Systemische Risiken in der Wirtschaftsinformatik

Dissertation

der Wirtschaftswissenschaftlichen Fakultät

der Universität Augsburg

zur Erlangung des Grades eines Doktors

der Wirtschaftswissenschaften

(Dr. rer. pol.)

vorgelegt

von

Peter Bartmann
(Diplom-Informatiker Univ.)

Augsburg, 02. November 2011

Erstgutachter:	Prof. Dr. Hans Ulrich Buhl
Zweitgutachter:	Prof. Dr. Robert Klein
Vorsitzender der mündlichen Prüfung:	Prof. Dr. Marco C. Meier
Datum der mündlichen Prüfung:	29. November 2011

„Niemand bestreitet die Wunder der modernen Wissenschaft.
Jetzt wäre es an der Zeit, dass sie auch für ihre Monster die Verantwortung übernimmt.“

Jakob von Uexküll

Inhaltsverzeichnis

Anmerkung: Eine fortlaufende Seitennummerierung wird pro Kapitel beziehungsweise pro Unterkapitel des jeweiligen Beitrags vorgenommen. Ein Literaturverzeichnis sowie die Anhänge werden jeweils am Ende eines jeden Beitrags aufgeführt.

Verzeichnis der Beiträge

In dieser Dissertation werden die folgenden Beiträge vorgestellt:

B.1 Bartmann P, Buhl HU, Hertel M (2009) Ursachen und Auswirkungen der Subprime-Krise. In: Informatik-Spektrum, 32, 2, S. 127-145; in: B. Nietert ed., Die Eskalation der Finanz- zur Wirtschaftskrise, Fritz Knapp Verlag, Frankfurt am Main (Deutschland), 2011, S. 11-45.
VHB-JOURQUAL 1: *6,37 Punkte (Kategorie C)*
VHB-JOURQUAL 2.1: *4,90 Punkte (Kategorie E)*
WI-Orientierungslisten: *Kategorie B*

B.2 Bartmann P (2009): Die Verantwortung der Wirtschaftsinformatik für die Finanzmarktkrise. In: Informatik-Spektrum, 32, 2, S. 146-152; in: B. Nietert ed., Die Eskalation der Finanz- zur Wirtschaftskrise, Fritz Knapp Verlag, Frankfurt am Main (Deutschland), 2011, S. 46-59.
VHB-JOURQUAL 1: *6,37 Punkte (Kategorie C)*
VHB-JOURQUAL 2.1: *4,90 Punkte (Kategorie E)*
WI-Orientierungslisten: *Kategorie B*

B.3 Bartmann P (2012): Technische Softwaredokumentation unter ökonomischen Gesichtspunkten – Ein formal-deduktiver Ansatz. In: Zeitschrift für Betriebswirtschaft, 82, 3, S. 243-274.
VHB-JOURQUAL 1: *7,37 Punkte (Kategorie B)*
VHB-JOURQUAL 2.1: *7,01 Punkte (Kategorie B)*
WI-Orientierungslisten: *Kategorie B*

B.4 Bartmann P, Meier MC, Lindermeir M (2011): Monetary Valuation of Alternative Modular Software Structures Considering both Technical and Factual Dependencies – A Formal Deductive Approach.

B.5 Jürjens J, Schreck J, Bartmann P (2008): Model-based Security Analysis for Mobile Communications. In: Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), Leipzig (Germany). ACM, 2008; p. 683-692.

