sich hierbei vielmehr um eine Feststellung handelt, wird die Modellkomplexität in Anlehnung an die Evaluierung im Rahmen des ersten Fallbeispiels als Kritikpunkt aufgefasst. Um die Interna des Modells auf Ihre Korrektheit und praktische Anwendbarkeit überprüfen zu können, muss das Modell am praktischen Beispiel angewendet werden. Es ist zu bezweifeln, dass ein ganzheitliches Modell zur Überprüfung sämtlicher Facetten des Meta-Modells existiert bzw. eine ebenfalls unüberschaubare Komplexität aufweist. Daher muss das Modell an mehreren Einzelbeispielen angewendet und überprüft werden, um auf seine Tauglichkeit schließen zu können.
Wird der Fokus von den Details des SIKKs auf den SI-KK als Ganzes verlagert, so kann vermutet werden, dass die Komplexität des Konzepts eine Behäbigkeit impliziert, welche die Anwendbarkeit des Konzepts in einem dynamischen Umfeld verringert. Auslöser für diese Vermutung ist

die Feststellung der Interviewpartner, dass einzelne Infrastrukturkompositionen wie beispielsweise Datenbanksysteme auf einem Zusammenspiel unterschiedlichster Soft- und Hardwarekomponenten beruhen. Nahezu jede dieser Komponenten wird von einem anderen Hersteller mit anderen Support- und Wartungsvereinbarungen geliefert. Garantiert der Hersteller des Betriebssystems beispielsweise nur dann, dass selbiges einwandfrei funktioniert, wenn der Kunde die zur Verfügung gestellten Patches einspielt, ändert sich bei der Akzeptanz dieser Patches die betrachtete Infrastrukturkomponente im Detail ab. Wird der Wartungszyklus des SI-Katalogs beispielsweise auf ein Jahr festgesetzt, kann es vorkommen, dass die einzelnen Hersteller mindestens 1 bis 3 Patches zur Verfügung gestellt haben. Das Akzeptieren und Aufspielen eines Patches (z. B. ein Storage-Treiber) kann jedoch dazu führen, dass eine Infrastrukturkomponente der Obergruppe *Datenbanken* nicht mehr mit einer bestimmten Infrastrukturkomponente der Obergruppe *Storage Systeme* kombiniert werden kann – obwohl dies laut Architekturmuster möglich ist. Die Folge dieser Dynamik ist, dass entweder der SI-Katalog bei jedem Patch erneuert werden muss, dass Patches nicht akzeptiert und aufgespielt werden oder dass Architekturmuster vor ihrer Konfiguration auf mögliche Änderungen hin überprüft werden müssen. Der Dokumentationsaufwand der Architekturmuster in UML-Form kann bei häufigen Änderungen zu einem erheblichen Mehraufwand führen.

Zeitmangel

Die Modellkomplexität impliziert einen hohen Pflegeaufwand, der durch die synchrone Dokumentation in UML- und Matrizenform erhöht wird. Dass ein entsprechender Dokumentationsaufwand notwendig und langfristig förderlich ist, wird nicht bezweifelt. Vielmehr stellten die Interviewpartner in Frage, wie sie diesen Dokumentationsaufwand zeitlich bewältigen können, wenn sie von Managementseite nicht anderweitig (z. B. durch mehr Personal) entlastet werden. Die reine Dokumentation in Matrizenform erscheint hierbei als akzeptable Dokumentationsform, die jedoch in ihrer Detailtreue nicht mit der UML vergleichbar ist.

Diversität der Architekten

Abgesehen von dem Zeitaufwand, den die Dokumentation in UML-Form erfordert, verlangt die Anwendung der UML entsprechende Kenntnisse von ihren Anwendern ab. Konkret bedeutet dies, dass derjenige, der die Rolle des Infrastrukturarchitekten ausfüllt, UML-Kenntnisse aufweisen

muss. Im betrachteten Unternehmen wird diese Rolle jedoch von Personen mit unterschiedlichsten Werdegängen, Altersklassen und Fähigkeiten ausgefüllt. Bevor mit der eigentlichen Anwendung des SI-KKs begonnen werden könnte, müsste die Mehrzahl derjenigen, die die Rolle des Architekten ausfüllen, neben der Schulung des SI-KKs auch in der Sprache UML geschult werden.

11.2.2.1.2 Potentiale

Standardisierung

Als generell positiv wurde die Einführung einer einheitlichen Verfahrensweise für das Infrastrukturarchitekturmanagement hervorgehoben. Der praktische Mangel an entsprechenden Verfahren bewirkt, dass jeder einzelne, der sich mit der Infrastrukturarchitektur auseinandersetzt, seine eigene Verfahrensweise entwickelt. Das Resultat sind unterschiedliche Notationen und teils inkompatible Dokumentationen, die sich in ihrer Detailtiefe drastisch unterscheiden.

Die Investition, die für die Erstellung des SI-Katalogs getätigt werden muss, wird sich sicherlich nicht im ersten Jahr des Einsatzes rentieren. Auch nicht im zweiten oder dritten Jahr. Inwieweit ein monetärer Nutzen überhaupt erbracht werden kann ist fraglich. Die Reduktion der Infrastrukturkomponentenvarianten bedeutet zwar auf der einen Seite eine Reduktion der Komplexität des Systems Infrastruktur, auf der anderen Seite schränkt eine solche Reduktion aber auch die Flexibilität des Architekten ein.

Qualitativer Aspekt

Unbestritten ist jedoch, dass die Anwendung des SI-KKs die Qualität der Leistungserbringung steigert. Dieser Vorteil stärkt die eigene Position der hauseigenen Finanzdienstleistungsinformatik sofern sie sich gegenüber konkurrierenden Outsourcinganbietern monetär rechtfertigen muss.

Hinzu kommt, dass die mit dem SI-KK verbundene Ist-Aufnahme sämtlicher Infrastrukturkomponenten einen Überblick über die tatsächliche Variantenvielfalt verschafft, die sich im Laufe der Jahrzehnte teils unkontrolliert entwickeln konnte. Das Wissen bzgl. der vorhandenen Komponenten und ihrer Varianten wird ebenfalls als Qualitätsmerkmal gewertet, da auf Basis dieses Wissens Optimierungen angestrebt werden können.

11.2.2.2 Auswertung der spezifischen Befragung

Bei der Auswertung der Befragungsergebnisse muss berücksichtigt werden, dass eine fundierte Aussage eine praktische Anwendung des SI-KKs über einen längeren Zeitraum erfordert. Die Interviewpartner kannten nicht alle positiven und negativen Facetten des SI-KKs, da das Konzept in einem sehr kurzen Zeitraum an einem Musterbeispiel vorgeführt wurde. Alle Bewertungsausprägungen basieren daher auf theoretischen Annahmen seitens der Befragten.

#	**Frage**	**Ergebnis**
1	Welche Auswirkung hat das SI-KK auf den Grad der Komplexität der Infrastrukturarchitektur?	senkend: 3x ohne Auswirkung: 2x steigernd: 0x
2	Welche Auswirkung hat das SI-KK auf den Grad der Dynamik der Veränderung, die den Bereich der Infrastrukturarchitektur betreffen?	senkend: 1x ohne Auswirkung: 4x steigernd: 0x
3	Welche Auswirkung hat das SI-KK auf die Variantenvielfalt einzelner von der Infrastruktur erfasster Komponenten?	senkend: 3x ohne Auswirkung: 2x steigernd: 0x
4	Inwieweit hilft das SI-KK dabei, die Komplexität der Infrastrukturarchitektur im Allgemeinen zu beherrschen?	überhaupt nicht: 0x wenig: 0x ausreichend: 5x sehr: 0x
5	Inwieweit hilft das SI-KK dabei, die Komplexität der Infrastrukturarchitektur speziell in Ihrem Unternehmen zu beherrschen?	überhaupt nicht: 1x wenig: 3x ausreichend: 1x sehr: 0x

Tabelle *11.5*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 1 (Fallstudie 2).

Problemdimension 1

Die in Tabelle 11.5 abgedruckten Ergebnisse der Befragung bzgl. der Komplexität der Infrastrukturarchitektur zeigen eine ähnliche Tendenz wie im Fall der Befragung im Rahmen der ersten Fallstudie.

Problemdimension 2

Die Problemdimension 2 ist ähnlich ausgeprägt wie im Fall der Postbank Systems. Die Beantwortung der Frage 5 wurde unter dem Vorbehalt beant-wortet, dass das betrachtete Unternehmen die notwendigen Ressourcen zur Verfügung stellt, um den Katalog zu erstellen und langfristig zu pflegen.