I Einleitung

Der Verlauf der seit Juli 2007 virulent gewordenen und bis heute andauernden Finanzkrise, welche sich spätestens mit dem Übergreifen auf die Realwirtschaft (o.V. 2008a) und mit ihren Auswirkungen auf die Stabilität ganzer Staaten wie Island (o.V. 2008b), Portugal, Italien, Irland, Griechenland, Spanien (o.V. 2011) und sogar die USA (Dash 2011) zur Weltwirtschaftskrise und internationalen Schuldenkrise entwickelte, hat systemische Risiken verstärkt in den Fokus von Wissenschaft und Praxis gerückt. Dabei finden sich in der existierenden Literatur unterschiedliche Definitionen des Begriffs *systemisches Risiko*. So benennt beispielsweise Davis (2003) im Kontext der Finanzwirtschaft ein systemisches Risiko als „*entailing heightened risk of a financial crisis - a major collapse of the financial system, entailing inability to provide payments services or to allocate credit to productive investment opportunities*". Während sich diese Definition überwiegend auf das Schadensausmaß bezieht, führen zum Beispiel Rochet und Tirole (1996) die Ausbreitung der finanziellen Notlage eines Akteurs auf andere Akteure, welche durch finanzielle Transaktionen mit dem ursprünglichen Akteur verbunden sind, als wesentliches Charakteristikum an: „*Systemic risk refers to the propagation of an agent's economic distress to other agents linked to that agent through financial transactions.*" Bemerkenswerterweise bilden sowohl ein ungewöhnlich hohes Schadensausmaß als auch ein derartiger Ausbreitungseffekt in einem stark vernetzten Finanzsystem zwei prägnante Merkmale der aktuellen Krise. So gilt insbesondere der amerikanische Immobilienkreditmarkt im Segment der einkommensschwachen Kunden mit problematischer Kredithistorie (*subprime*) als Ausgangspunkt der Finanzkrise. Während der Subprime-Markt jedoch zum Ende des Jahres 2006 mit einem Volumen von 1,24 Billionen US-Dollar (o.V. 2009a) seinen Höhepunkt erreichte, bezifferte die Commerzbank die weltweiten Verluste durch die Finanzkrise bereits im Jahre 2009 auf über 10 Billionen US-Dollar (o.V. 2009b). Daneben verdeutlicht beispielsweise die Einstufung der *Hypo Real Estate Group* (HRE) als systemrelevante Bank (o.V. 2009c) und die damit als zwingend notwendig begründete Rettung, wie stark das Risiko eines Unternehmens nicht nur durch sich selbst, sondern auch durch seine Abhängigkeiten im System bestimmt wird.

Für den Ausbruch und den Verlauf der Krise spielen Informations- und Kommunikationssysteme (IKS) eine entscheidende Rolle. Einerseits bildet die Informationstechnologie (IT) die Grundlage für die hochgradige und internationale Vernetzung der Finanzdienstleistungsbranche. Ein Großteil der Finanztransaktionen wird heutzutage mit IT-Unterstützung abgewickelt, so dass ohne den umfassenden Einsatz von IT vermutlich

nicht eine derart starke Vernetzung zwischen den Akteuren existieren würde, über die sich entsprechende Risiken in dem seit Juli 2007 beobachtbaren Maße ausbreiten könnten. Andererseits werden IKS unter anderem auch im Risikomanagement eingesetzt. Sowohl das Risikomanagement vieler Finanzdienstleister als auch die Erstellung der Ratings der strukturierten Wertpapiere basiert auf finanzmathematischen Modellen, welche aufgrund ihrer enormen Komplexität erst durch ihre Implementierung in IKS sinnvoll einsetzbar werden.

Grundsätzlich werden IKS unter anderem in Wirtschaft und Verwaltung zur Unterstützung, Steuerung und Automatisierung von Wertschöpfungsprozessen mittels Informationen sowie zur Befriedigung der Informationsnachfrage von Aufgabenträgern als zentraler Gegenstand der Wirtschaftsinformatik formuliert (WKWI 1994, WKWI und FB GI WI 2011). Information bildet eine elementare Grundlage für Entscheidungen in Unternehmen und einen zentralen Produktionsfaktor im betrieblichen Leistungserstellungsprozess (Krcmar 2005). Insbesondere IT-basierte Entscheidungsunterstützungssysteme (EUS) als spezielle Form von IKS dienen in diesem Kontext der Unterstützung von Entscheidungsträgern der Unternehmensführung bei der Entscheidungsfindung im schlecht strukturierten Aufgabenumfeld (Gluchowski et al. 2008), indem diese Informationen und Entscheidungsvorschläge präsentieren (Mertens und Meier 2009). Eben solche EUS, für deren Erforschung, Konzeption, Entwicklung, Nutzung und Bewertung die Wirtschaftsinformatik verantwortlich zeichnet (WKWI 1994, WKWI und FB GI WI 2011), finden unter anderem im Rahmen des Risikomanagements zahlreicher Finanzdienstleister und für die Erstellung von Ratings Verwendung. Zugleich führt beispielsweise Rudolph (2008) technische Mängel im Risikomanagement der Banken sowie inhärente Unzulänglichkeiten im Rating von strukturierten Finanzprodukten als häufig genannte Ursache der Krise an. Darüber hinaus wird auch eine fehlende Transparenz über die tatsächlichen Risiken der strukturierten Wertpapiere (Hamerle und Plank 2010) und über die von den Akteuren in ihren Büchern gehaltenen Risiken (Rudolph 2008) als zentrale Ursache genannt. Stellt man nun dieser Analyse der Ursachen der Finanzkrise den oben dargestellten Anspruch der Wirtschaftsinformatik gegenüber, kann und muss die Wirtschaftsinformatik insbesondere als Interdisziplin zwischen der Betriebswirtschaft und der Informatik als „Werkzeugmacher“ für die Finanzwirtschaft und für die der Krise ursächlichen Systeme und Strukturen eingestuft werden. Folglich ergibt sich im ersten Schritt die Frage nach der Rolle und Verantwortung der Wirtschaftsinformatik für die im Zuge der Finanzkrise eingetretenen systemischen Risiken. Im zweiten Schritt erscheint darüber hinaus ebenfalls eine Diskussion über den möglichen Beitrag der Wirtschaftsinformatik zur Vermeidung von systemischen Risiken sowohl in der Finanzbranche als

auch in anderen Branchen lohnenswert.

Die Wirtschaftsinformatik sieht sich jedoch nicht nur in ihrer Rolle als „Werkzeugmacher“ mit systemischen Risiken konfrontiert. So bilden beispielsweise Softwaresysteme insbesondere bei komponenten- oder objektorientierten Entwicklungsparadigmen ein vernetztes System aus einzelnen Komponenten, die über verschiedenartige Abhängigkeiten wie Datenaustausch etc. in Verbindung stehen. Legt man nun eine allgemeinere und von der Finanzbranche unabhängige Definition des Begriffs *systemisches Risiko* an und versteht dieses als eine Situation, in der „*sich ein auf ein Element eines Systems einwirkendes Ereignis aufgrund der dynamischen Wechselwirkungen zwischen den Elementen des Systems auf das System als Ganzes negativ auswirken kann*“ oder in der „*sich aufgrund der Wechselwirkungen zwischen den Elementen die Auswirkungen mehrerer auf einzelne Elemente einwirkender Ereignisse so überlagern, dass sie sich auf das System als Ganzes negativ auswirken können*“ (Romeike und Hager, 2009), so können auch innerhalb einer Software systemische Risiken eintreten. Ein Beispiel dafür bilden nachträgliche Änderungen eines Anwendungssystems im Rahmen der Softwarewartung, welche beispielsweise aufgrund eines unzureichenden Verständnisses der zuständigen Entwickler bezüglich des Softwaresystems fehlerhaft und inkonsistent zur ursprünglichen Struktur der Software sein können (Krishnan et al. 2004). Die nachträgliche Modifikation einer Softwarekomponente kann zur Folge haben, dass angrenzende Komponenten nicht mehr fehlerfrei mit der ursprünglichen Komponente interagieren und folglich ebenfalls einer Änderung bedürfen. Diese Änderungen können wiederum neue Änderungen erfordern und so weiter (Beszédes et al. 2007). Aufgrund dieses Ausbreitungseffekts von Fehlern auf angrenzende Komponenten über deren Abhängigkeiten kann ein kleiner Fehler zu vielen weiteren Fehlern führen, welche sich über das gesamte Softwaresystem verteilen (Eaddy et al. 2008). Als Ursache für eine derartige Fehlerausbreitung bestätigt die existierende Literatur vor allem das Fehlen von Informationen über die Funktionalitäten der Softwarekomponenten und über deren Interaktionen untereinander beispielsweise in Form einer technischen Programmdokumentation (Arisholm et al. 2006, Dzidek et al. 2008, Prechelt et al. 2002), in abstrahierender Analogie zur Finanzmarktkrise also die Intransparenz der Elemente eines Systems und deren Abhängigkeiten. Gleichzeitig bildet damit die Erstellung einer adäquaten Programmdokumentation aber auch eine präventive Maßnahme zur Reduzierung der Auswirkungen einer derartigen Fehlerausbreitung. Für jedes Softwareprojekt ergibt sich folglich die Frage, in welcher Höhe eine Investition in die Erstellung einer zweckmäßigen Programmdokumentation während der Softwareentwicklung ökonomisch sinnvoll ist.