#	Frage	Ergebnis
1	Inwieweit hilft das SI-KK dabei, die Wissenskomplexität bzw. das Wissensvolumen der Infrastrukturarchitektur zu beherrschen?	überhaupt nicht: 0x wenig: 1x ausreichend: 4x sehr: 0x
2	Inwieweit trägt das SI-KK dazu bei die sog. Kopfmonopole abzubauen und das Wissen bzgl. komplexer, von der Infrastrukturarchitektur betrachteter Systeme allgemein zugänglich zu dokumentieren?	überhaupt nicht: 0x wenig: 0x ausreichend: 4x sehr: 1x
3	Inwieweit trägt das SI-KK dazu bei die in einzelnen Projekten getroffenen Entscheidungen bzgl. der Ausgestaltung der Infrastrukturarchitektur nachvollziehbar zu dokumentieren, so dass Sie in mehr als 60% aller Fälle nachvollziehen können weshalb sich das Projektteam in einem spezifischen Vorhaben für die Architektur vom Typ A und gegen Typ B entschieden hat?	überhaupt nicht: 0x wenig: 0x ausreichend: 5x sehr: 0x
4	Inwieweit trägt das SI-KK dazu bei, dass eine standardisierte Notation zur Dokumentation der Infrastrukturarchitektur verwendet wird?	überhaupt nicht: 0x wenig: 0x ausreichend: 3x sehr: 2x
5	Inwieweit trägt das SI-KK dazu bei, die Wissenskomplexität bzw. das Wissensvolumens der Infrastrukturarchitektur Ihrem Unternehmen zu beherrschen?	überhaupt nicht: 0x wenig: 2x ausreichend: 3x sehr: 0x

Tabelle *11.6*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 2 (Fallstudie 2).

Problemdimension 3

Deutlich positiver als im Rahmen der ersten Fallstudie sind die Befragungsergebnisse in Bezug auf die 3. Problemdimension ausgeprägt. Es kann spekuliert werden, dass dies mit der im Vergleich zur Postbank Systems unterschiedlich ausgeprägten Problemdimension zusammenhängt. Der Vergleich von Tabelle 11.4 und Tabelle 11.7 zeigt, dass zum einen die Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik ausübt, bei der Postbank Systems stärker ausgeprägt ist, und dass zum anderen der Kostendruck, der auf die Postbank Systems ausgeübt wird, höher eingeschätzt wurde als im Fall der BHW.

#	Frage	Ergebnis
1	Inwieweit trägt das SI-KK dazu bei, sich der Dynamik, die das System Finanzdienstleistung auf das System Finanzdienstleistungsinformatik in Ihrem Unternehmen auswirkt, anzupassen?	überhaupt nicht: 0x wenig: 1x ausreichend: 3x sehr: 1x
2	Inwieweit trägt das SI-KK dazu bei, sich dem Kostendruck, der auf Ihr Unternehmen ausgeübt wird, anzupassen?	überhaupt nicht: 0x wenig: 1x ausreichend: 3x sehr: 1x

Tabelle *11.7*: Ergebnisse der Befragung bzgl. der konkreten Ausprägung der Problemdimension 3 (Fallstudie 2).

12 Zusammenfassung & Ausblick

Die zentrale Forschungsfrage, mit der sich diese Arbeit auseinandergesetzt hat, suchte nach der Gestaltung eines Modellbasierten Lösungsansatzes und dessen Einführung in ein Unternehmen mit dem Zweck, die Komplexitätsbedingten Entscheidungsprobleme in der Infrastrukturarchitektur der Finanzdienstleistungsinformatik unter der Beachtung dreier Problemdimensionen zu erfassen und entsprechend methodisch zu unterstützen. Die Betrachtung und Bewertung der existierenden Modellansätze hat gezeigt, dass das Thema Infrastrukturarchitektur bzw. deren Entwurf trotz seiner bekannten Komplexität noch immer eine sehr untergeordnete Rolle in der Basisliteratur spielt. Vorwiegend steht die Softwareentwicklung im Vordergrund. Einzig ITIL nimmt sich dieser Thematik an, stellt jedoch nur ein allgemeingültiges Rahmengerüst zur Verfügung, das sich eher auf die Management- und Prozessaktivitäten konzentriert, als ein konkret anwendbares Lösungsmodell zur Verfügung zu stellen.

Ideengeber für die eigene Modellentwicklung waren die Industrialisierung im Umfeld der Softwareentwicklung und die aus dem klassischen Maschinenbau bekannte Plattformtechnik, bei der die innere Produktkomplexität durch eine Fixierung der Elementkompositionen reduziert wird.

Im Rahmen eines Aktionsforschungsprojektes bei der Postbank Systems konnten die komplexen Zusammenhänge der Infrastrukturarchitektur sozusagen „am lebenden Objekt“ untersucht und mit Hilfe eines Meta-Modells beschrieben werden. Die Ergebnisse der Aktionsforschung haben gezeigt, dass das Meta-Modell eine solide Ausgangsbasis für zukünftige Optimierungen darstellt. Voraussetzung hierfür ist, dass es kontinuierlich an die Erfahrungen im Umfeld des Infrastrukturarchitekturmanagements in einem Unternehmen angepasst wird. Eine weitere Voraussetzung ist die Institutionalisierung des Infrastrukturarchitekturmanagements, zu deren Mitteln auch die Methodik des SI-Katalogs zählen kann.

Inwieweit der Einsatz der Methodik einem Unternehmen einen monetären Gewinn verspricht, konnte im Rahmen dieser Arbeit nicht ermittelt

werden. Grund hierfür ist, dass das auf zwei Jahre begrenzte Forschungsprojekt zu kurz war, um reale Einsparpotentiale messen zu können. Daher lässt sich die Auswirkung der Methodik auf den Infrastrukturarchitekturentwurf nur schätzen. Aus wissenschaftlicher Sicht ist es allerdings sehr empfehlenswert, die bei der Postbank Systems AG implementierte und angewendete Methodik über einen weiteren Zeitraum von 2 bis 4 Jahren zu beobachten und ggf. zu modifizieren. Zudem empfiehlt sich auch, die gewonnen Ergebnisse mit einer weiteren Studie zu vergleichen, die bei einem anderen Finanzdienstleister über einen ähnlich langen Zeitraum durchzuführen ist. Dass die Investition eines Finanzdienstleisters in eine solche Studie bzw. in die Einführung der Methodik sukzessiv Teilerträge einbringen kann, zeigt das Ergebnis, das im Rahmen der Bewertung der Methodik gewonnen wurde. Auch wenn die Teilerträge vorwiegend qualitativer Natur sind, versprechen sie einen monetären Ertrag in der Zukunft. Grundsätzlich stellt sich die Frage, welche Kosten entstünden, wenn die Methodik nicht angewendet wird, sondern wenn die Finanzdienstleistungsinformatik so fortfährt, wie sie dies bis dato praktiziert hat.

Neben der Fortführung der wissenschaftlichen Untersuchung bzgl. des in dieser Arbeit vorgeführten Konzepts ist es ebenfalls interessant, inwieweit das Plattformkonzept einen wissenschaftlich undokumentierten Einzug in die Finanzdienstleistungsinformatik gehalten hat. Daher bleibt zu hoffen, dass sich zum einen ein weiteres Unternehmen finden lässt, das die mit dieser Arbeit initiierte Forschung fortführt, und zum anderen, dass ein anderer ambitionierter Forscher die hier gewonnenen Erkenntnisse empirisch hinterfragt.

Literaturverzeichnis

A

Akao (1990) Akao, Y., *QFD – Integrating Customer Requirements into Product Design*, Cambridge, 1990

Albert (1984) Albert, H., *Wissenschaftstheorie*. In: Grochla, E., Wittmann, W. (Hrsg.), *Handwörterbuch der Betriebswirtschaft*, 4. Auflage, Stuttgart, 1984, S. 4674-4692

Alexander et al. (1977) Alexander, Ch.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, Sh., *A Pattern Language*, New York, 1977

Ansoff (1965) Ansoff, H. I., *Corporate Strategy*, New York, 1965

Applegate et al. (2001) Applegate, L. M.; McFarlan, F. W.; McKenney, J. L., *Corporate Information Systems Management*, 5. Auflage, Homewood, 2001

Ardissino et al. (2001) Ardissono L.; Felfernig A.; Friedrich G.; Jannach D.; Schäfer R.; Zanker M, *Customer-Adaptive and Distributed Online Product Configuration in the CAWICOMS Project*. In: Aldanondo, M. (Hrsg.), *17th International Joint Conference on Artificial Intelligence - Configuration Workshop*, Lyon, 2001, S. 8-14

Argyris et al. (1985) Argyris, C.; Putnam, R.; Smith, D., *Action Science: Concepts, Methods and Skills for Research and Intervention*, San Francisco, 1985

Ashby (1956) Ashby, W. R., *An Introduction to Cybernetics*, London, 1956

Avison et al. (1999) Avison, D.; Lau, F.; Myers, M.; Nielsen, P.A., *Action Research*. In: Communications of the ACM, Jg. 42, Nr. 1, S. 94-97

B

Baldwin/Clark (1998) Baldwin, C. Y.; Clark, K. B., *Modularisierung: Ein Konzept wird universell*. In: Harvard Business Manager, 20. Jahrgang, 1998, Nr. 2, S. 39-48

Balzert (1998) Balzert, H., *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*, Heidelberg/Berin, 1998

Balzert (2001) Balzert, H., *Lehrbuch der Software-Technik*, Band 1, Heidelberg, 2001, 2. Auflage

Bartsch (2005) Bartsch, S. M., *Die Entwicklung der Rolle des Chief Information Officer (CIO): Einflussfaktoren, Aufgaben und Qualifikationen*, Diplomarbeit im Fach Wirtschaftsinformatik, vorgelegt in der Diplomprüfung im Studiengang Wirtschaftsinformatik der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität Köln, Köln, 2005

Baskervil- Baskerwille, R.; Wood-Harper, A. T., *Diversity of Information Sys-*

le/Wood-Harper (1998)	*tems Action Research Methods.* In: European Journal of Information Systems, Heft 2, 1998, S. 90-107
Bates et al. (2003)	Bates, M. D.; Davis, K. B.; Haynes, D. D., *Reinventing IT Services*. In: The McKinsey Quarterly, Heft 5, 2003, S. 143-153
Bauer (1993)	Bauer, F. L., *Software-Engeneering – wie es begann*. In: Informatik Spektrum 16, S. 259-260
Bayer et al. (2005)	Bayer, J.; Forster, T.; Lehner, A.; Ocampo, A.; Richter, E.; Weiland, J., *Feature- und Entscheidungsmodellbasierte Varianteninstanziierung im PESOA-Prozess*, PESOA Report 21, 2005
BCG (2001)	The Boston Consulting Group GmbH, *Wofür IT? – Informationstechnologie strategisch einsetzen*, Studie, München, 2001
Behrendt et al. (1998)	Behrendt, S.; Pfitzner, R.; Kreibich, R.; Hornschild, K., *Innovationen zur Nachhaltigkeit - Ökologische Aspekte der Informations- und Kommunikationstechniken*, Berlin/Heidelberg/New York, 1998
Benaroch/Kaufmann (1999)	Benaroch, M.; Kaufmann, R., *A case for using real option pricing analysis to evaluate information technology project investments*. In: Information Systems Research, Vol. 10, No. 1, 1999, S. 70-86
Benson et al. (1985)	Benson, R. J.; Parker, M. M., *Enterprise-wide Information Management – An Introduction to the Concepts*. In: IBM Los Angeles Scientific Center Reports G320-2768, Los Angeles, 1985
Berensmann (2004)	Berensmann, D., *Gesamtbankarchitektur der Deutschen Postbank AG*. In: Moormann/Fischer (2004), S. 59-77
Berensmann (2005)	Berensmann, D., *IT matters – but who cares?*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 273-277
Birkhofer (1980)	Birkhofer, H., *Analyse und Synthese der Funktionen technischer Produkte*. In: Fortschrittsberichte, Reihe 1, Nr. 70, Düsseldorf
Birkhölzer/Vaupel (2003)	Birkhölzer, T.; Vaupel, J., *IT-Architekturen. Planung, Integration, Wartung*, Berlin/Offenbach, 2003
Black/Scholes (1973)	Black, F.; Scholes, M., *The Pricing of Options and Corporate Liabilities*. In: Journal of Political Economy, No. 81, May/Jun, 1973, S. 637-659
Blicher/Quast (2005)	Blicher, K; Quast, N., *Variantenmanagement im Maschinenbau*, erhältlich unter http://www.logistik-inside.de/fm/2248/bichler.pdf, Abruf am 20.11.2005
Bliss (2000)	Bliss, C., *Management von Komplexität: Ein integrierter, systemtheoretischer Ansatz zur Komplexitätsreduktion*, Wiesbaden, 2000
Böhmann (2004)	Böhmann, T., *Modularisierung von IT-Dienstleistungen*, Wiesbaden, 2004
Böhmann/Kcrmar (2005a)	Böhmann, T.; Krcmar, H., *Einfach besser? Zur Anwendbarkeit des industriellen Komplexitätsmanagements auf variantenreiche IT-Dienstleistungen*. In: Ferstl, O. K.; Sinz, E. J.; Eckert, S.; Issen-

horst, T. (Hrsg.), Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety, 7. Internationale Tagung Wirtschaftsinformatik 2005, Bamberg, 23.2.2005 - 25.2.2005, S. 449-468

Böhmann/Krcmar (2005b) Böhmann, T.; Krcmar, H., *Modularisierung: Grundlagen und Anwendung bei IT-Dienstleistungen*. In: In Hermann, T.; Krcmar, H.; Kleinbeck, U. (Hrsg.), *Konzepte für das Service Engineering - Modularisierung, Prozessgestaltung und Produktivitätsmanagement,*Heidelberg, S. 45-48

Bongulielmi (2002) Bongulielmi, L., *Die Konfigurations- & Verträglichkeitsmatrix als Beitrag zur Darstellung konfigurationsrelevanter Aspekte im Produktentstehungsprozess*, Dissertation, Zürich, 2002

Bongulielmi et al. (2002) Bongulielmi, L.; Henseler, P.; Puls, Ch.; Maier, M., *The K- & V-Matrix-Method in Comparison with Matrix-Based Methods supporting Modular Product Family Architectures*. NordDesign 2002 - Visions and Values in Engineering Design, 14-16 August, Norwegian University of Science and Technology, NTNU Trondheim, Norway, 2002

Booch (1991) Booch, G., *Object-oriented design with applications*, Redwood City, 1991

Booch et al. (1999) Booch, G.; Rumbaugh, J.; Jacobson, I., *The Unified Modeling Language User's Guide*, New York, 1999

Bortz (1984) Bortz, J., *Lehrbuch der empirischen Forschung*, Berlin, 1984

Bracker (1980) Bracker, J., *The Historical Development of the Strategic Management Concept*. In: Academy of Management Review, 5. Jg, Heft 2, New York, 1980, S. 219-224.

Brockhaus (1997) Brockhaus, *Der Brockhaus – in fünfzehn Bänden*, Leipzig/Mannheim, 1997

Bubik et al. (2000) Bubik, R.; Quenter, D.; Ruppelt, T., Informationstechnik – selten geschäftsbezogen geführt. In: Harvard Business Manager, Nr. 2, 2000, S. 102-111

Bullinger/Thaler (1992) Bullinger, H.-J.; Thaler, K., *Vom Teilefertiger zum Wertschöpfungspartner*. In: Technische Rundschau, Heft 46, 1992, S. 13-16

Bund (1998) Bund, M., *Modulare Fertigungsstrukturen*. In: Das Wirtschaftsstudium (WISU), 27. Jahrgang, 1998, Nr. 5, S. 566-568

Buschmann et al. (1998) Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Stal, M., *Pattern-orientierte Softwarearchitektur. Ein Pattern-System*, Bonn, 1998

C

Caroll et al. (1980) Caroll, J. M.; Thomas, J. C. , *Presentation and representation in design problem solving*. In: British Journal of Psychology, Jahrgang 93, 1980, S. 143-153

Chandler (1964) Chandler, A. D. Jr., *Giant Enterprise - Ford, General Motors, and*

the Automobile Industry, New York, 1964

Clausewitz (2002) — Clausewitz, C. v., *Vom Kriege*, München 2002 (Original: *Vom Kriege*, Hinterlassenes Werk des Generals Carl von Clausewitz, Berlin, 1832)

Coad/Yourdon (1991) — Coad, P, Yourdon, E., *Object-Oriented Analysis,* 2. Auflage, Englewood Cliffs, 1991

Coetzee (2002) — Coetzee, J. M., *Die jungen Jahre*, Frankfurt, 2002

CW (2005) — o. N., *Top 100 Liste*. In: *Top 100. Die erfolgreichsten deutschen ITK-Unternehmen*, Computerwoche Spezial, September 2005, S. 46f

Czarnecki/ Eisenecker (2000) — Czarnecki, K.; Eisenecker, W. U., *Generative Programmierung*, Boston, 2000

D

Decker (2002) — Decker, C., *Variantenmanagement im Sondermaschinenbau*. In: Firchau et al. (2002), S. 124-147

Denert (1994) — Denert, E., *Software Engineering in Business*, 13. IFIP Congress, 1994, Hamburg

Dern (2003) — Dern, G., *Management von IT-Architekturen*, Wiesbaden, 2003

Dietrich (2004) — Dietrich, L., *Die ersten 100 Tage des CIO – „Quick Wins" und Weichenstellung*. In: Dietrich/Schirra (2004), S. 45-82

Dietrich/Schirra (2004) — Dietrich L.; Schirra, W. (Hrsg.), *IT im Unternehmen*, Heidelberg, 2004

Dietz et al. (1999) — Dietz, J.; Mallens, P.; Goedvolk, H.; Rijsenbrij, D., *A Conceptual Framework fort the Continuous Alignment of Business and ICT*, Technical University Delft & Cap Gemini, Whitepaper, Delft, 1999

DIN 199 — DIN 199, Teil 2: Begriffe im Zeichnungs- und Stücklistenwesen, Berlin et al., 1977

Dresdner Bank (2004) — Dresdner Bank AG, *Finanzbericht Dresdner Bank Konzern*, Frankfurt, 2004

Duden (1990) — Duden, *Das Fremdwörterbuch*, Mannheim/Leipzig/Wien/Zürich, 1990

Dudenhöfer (2001) — Dudenhöfer, F., *Trends der Automobilwirtschaft: Ein neues Branchenbild entsteht*. In: VDI-Berichte, Band 1653, Düsseldorf, 2001, S. 12-34

Dujmovic (1997) — Dujmovic, S., *Katalog-basierte Unterstützung der Wiederverwendung*, 7. Kolloquium Software-Entwicklung, Technische Akademie Esslingen (TAE), 1997

Dymond (2002) — Dymond, K. M., *CMM Handbuch*, Berlin/Heidelberg/New York, 2002

E

Earl (1987) — Earl, M. J., *Information systems strategy formulation*. In: Boland,

R. J.; Hirschheim, R. A. (Hrsg.), *Critical Issues in Information Systems Reasearch*, 1987, S. 157-178

Earl (1992) Earl, M. J., *Putting IT in its place: A polemic for the nineties*. In: Journal of Information Technology, 7. Jg., Nr. 2, Oxford, S. 100-108

Earl (1996) Earl, M. J., *Integrating IS and the Organization*. In: Earl, M. J. (Hrsg.), *Information Management*, Oxford et al., 1996, S. 485-502

eBanker (2003) o. V., *Rückkehr zur Kernkompetenz*. In: eBanker, Ausgabe 05-06, 2003, S. 24

Eberhard (1996) Eberhard, D., *Zur Relevanz der Dokumentation und Kommunikation in der betrieblichen Softwareentwicklung und –wartung*, Dissertation, Paderborn, 1996

Ehrlenspiel et al. (1998) Ehrlenspiel, K.; Kiewert, A.; Lindemann, U., *Kostengünstig Entwickeln und Konstruieren*, Berlin/Heidelberg, 1998

Eisenhardt (1989) Eisenhardt, K. M., *Building theories from case study research*. In: Academy of Management Review, Vol. 14, No. 4, S. 488-511, 1989

Ells/Dewar (1979) Ells, J. G.; Dewar, R. E., *Rapid Comprehension of Verbal and Symbolic Traffic Sign Messages*, New York, 1979

entory (2005) entory AG, *„Nichts geht ohne Kommunikation“*. In: FIN.KOM, Ausgabe 2/2005, S. 10-11

Erder/Pureur (2003) Erder, M; Pureur, P., *QFD in the Architecture Development Process*. In: IT Professional, Nov/Dez 2003, Volume 5, Issue 6, S. 44-52

F

Feiler/Humphrey (1992) Feiler, P. H.; Humphrey, W. *S., Software Process Development and Enactment: Concepts and Definitions*. In: *Proceedings of the 2nd International Conference on the Software Process*, IEEE Computer Society Press, Februar 1992, S. 28–40

Felfernig et al. (2000) Felfernig, A.; Friedrich, G. E.; Jennach, D., *UML as Domain Specific Language for the Constructionof Knowledge-Based Configuration Systems*. In: International Journal of Software Engineering an Knowledge Engineering, Vol. 10, No. 4, 2000, S. 449-469

Felfernig et al. (2001) Felfernig, A.; Friedrich, G.; Jannach, D., *Conceptual modelling for configuration of mass-customizable products*. In: Artificial Intelligence in Engineering 15 2, 2001, S. 165-176.

Firchau et al. (2002) Firchau, N. L.; Franke, H.-J.; Huch, B.; Menge, M., *Variantenmanagement: Variantenvielfalt in Produkten und Prozessen erfolgreich beherrschen*. In: Franke et al. (2002), S. 1-25

Fischer/Rothe (2004) Fischer, Th.; Rothe, A., *Wertbeitrag der Informationstechnologie*. In: Moormann/Fischer (2004), S. 19-41

Flaig et al. Flaig, G.; Nierhaus, W.; Kuntze, O.-E.; Gebauer, A.; Henzel, S.;

(2005) Hülsewig, O.; Kaltschütz, A.; Langmantel, E.; Meister, W.; Ruschinski, M.; Schimpermann, B.; Wollmershäuser, T., *ifo Konjunkturprognose 2005/2006: Nur zögerliche Erhohlung*, ifo Institut für Wirtschaftsforschung, München, Juni 2005

Frank et al. (1998) Frank, U.; Klein, S.; Krcmar, H.; Teubner, A., *Aktionsforschung in der WI – Einsatzpotentiale und –probleme*. In: Schütte, R.; Siedentopf, J.; Zelewski, S. (Hrsg.), *Wirschaftsinformatik und Wissenschaftstheorie. Grundpositionen und Theoriekerne*. Arbeitsberichte des Instituts für Produktion und Industrielles Informationsmanagement, Nr. 4, Essen, 1998, S. 71-90

Franke (1998) Franke, H.-J., *Produkt-Variantenvielfalt. Ursachen und Methoden zu ihrer Bewältigung.* VDI-Berichte 1434, Düsseldorf, 1998

Franke (2003) Franke, D., *Die 1.000 größten Banken der Welt: Wieder auf Kurs*, Online-Artikel der Zeitschrift *die bank*, http://www.die-bank.de, Abruf am 22.07.2005

Franke et al. (2002) Franke, H.-J.; Hesselbach, J.; Huch, B.; Firchau, N. L. (Hrsg.), *Variantenmanagement in der Einzel- und Kleinserienfertigung*, München/Wien, 2002

Franke/Firchau (1998) Franke, H.-J.; Firchau, N. L., *Zusammenfassender Zwischenbericht des Kalenderjahres 1998 für das BMBF-Projekt „Methoden und Werkzeuge zur Konstruktion variantenreicher Produktspektren in der Einzel und Kleinserienfertigung – EVAPRO“*. Institut für Konstruktionslehre, Maschinen und Feinwerkelemente, TU Braunschweig, 1998

Fuchs (2005) Fuchs, H. J., *Grenzüberschreitende Bankenfusionen – Elefanten-Hochzeiten in Europa*. In: die bank, Nr. 8, 2005, S. 8-13

G

Galitz (1997) Galitz, W., *The essential guide to user interface design: An introduction to GUI design, principles and techniques*, New York/Toronto/Singapore, 1997

Gallagher/Appen-zeller (1999) Gallagher, R.; Appenzeller, T., *Beyond Reductionism*. In: Science 2, Vol. 284, No. 5411, April 1999, S. 79-109

Gamma (1991) Gamma, E., *Objektorientierte Softwareentwicklung am Beispiel von ET++: Design-Muster, Klassenbibliothek, Werkzeuge*, Doktorarbeit, Universität von Zürich, Zürich, 1991

Gamma et al. (1995) Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J., *Design Patterns. Elements of Reusable Object-Oriented Software*. New York, 1995

Ganswindt (2004) Ganswindt, C., *Konsolidierung von IT-Plattformen am Beispiel der Deutschen Lufthansa AG und ihrer Allianzpartner*. In: Dietrich/Schirra (2004), S. 157-171

Gausemeier et Gausemeier, J.; Flath M., *Geschäftsprozessübergreifendes Pro-*

al. (1998) *duktstrukturmanagement - Integration der bereichsspezifischen Sichten bei der durchgängigen rechnerunterstützen Entwicklung von variantenreichen Erzeugnissen*. In: Effektive Entwicklung und Auftragsabwicklung variantenreicher Produkte, Düsseldorf, VDI-Bericht, 1434, 1998

Gino (2002) Gino, F., *Complexity measures in decomposable structures*, 2nd EURAM Conference on "Innovative Research in Management", 9.-11. Mai, Stockholm, 2002

Gittens (1986) Gittens, D., *Icon-based human-computer interaction*. In: International Journal of Man Machine Studies, Vol. 24, 1986, S. 519-543

Glinz (1999) Glinz, M., *Eine geführte Tour durch die Landschaft der Software-Prozesse und –Prozessverbesserung*. In: Informatik 6, 1999, S. 7-12

Göpfert/Steinbrecher (2000) Göpfert, J.; Steinbrecher, M., *Modulare Produktentwicklung leistet mehr*. In: Harvard Business Manager, 3/2000, S. 20-28

Gorriz/Habermann (2004) Gorriz, M.; Habermann, T., *Gezieltes und strukturiertes Outsourcing von IT*. In: Dietrich/Schirra (2004), S. 235-260

Götze (1994) Götze, U., *Strategische Planung auf der Grundlage von Szenarien*. In: Bloech, J. et. al. (Hrsg.), *Strategische Planung – Instrumente, Vorgehensweisen und Informationssysteme*, Heidelberg, 1994, S. 100-124

Grant (1998) Grant, R. M., *Contemporary Strategy Analysis*, Malden/Oxford, 1998

Greiner et al. (2005) Greiner, T.; Durst, M.; Maget, S., *IT-Architektur-Controllingframework – Entwicklung und Einsatz bei der norisbank AG.* In: Banking and Information Technologie, Band 6, Heft 4, Dezember, 2005, S. 21-36

Griffiths (1994) Griffiths, C., *Responsibility for IT: a grey area of management*. In: Willcocks, L. (Hrsg.), *Information Management: The Evaluation of Information Systems Investments*, London, S. 233-251

Grohmann (2003) Grohmann, H. H., *Prinzipien der IT-Governance*. In: Meier, A. et al. (Hrsg.), *Strategisches IT-Management*, HMD – Praxis der Wirtschaftsinformatik, Heft 232, Heidelberg, 2003, S. 17-23

Größler et al. (2003) Größler, A.; Grübener, A.; Hasenpusch, J., *Configurations of international and external complexity in manufacturing companies*, EurOMA Konferenz, Cernobbio, 2003

Groth (2004) Groth, A., *Realisierung von IT-Synergien in einem paneuropäischen Konzern*. In: Dietrich/Schirra (2004), S. 173-200

Günter/Krebs (1999) Günter, A.; Kühn, C., *Knowledge-based configuration – survey and future directions*. In: Puppe, F., *XPS-99: Knowledge Based*

Systems, Proceedings 5th Biannual German Conference on Knowledge Based Systems, 1999

H

Habermas (1987) — Habermas, J., *Theorie des kommunikativen Handelns*, 2. Bank, 4. Auflage, Frankfurt, 1987

Hackethal/Schmidt (2005) — Hackethal, A.; Schmidt, H., *Structural Change in the German Banking System?*. In: Revue d'Economie Financière, Vol. 78, Paris, 2005, S. 125-169

Hafner et al. (2004) — Hafner, M.; Schelp, J.; Winter, R., *Architekturmanagement als Basis effizienter und effektiver Produktion als IT-Services*. In: Meier/Myrach (2004), S. 54-66

Hafner/Hunziker (2003) — Hochstein, A.; Hunziker, A., *Serviceorientierte Referenzmodelle des IT-Managements.* In: HMD – Praxis der Wirtschaftsinformatik, 40. Jg., Heft 232, 2003, S. 46-56

Handelsblatt (2005) — o. V., *Fusion setzt Konkurrenz unter Zugzwang*, Handelsblatt Online, 13. Juni 2005, http://www.handelsblatt.com, Abruf am 22.07.2005

Harrison/Varveris (2004) — Harrison, D.; Varveris, L., *Introduction to TOGAF*, Popkin Software, 2004

Hartenstein (2003) — Hartenstein, K., *Am Puls der Börse*. In: eBanker, Ausgabe 05-06, 2003, S. 28-29

Hegering et al. (1999) — Hegering, H.-G.; Abeck, S.; Neumair, B., *Integriertes Management vernetzter Systeme – Konzepte, Architekturen und deren betrieblicher Einsatz*, Heidelberg, 1999

Heinrich (2001) — Heinrich, L. J., *Wirtschaftsinformatik*, 2. Auflage, München/Wien, 2001

Heinrich (2002) — Heinrich, L. J., *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*, München, 2002

Heinzl (2001) — Heinzl, A., *Zum Aktivitätsniveau empirischer Forschung in der Wirtschaftsinformatik – Erklärungsansatz und Handlungsoptionen,* Lehrstuhl für Betriebswirtschaftslehre VII, Wirtschaftsinformatik, Universität von Bayreuth, Arbeitspapier 7/2001, Bayreuth, 2001

Hellingrath et al. (2000) — Hellingrath, B.; Wloka, J.; Wagenitz, A., *Simulation des Order-to-Delivery Prozesses in der Automobilindustrie*. In: VDI-Berichte, Band 1571, Düsseldorf, 2000, S. 91-102

Henderson/Venkatraman (1993) — Henderson, J. C; Venkatraman, N., *Strategic alignment: Leveraging information technology for transforming organizations*. In: IBM Systems Journal, Vol. 32, Nr. 1, 1993

Herz et al. (2001) — Herz, O.; Seybold, H.-J.; Strobl, G.-G., *Bildung für nachhaltige Entwicklung. Globale Perspektiven und neue Kommunikationsmedien*, Opladen, 2001

Herzwurm (1992)	Herzwurm, G., *Möglichkeiten und Grenzen des Einsatzes wissensbasierter Systeme im Computer Aided Software Engineering (CASE),* Dissertation, Universität zu Köln, Köln, 1992
Herzwurm et al. (1994)	Herzwurm, G.; Mellis, W.; Schmolling, K., *Software Factory – Ein Statusbericht*. In: Ein Statusbericht. HMD – Handbuch der Datenverarbeitung, Heft 180, 1995, S. 8ff
Herzwurm/Langner (2005)	Herzwurm, G.; Langner, T., *Architekturmanagment als Regulator zwischen Finanzdienstleistung und Finanzdienstleistungsinformatik am Beispiel der Postbank Systems AG*. In: BIT – Banking and Information Technology, Sonderheft zur Multikonferenz Wirtschaftsinformatik 2006, Heft 3/2005, S. 51-62
Hildebrand (2001)	Hildebrand, K., *Informationsmanagement: Wettbewerbsorientierte Informationsverarbeitung mit Standard-Software im Internet*, 2. Auflage, München/Wien, 2001
Hinterhuber (1990)	Hinterhuber, H. H., *Wettbewerbsstrategie*, Berlin, 1990
Hofer (2001)	Hofer, A. P., *Management von Produktfamilien – Wettbewerbsvorteile durch Plattformen*, Wiesbaden, 2001
Horton (1981)	Horton, F. W., *The Information Management Workbook : IRM made simple*, Washington DC, 1981
Humphrey (1987)	Humphrey, W. S., *A Method for Assessing the Software Engeneering Capability of Contractors*. Software Engineering Institute (SEI), CMU/SEI-87-TR-23, Pittsburgh, 1987
Hunt (1963)	Hunt, J. M. V., *Motivation inherent in information processing and action*. In: Harvey, O. J. (Hrsg.), *Motivation and social interaction: cognitive determinants*, New York, 1963
HypoVereinsbank (2003)	HypoVereinsbank AG, *Geschäftsbericht 2003 Jahresabschluss AG*, München, 2003
I	
IBM (1988)	IBM Deutschland GmbH, *Information Systems Management, Management der Informationsverarbeitung, Architektur und Überblick*, Band 1, o. O., 1988, S. 20ff
IEEE (2000)	IEEE, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE Std 1471-2000)*, IEEE Standards Collection, Software Engineering, IEEE, New York, 2000
In et al. (2001)	In, H.; Kazman, R.; Olson, D., *From requirements negotiation to software architectural decisions*, Proceedings of the First International Workshop From Software Requirements to Architectures (STRAW'01), Toronto, 2001
J	
Janßen (2005)	Janßen, R., *Die Psychologie des Entwicklers*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 284-286

Jeckle/Kern-Bausch (2000) Jeckle, M., Kern-Bausch, L., *Automatische Generierung objekt-orientierter Strukturen*. In: Proceedings KnowTech, Leipzig, 2000

Jochum (2005) Jochum, C., *Versinkt das IT-Management in der Bedeutungslosigkeit?*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 278-280

K

Kahlbrandt (1998) Kahlbrandt, B., *Software-Engineering*, Berlin/Heidelberg/New York, 1998

Kang et al. (1990) Kang, K. C.; Cohen, S. G.; Hess, J. A; Novak, W. E.; Peterson, A. S., *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report No. CMU/SEI-90-TR-2, Software Engineering Institute (SEI), Carnegie Mellon University, Pittsburgh, PA, 1990

Kaufmann/Schlitt (2004) Kaufmann, T.; Schlitt, M., *Effektives Management der Geschäftsbeziehung im IT-Outsourcing*. In: Meier/Myrach (2004), S. 43-53

Kazman et al. (2001) Kazman, R.; Asundi, J.; Klein, M., *Quantifying the Costs and Benefits of Architectural Decisions*, 23rd International Conference on Software Engineering (ICSE'01), 2001

Keller/Wendt (2003) Keller, F.; Wendt, S., *FMC: An Approach Towards Architecture-Centric System Development*, Proceedings of 10th IEEE Symposium and Workshops on Engineering of Computer Based Systems, Huntsville, Alabama, 2003

Kempa/Mann (2005) Kempa, M.; Mann, Z. A., *Model Driven Architecture*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 298-302

Kemper et al. (2004) Kemper, H.-G.; Hadjicharalambous, E.; Paschke, J., *IT-Servicemanagement in deutschen Unternehmen – Ergebnisse einer empirischen Studie zu ITIL*. In: Meier/Myrach (2004), S. 22-31

Kersten et al. (2004) Kersten, W.; Koppenhagen, F.; Meyer, Ch. M., *Strategisches Komplexitätsmanagement durch Modularisierung in der Produktentwicklung*. In: Spath, D., *Forschungs- und Technologiemanagement. Potentiale nutzen – Zukunft gestalten*, München, 2004, S. 211-217

Klawa (2004) Klawa, M.-A., *Konzeption und Implementierung von CRM-Systemen*. In: Moormann/Fischer (2004), S. 253-270

Klein et al. (1999) Klein, M.; Kazman, R.; Bass, L.; Carriere, S. J.; Barbacci, M.; Lipson, H., *Attribute-Based Architectural Styles, Software Architecture*. In: Proceedings of the First Working IFIP Conference on Software Architecture (WICSA1), San Antonio, Texas, February 1999, S. 225-243

Klostermeier/Seeger (2002) Klostermeier, J.; Seeger, H., V*orsprung für die Banken-IT*. In: CIO IT-Strategie für Manager, 2. Jahrgang, Mai, 2002, S. 21-30

Kneer (2003) Kneer, H., *Extended Service Level Management for the Provisioning of Streaming Internet Services*, Dissertation, Universität Zürich, 2003

Kneuper (2003) Kneuper, R., *CMMI. Verbesserung von Softwareprozessen mit dem Capability Maturiy Model Integration*, Heidelberg, 2003

König (2000) König, W., *Wolfgang König interviewt Dirk Berensmann zum Aufbau einer neuen IT-Tochter eines Dienstleistungsgroßunternehmens*. In: Wirtschaftsinformatik, Heft 42, Sonderheft IT & Personal, 2000, S. 56-59

Krcmar (1991) Krcmar, H., *Annäherungen an Informationsmanagement – Managementdisziplin und/oder Technologiedisziplin?* In: Staehle, W. H.; Sydow, J. (Hrsg.), *Managementforschung*, Berlin, 1991, S. 163-203

Krcmar (2005) Krcmar, H., *Informationsmanagement*, 4. Auflage, Berlin/Heidelberg/New York, 2005

Krebs et al. (2002) Krebs, Th.; Hotz, L.; Günter, A., *Knowledge-based Configuration for Configuring Combined Hardware/Software Systems*. In: Proceedings of 16. Workshop, Planen, Scheduling und Konfigurieren, Entwerfen (PuK2002), Freiburg, 10.-11. Oktober, 2002

Krönung (1998) Krönung, H.-D., *Die Bedeutung der Informationstechnologie für den Bankenmarkt der Zukunft*. In: PASS IT-Consulting AG, *Kreativ die Zukunft gestalten. Futurologische Beiträge zur Wettbewerbsfähigkeit*, Frankfurt, 1998, S. 39ff

Krupp (1995) Krupp, G., *Allfinanzpolitik*. In: Gerke, W.; Steimer, M. (Hrsg.), *Handwörterbuch des Bank- und Finanzwesens*, 2. Auflage, Stuttgart, 1995, S. 55-67

Kruschwitz (2005) Kruschwitz, L., *Investitionsrechnung*, 10. Auflage, München, 2005

Kubicek (1974) Kubicek, H., *Heuristischer Bezugsrahmen und heuristisch angelegtes Forschungsdesign als Elemente einer Konstruktionsstrategie empirischer Forschung*, Institut für Unternehmensführung der Freien Universität Berlin, Arbeitspapier 16/76, Berlin, 1976

Kühn (2000) Kühn, C., *Modeling structure and behaviour for knowledge based software configuration*. In: Sauer, J. (Hrsg.), *Proceedings Workshop Planen und Konfigurieren*, 14th European Conference on Artificial Intelligence (ECAI), Berlin (2000)

Kütz (2005) Kütz, M., *IT-Contolling für die Praxis. Konzeption und Methoden*, Heidelberg, 2005

L

Lea (1998) Lea, D., Christopher Alexander: An Introduction for Object-Oriented Designers. In: The Patterns Handbook: Techniques, Strategies, and Applications, Cambridge, 1998, S. 407-422

Lederer (2004) Lederer, A., *IT-Gesamtbankarchitektur in der Genossenschaftsorganisation*. In: Moormann, J.; Fischer, T. (Hrsg.), *Handbuch Informationstechnologie in Banken*, 2. Auflage, Wiesbaden, 2004, S. 78-94

Lewis/Stewart (2003) Lewis, G. J.; Stewart, N., *Measurement of Environmental Performance: An Application of Ashby's law*. In: Systems Research and Behavioural Science, No. 20, 2003, S. 31-52.

Lewis/Stewart (2003) Lewis, G. J.; Stewart, N., *Measurement of Environmental Performance: An Application of Ashby's law*. In: Systems Research and Behavioural Science, No. 20, 2003, S. 31-52.

Lingnau (1994) Lingnau, V., *Variantenmanagement. Produktionsplanung im Rahmen einer Produktdifferenzierungsstrategie*, Berlin, 1994

Liu (2002) Liu, S., *A Practical Framework for Discussing IT Infrastructure*. In: IT Professional, Vol. 04, No. 4, S. 14-21, Juli/August, 2002

Löckenhoff/Messer (1994) Löckenhoff, C.; Messer, T., *Configuration*. In: Breuker, J.; Van de Velde, W., *CommonKADS Library for Expertise Modelling*, Amsterdam, 1994, S. 197-212

M

Maher (1990) Maher, M. L., *Process Models for Design Synthesis*. In: AI Magazine, Winter Issue, 1990, S. 49-58

Maidl (2005) Maidl, J., *Spannungsfeld zwischen Standard und Prozessführerschaft*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 281-283

Malik (1996) Malik, F., *Strategie des Managements komplexer Systeme*, 5. Auflage, Bern, 1996

Malik (2000) Malik, F., *Führen Leisten Leben: Wirksames Management für eine neue Zeit*, 3. Auflage, Stuttgart/München, 2000

Männistö et al. (1996) Männistö, T.; Peltonen, H.; Sulonen, R., *View to Product Configuration Knowledge Modelling and Evolution*. In: Configuration, Papers from the 1996 AAAI Fall Symposium, AAAI Press, 1996, S. 111-118.

Männistö et al. (1996) Männisto, T.; Peltonen, H.; Sulonen, R., *View to Product Configuration Knowledge Modelling and Evolution*. In: Configuration Papers from the 1996 Fall Symp., Tech. Report FS-96-03, AAAI Press, Menlo Park, Californien, 1996, S. 111-118.

Männistö et al. (2000) Männistö, T.; Soininen, T.; Sulonen, R., *Configurable Software Product Families*, 14th European Conference on Artificial Intelligence ECAI 2000, Configuration Workshop, Berlin, 2000

Marchand (2000) Marchand, D., *Creating Business Value with Information*. In: Marchand D., *Competing with information: a manager's guide to creating business value with information content*, Chichester, UK, 2000,

S. 17-30

Marighetti (2001) Marighetti, L. P., *Management der Wertschöpfungsketten in Banken. Outsourcing, Reengineering und Workflow in der Praxis*, Wiesbaden, 2001

Marty (2004) Marty, R., *IT-Architektur: Gestaltungsmittel zur Umsetzung der IT-Strategie*. In: Moormann/Fischer (2004), S. 45-58

Mastor (1970) Mastor, A. A., *An Experimental Investigation and Comparative Evaluation of Production Line Balancing Techniques*. In: Management Science, Volume 16, 1970, S. 728-746

May (1998) May, L. L., *Major Causes of Software Project Failures*. In: Crosstalk, Juli 1998, S. 9-12

Mayer (1988) Meyer, B., *Object-Oriented Software Construction*, Hertfordshire, 1988

Meffert (2000) Meffert, H., *Marketing – Grundlagen marktorientierter Unternehmensführung*, Wiesbaden, 2000

Mehner/Wagner (2000) Mehner, K.; Wagner, A., *Ablaufvisualisierung für nebenläufige Java-Programme mit UML-Interaktionsdiagrammen*, Tagungsband GI-Workshop „Softwarevisualisierung“, 2000

Meier/Myrach (2004) Meier, A.; Myrach, T. (Hrsg.), *IT-Servicemanagement*, HMD – Praxis der Wirtschaftsinformatik, Heft 237, Heidelberg, 2004

Menge (2001) Menge, M., *Ein Beitrag zur Beherrschung der Variantenvielfalt in der Einzel- und Kleinserienfertigung komplexer Produkte*, Braunschweig, 2001

Meyer et al. (1969) Meyer, A.; Seibert, G.; Wendelberger, E. (Hrsg.), *Enzyklopädie 2000 – Das moderne Farblexikon in Folgen*, Stuttgart, 1969

Mintzberg (1994) Mintzberg, H., *The Rise and Fall of Strategic Planning*, New York, 1994

Mittal/Frayman (1989) Mittal, S.; Frayman, F., *Towards a generic model of configuration tasks*. In: Kaufman, M. (Hrsg.), *Proceedings of the 11th IJCAI*, San Mateo, CA, 1989, S. 1395-1401

Moll et al. (2004) Moll, K.-R.; Broy, M.; Pizka, M.; Seifert, T.; Bergner, K.; Rausch, A., *Erfolgreiches Management von Software-Projekten*. In: Informatik Spektrum, Band 27, Heft 5, Oktober 2004, S. 419-432

Moormann (2004) Moormann, J., *Transformation des Bankensektors*. In: Moormann/Fischer (2004), S. 1-17

Moormann/ Wölfing (1999) Moormann, J.; Wölfing, D., *Auf dem Weg zur neuen Informatik*. In: Die Bank, Nr. 7, S. 462-467

Moormann/Fischer (2004) Moormann, J.; Fischer, Th. (Hrsg.), *Handbuch Informationstechnologie in Banken*, 2. Auflage, Wiesbaden, 2004

Müller (1982) Müller, K., *Kostensenken in der technischen Abwicklung durch Standardisieren*, Baden (Schweiz), 1982

N

Neumann (2004) Neumann, F., *IT-Gesamtbankarchitektur am Beispiel der Sparkassen Informatik*. In: Moormann/Fischer (2004), S. 95-109

Niemann (2005) Niemann, K.-D., Von der Unternehmensarchitektur zur IT-Governance, Wiesbaden, 2005

Ninck et al. (2002) Ninck, A.; Bürki, L.; Hungerbühler, R.; Mühlemann, H., *Systemik – Integrales Denken, Konzipieren und Realisieren,* Zürich, 2002

Ninck/Büsser (2003) Ninck, A.; Büsser, M., BrainSpace – Problemlösung durch virtuelle Kollaboration. In: Szwillus, G.; Ziegler, J. (Hrsg.): *Mensch & Computer 2003: Interaktion in Bewegung*, Stuttgart, 2003, S. 77-86

Nippa (1999) Nippa, M., *Risikoverhalten von Managern bei strategischen Unternehmens-Entscheidungen*, Quelle: http://www.wiwi.tu-freiberg.de/up/download/publikationen/ap_33_1999.pdf, Abruf am 16.03.2005

Nogueria/ Reinhard (2000) Nogueira, A. R. R.; Reinhard, N., *Strategic IT Management in Brazilian Banks*, Proceedings of the 33rd Annual Hawaii International System Sciences, 2000.

Norman (1988) Norman, D., *The Psychology of Everyday Things*, New York, 1988

O

o. V. (2001) o.V., *Unternehmensbewertung - IT steigert den Firmenwert*. In: http://www.cio.de/markt/806153/index.html, zugegriffen am 03.01.2007.

OGC (1999) Office of Government Commerce (OGC) (Hrsg.), *Security Management. IT Infrastructure Library (ITIL)*, Norwich, 1999

OGC (2000) Office of Government Commerce (OGC) (Hrsg.), *Service Support. IT Infrastructure Library (ITIL)*, Norwich, 2000

OGC (2001) Office of Government Commerce (OGC) (Hrsg.), *Service Delivery. IT Infrastructure Library (ITIL)*, Norwich, 2001

OGC (2002a) Office of Government Commerce (OGC) (Hrsg.), *ICT Infrastructure Management. IT Infrastructure Library (ITIL)*, Norwich, 2002

OGC (2002b) Office of Government Commerce (OGC) (Hrsg.), *Application Management. IT Infrastructure Library (ITIL)*, Norwich, 2002

OGC (2002c) Office of Government Commerce (OGC) (Hrsg.), *Planning to Implement Service Management. IT Infrastructure Library (ITIL)*, Norwich, 2002

OGC (2003) Office of Government Commerce (OGC) (Hrsg.), *The Business Perspective. IT Infrastructure Library (ITIL),* Norwich, 2003

OMG (2004) OMG, *Unified Modeling Language (UML) Specification: Infrastructure*, Version 2.0, November 2004

Östereich (1998) Östereich, B., *Objektorientierte Softwareentwicklung*, 4. Auflage, München/Wien, 1998

Ott (1994) Ott, H. J., *Das „ingenieurmäßige“ am Software Engineering*. In: GI

Softwaretechnik-Trends: Mitteilungen der Fachgruppen „Software-Engineering“ und „Requirements-Engineering“, Band 14, Heft 1, Februar, 1994

P

Pahl/Beitz (1997) Pahl, G.; Beiz, W., *Konstruktionslehre: Methoden und Anwendung*, Berlin, 1997

Parnas (1972) Parnas, D., *On the Criteria to be Used in the Decomposition of Systems into Modules*. In: Communications of the ACM, Vol. 15, No. 2, December, 1972, S. 1053-1058

Passardi (1994) Passardi, A., *Bank-Management und Bank-Kostenrechnung: Die Bankkostenrechnung als Instrument für strategische Führung und Controlling*, 2. Auflage, Bern/Stuttgart/Wien, 1994

Patzak (1982) Patzak, G., *Systemtechnik – Planung komplexer innovativer Systeme*, Berlin, 1982

Paulk et al. (1993) Paulk, M. C.; Curtis, B.; Chrissis, M. B.; Weber, Ch. V., *Capabilty Maturity Model for Software, Version 1.1*. Software Engineering Institute, CMU/SEI-93-TR-24, Pittsburgh, 1993

Pelikan (1976) Pelikan, T., *Automated Design Engineering - Das System ADE-77*. In: Konstrukteur, Nr. 3/76, 1976

Penzel (2004) Penzel, H.-G., *Architekturmanagement aus Sicht einer Großbank*. In: Moormann/Fischer (2004), S. 111-130

Petrovic (1994) Petrovic, O., *Lean Management und informationstechnologische Potentialfaktoren*. In: Wirtschaftsinformatik 36, Nr. 6, 1994, S. 580-590

Phillips (2002) Phillips, M., *CMMI Version 1.1: What has Changed?*. In: Crosstalk, Februar 2002, S. 4-6

Piech (2004) Piech, F., *Auto. Biographie*, München, 2004

Piller (2000) Piller, F. T., *Mass Customization*, Wiesbaden, 2000

Piller/Waringer (1999) Piller, F. T.; Waringer, D., *Modularisierung in der Automobilindustrie – neue Formen und Prinzipien*, Aachen, 1999

Porter (1980) Porter, M. E., *Competitive Strategies. Techniques for analysing Industries and Competitors*, New York, 1980

Porter (1996) Porter, M. E., *What is Strategy?* In: Harward Business Review, November-Dezember 1996, S. 61-88

Porter/Millar (1985) Porter, M. E.; Millar, V. E., *How Information Gives You Competitive Advantage*. In: Harward Business Review, Nr. 4, Juli-August, 1985, S. 149-160

Puppe (1993) Puppe, F., *Systematic Introduction to Expert Systems*, Berlin, 1993

Q

Quinn (1996) Quinn, J. B., *Strategies for Change*. In: Minzberg, H.; Quinn, J. B., *The Strategy Process*, Upper Sddle River, 1996, S. 3-17

R

Rapp (1999) Rapp, T., *Produktstrukturierung*, Wiesbaden, 1999

Rausch/Rothe (2004) Rausch, K.; Rothe, A., *Wertorientiertes IT-Kostenmanagement*. In: Dietrich/Schirra (2004), S. 83-102

Rauterberg (1989) Rauterberg, M., *Über das Phänomen: "Information"*, Vortrag auf dem KI-Workshop der GMD, F3-XPS vom 16. - 18.11.1988 in Sankt Augustin

Rechtin/Maier (1997) Rechtin, E.; Maier, M., *The art of systems architecting*, Florida, 1997

Robak et al. (2002) Robak, S.; Franczyk, B.; Politowicz, K., *Extending the UML for Modelling Variability for System Families*. In: Int. J. App. Math. Computer Science, Vol. 10, No. 2, 2002, S. 285-298

Rohloff (2005) Rohloff, M., *Enterprise Architecture - Framework and Methodology for the Design of Architectures in the Large*. In: Proceedings of the 13th European Conference on Information Systems (ECIS), Regensburg, 2005

Rommel et al. (1993) Rommel, G.; Brück, F., Diederichs, R.; Kempis, R.-D.; Kluge, J., *Einfach überlegen*, Stuttgart, 1993

Rössler (2005) Rössler, A.; Spiegelhoff, A.; Klein, E., *Kostensenkung durch Software-Offshoring. Wunsch und Wirklichkeit*. In: die bank, Nr. 3, 2005, S. 62-65

Rothe (2004) Rothe, A., *Systematische Wiederverwendung von Softwarekomponenten bei Finanzdienstleistern*, Dissertation, Universität Stuttgart, 2004

Rumbaugh et al. (1991) Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorenson, W., *Object-Oriented Modelling and Design*, Englewood Cliffs, 1991

S

Sabin/Freuder (1996) Sabin, D.; Freuder, E. C., *Configuration as composite constaint satisfaction*. In : Luger, G. F. (Hrsg.), *Proc. Artificial Intelligence and Manufacturing Research Planning Workshop*, Albuquerque, NM, 1996, S. 153-161

SAP (2002) SAP AG, *mySAP™ Banking. Operatives Bankgeschäft*, White Paper, Walldorf, 2002

SAP (2005) SAP AG, *EMEA Retail Banking Study 2005*, Studie, Walldorf, 2005

Scheer (1993) Scheer, A. W., *Betriebs- und Wirtschaftsinformatik*. In: Wittmann, W. (Hrsg.), *Handwörterbuch der Betriebswirtschaft*, Band 2, Stuttgart, 1993, S. 390-408

Scheller (2006) Scheller, T., *Bewertung von Einsatz und Nutzen serviceorientierter Architekturen im Unternehmen*, Diplomarbeit, Lehrstuhl für Wirtschaftsinformatik, TU München, 2007

Schick/Binder Schick, M.; Binder, M., *Sicherstellen der rechtzeitigen Teileverfüg-*

(1998) *barkeit durch Problemmanagement zum Serienanlauf – dargestellt am Beispiel der A-Klasse*. In: Horváth, P.; Fleig, G., *Integrationsmanagement für neue Produkte*, Stuttgart, 1998, S. 273-297

Schlingheider (1994) Schlingheider, J., *Methodik zur Entwicklung rechnerunterstützter Konfigurationssysteme*, München, 1994

Schlote et al. (1998) Schlote, S.; Deysson, C.; Viehöver, U., *Ende der Legende*. In: Wirtschaftswoche, Ausgabe Nr. 10, 1992, S. 160-168

Schönung (2004) Schönung, M., *Komplexität in Unternehmen*,

Schuh/Schwenk (2001) Schuh, G.; Schwenk, U., *Produktkomplexität managen. Strategien, Methoden, Tools*, München/Wien, 2001

Schwinn/Winter (2005) Schwinn, A.; Winter, R., *Entwicklung von Zielen und Messgrößen zur Steuerung der Applikationsintegration*. In Ferstl O. K.; Sinz E. J; Eckert S.; Isselhorst, T. (Hrsg.), Wirtschaftsinformatik 2005, Heidelberg, 2005, S. 587–606

SEI (2002) Carnegie Mellon University, Software Engineering Institute (SEI): *Capability Maturity Model Integration (CMMI), Version 1.1. CMMI for Software Egeneering, Staged Representation*. CMU/SEI-2002-TR-029, ESC-TR-2002-029, Pittsburgh, 2002

Seiffert (1992) Seifert, H., *Einführung in die Wissenschaftstheorie – Handlungstheorie, Modallogik, Ethik, Systemtheorie*, Band 3, 2. Auflage, München, 1992

Simmons et al. (1996) Simons, M.; Creps, D.; Levine, L.; Allemang, D., *Organization Domain Modeling (ODM) Guidebook, Version 2.0*, Informal Technical Report from STARS, STARS.VCA205/001/00, Juni, 1996

Sinn et al. (2005) Sinn, W.; Dayal, R.; Pitman, D.; Luther, L.-U.; Grasshoff, G.; Herbeck, Th., *Succeeding with Growth – Creating Value in Banking 2005*, Studienpapier der Boston Consulting Group, München, 2005

Sinz (2003) Sinz, C., *Verifikation regelbasierter Konfigurationssysteme*, Dissertation, Eberhard-Karls-Universität, Tübingen, 2003

SPICE (2004) International Organisation for Standardisation (ISO), ISO 15504, Genf, 2004

Spiewak (1999) Spiewak, M., *Der schmale Grad des Erfolgs*. In: Die Zeit, Ausgabe 35, 1999

Spur (1996) Spur, G. (1996), *Ingenieurwissenschaften*. In: Kern, W. (Hrsg.), *Handwörterbuch der Produktionswirtschaft*, 2. Auflage, Stuttgart, 1996, S. 728-735

Stahlknecht/ Hasenkamp (1997) Stahlknecht P.; Hasenkamp U., *Einführung in die Wirtschaftsinformatik*, 8. Auflage, Berlin et al., 1997

Stahlknecht/Hasenkamp (2005)	Stahlknecht P.; Hasenkamp U., *Einführung in die Wirtschaftsinformatik*, 11. Auflage, Berlin et al., 2005
Steiger/Lippmann (1999)	Steiger, Th.; Lippmann, E. (Hrsg.), *Handbuch angewandte Psychologie für Führungskräfte. Führungskompetenz und Führungswissen*, Band 1, Heidelberg, 1999
Stelzer (1998)	Stelzer, D., *Möglichkeiten des prozeßorientierten Software-Qualitätsmanagements*, Habilitationsschrift vorgelegt an der Wirtschafts- und Sozialwissenschaftlichen Fakultät der Universität zu Köln, Köln, 1998
Stoßberg/Hellingrath (2002)	Stoßberg, T.; Hellingrath, B., *OTD-Simulation. Ein mächtiges Gestaltungswerkzeug für die VW-Logistik*. In: VDI-Berichte, Düsseldorf, Band 1698, 2002, S. 79-91
Streuffert/ Streuffert (1978)	Streuffert, S.; Streuffert S. S., *Behaviour in the complex environment*, Washington, 1978
Strunz (1998)	Strunz, H., *Investition*. In: Krabbe. E. (Hrsg.), *Leitfaden zum Grundstudium der Betriebswirtschaftslehre*, 6. Auflage, Gernsbach, 1998, S. 287-362
Stützle (2002)	Stützle, R., *Wiederverwendung ohne Mythos: Empirisch fundierte Leitlinien für die Entwicklung wieder verwendbarer Software*, Dissertation, TU München, 2002
Sullivan et al. (1999)	Sullivan, K.; Chalasani, P.; Jha, S.; Sazawal, V., *Software Design as an Investment Activity: A Real Options Perspective*. In: Trigeorgis, L. (Hrsg.), *Real Options and Business Strategy: Applications to Decision Making*, London, 1999, S. 215--261
Svahnbrg/Bosch (1999)	Svahnberg, M.; Bosch, J., *Characterizing Evolution in Product-Line Architectures*, Proceedings of the 3rd annual IASTED International Conference on Software Engineering and Applications (SEA'99), Oktober, 1999
Szyperski/ Eschenröder (1983)	Szyperski, N.; Eschenröder, G., *Information-Ressource-Management*. In: Kay, R. (Hrsg.), *Management betrieblicher Informationsverarbeitung*, München, 1983
T	
Taubner (2005)	Taubner, D., *Software-Industrialisierung*. In: Informatik Spektrum, Band 28, Heft 4, August 2005, S. 292-296
Thaler (1999)	Thaler, K., Supply Chain Management. Prozessoptimierung in der logistischen Kette. Bremen, 1999
Tiihonen et al. (1999)	Tiihonen, J.; T. Lethonen; Soinien, T.; Pulkkinen, A.; Sulonen, R.; Ritahuhta, A, *Modelling Configurable Product Families*, International Conference on Engineering Design, ICED-99, München, August, 1999
U	

Ulrich (1984) Ulrich, H., *Management*, Bern, 1984

Ulrich/Hill (1976) Ulrich, P.; Hill, W., *Wissenschaftstheoretische Grundlagen der Betriebswirtschaftslehre (Teil 1)*. In: Wirtschaftswissenschaftliches Studium, 5. Jahrgang, Heft 7, 1976

Ungeheuer (1985) Ungeheuer, U., *Produkt- und Montagestrukturierung: Methodik einer anforderungsgerechten Produkt- und Montagestruktur für komplexe Erzeugnisse der Einzel- und Kleinserieproduktion*, Düsseldorf, Fortschritt-Berichte VDI, Reihe 2, 1985

V

Vogt et al. (2000) Vogt, W.; Fremmer, J.; Buser, F., *Nutzen ohne Frust*, Basel, 2000

VW (2004) Volkswagen AG, *Umweltbericht 2003/2004*, erhältlich unter: http://www.econsense.de/_PUBLIKATIONEN/_PUBLIKATIONEN_MITGLIEDER/images/Volkswagen/VW_U_Bericht_03_04.pdf, Abruf am: 01.03.2007

W

Ward/Peppard (2002) Ward, J.; Peppard, J., *Strategic Planning for Information Systems*, Chichester, UK, 2002

Weber (1992) Weber, H., *Die Softwarekrise und ihre Macher*, Berlin/Heidelberg, 1992

Webster (1995) Webster, B., *Pitfalls of Object-Oriented Development: A Guide to the Wary and the Enthusiastic*, New York, 1995

Weill/Woodham (2002) Weill, P.; Woodham, R., *Don't just Lead, Govern: Implementing Effective Governance*, Working Paper 236, Cambridge Sloan School of Management, Cambridge, 2002

Weimer/ Wisskirchen (1999) Weimer, T.; Wisskirchen, C., *Sechs Thesen zur Fusionswelle im Bankenbereich*. In: Die Bank, Nr. 11, 1999, S. 758-764

Westphal (2005) Westphal, R., *Industrialisierung der Softwareentwicklung*. In: Objekt Spektrum, Heft 3, 2005, S. 32

Wielinga/Schreiber (1997) Wielinga, B. J.; Schreiber, A. Th., *Configuration Design Problem Solving*. In: AI and Design, IEEE Expert, Vol. 12, No. 2, 1997, S. 49-56

Wildemann (1998) Wildemann, H., *Komplexitätsmanagement durch Prozeß- und Produktgestaltung*. In: Adam, D. (Hrsg.), *Komplexitätsmanagement*, Schriften zur Unternehmensführung, Band 61, Wiesbaden, 1998, S. 47-68

Wöhe (1996) Wöhe, G., *Einführung in die Allgemeine Betriebswirtschaftslehre*, 19. Auflage, München, 1996

Womack et al. (1996) Womack, J. P.; Jones, D. T.; Roos, D.; Stotko E. C., *Die zweite Revolution in der Autoindustrie Konsequenzen aus der weltweiten Studie aus dem Massachusetts Institute of Technology*, 6. Aufla-

ge, Frankfurt, 1992

Y

Yin (1989) Yin, R. K., *Case Study Research. Design and Methods*, Thousand Oaks, 1989

Yu/MacCallum (1995) Yu, B.; MacCallum, K., *Modelling of Product Configuration Design and Management by Using Product Structure Knowledge*. In: International Workshop on Knowledge Intensive CAD, Finland, 1995

Z

Zachman (1987) Zachman, J. A., *A Framework for Information Systems Architecture*. In: IBM Systems Journal, Vol. 26, No. 3, 1987

Zachman (2001) Zachman, J. A, *You Can't 'Cost-Justify' Architecture*, DataToKnowledge Newsletter, Vol. 29, No. 3, (Business Rule Solutions LLC, May/June 2001)

Zimmermann (1995) Zimmermann, V., *Quality Function Deployment (QFD) im Entwicklungsprozess. Konzepte, Modelle, Methoden und Hilfsmittel*, Kaiserslautern, 1995

Zink (1995) Zink, K. J., *TQM als integratives Managementkonzept*, München-Wien, 1995