Neben der Verfügbarkeit einer zweckmäßigen Programmdokumentation identifiziert die

existierende Literatur mit der Anzahl der Abhängigkeiten einer Softwarekomponente zu anderen Komponenten einen weiteren Einflussfaktor auf die Fehleranfälligkeit von Wartungsmaßnahmen (Briand et al. 2002, Gyimóthy et al. 2005, Singh et al. 2010). Ein ähnlicher Zusammenhang, nämlich zwischen der Stärke der Abhängigkeiten von Softwaremodulen (*Kopplung*, engl. *coupling*) und der Anzahl der nachträglichen Quellcode-Modifikationen als Indikator für die Fehleranzahl, wird von Kemerer (1995) bestätigt. Da sich Fehler im Speziellen und die Notwendigkeit von nachträglichen Änderungen insbesondere über die entsprechenden Abhängigkeiten auf angrenzende Softwarekomponenten ausbreiten (Baldwin und Clark 2000), bildet neben der Erstellung einer adäquaten Programmdokumentation die Entwicklung einer modularen Softwarestruktur mit möglichst loser Kopplung der Module eine weitere präventive Maßnahme zur Reduzierung der während der Softwarewartung auftretenden Fehler- beziehungsweise Änderungsausbreitung (Marinescu 2002, Yu et al. 2002, Henry und Kafura 1981, Kemerer 1995). In diesem Zusammenhang gilt es jedoch zu beachten, dass die im Zuge der Wartung erforderlichen Anpassungen eines Softwaresystems überwiegend aus Änderungen der fachlichen Anforderungen resultieren (Krishnan et al. 2004). Beruhen nun mehrere Funktionalitäten einer Software auf derselben fachlichen Anforderung, so bedürfen diese Funktionalitäten im Falle einer Änderung der entsprechenden Anforderung mit hoher Wahrscheinlichkeit einer gemeinsamen Modifikation. Für die Softwarewartung ist es also ungeachtet einer losen Kopplung ebenfalls von Vorteil, wenn sich diejenigen Funktionalitäten, welche mit hoher Wahrscheinlichkeit gemeinsam modifiziert werden müssen, auf möglichst wenige Module verteilen, so dass für die Anpassung eines Softwaresystems an geänderte Anforderungen die Modifikation einer entsprechend geringeren Anzahl an Modulen notwendig ist. Folglich müssen für die Bewertung alternativer Softwarearchitekturen in Bezug auf den zu erwartenden Wartungsaufwand neben den technischen Abhängigkeiten von Softwarekomponenten (Kopplung) auch fachliche Abhängigkeiten berücksichtigt werden. Überlappen sich die technischen und fachlichen Abhängigkeiten in einigen Fällen nicht vollständig, bildet die Entwicklung einer modularen Softwarearchitektur mit dem Ziel, die Auszahlungen für die nachgelagerte Wartung unter Berücksichtigung eines entsprechenden Ausbreitungseffekts von Fehlern und Änderungen zu reduzieren, eine nicht triviale Problemstellung. Für die Wirtschaftsinformatik ergeben sich also ebenfalls im Rahmen der Softwareentwicklung und des Managements von Anwendungssystemen entsprechende Herausforderungen zur Verminderung systemischer Risiken.

Systemische Risiken besitzen jedoch nicht nur für einzelne Softwaresysteme eine hohe Relevanz. Spätestens seit dem umfassenden Zusammenschluss weltweit verteilter

Rechnernetzwerke durch das Internet bilden unzählige Großrechner, Workstations, Personal Computer etc. ein gigantisches, globales Netzwerk. Eben diese hochgradige Vernetzung stellt eine große Herausforderung insbesondere für die IT-Sicherheit dar. So kann sich ein Angriff, dessen Ursprung in einer lokalen Schwachstelle im Netzwerk liegt, beispielsweise über das Internet auf zahlreiche weitere Rechner ausbreiten. Als Beispiel seien an dieser Stelle Viren und Trojanische Pferde genannt. Sobald ein derartiges Schadprogramm einen Rechner infiziert hat, dient dieser in der Regel als Ausgangspunkt für die weitere Ausbreitung (Sommer und Brown 2011). Das mögliche Ausmaß der Ausbreitung solcher Schadprogramme illustriert das sogenannte *Mariposa Botnetz*, welches sich aus ca. 13 Millionen infizierter Rechner zusammensetzte und unter anderem für verteilte Denial-of-Service (DDoS) Angriffe genutzt wurde (McMillan 2010). Darüber hinaus können Schadprogramme enorme Schäden verursachen. Alleine der Computervirus „*I love you*“ verursachte im Jahr 2000 einen geschätzten Schaden von 2,6 Milliarden US-Dollar (o.V. 2001).

Dabei können sich derartige und viele weitere Angriffsarten nicht nur auf die betroffenen Rechnernetze selbst negativ auswirken, sondern auch auf andere, mit den betroffenen Rechnernetzen gekoppelte Netze. So zeigen beispielsweise Buldyrev et al. (2010), dass interdependente, d.h. wechselseitig abhängige Netze bereits auf kleine Störungen sehr anfällig reagieren können. Die Autoren führen als Beispiel Internet und Stromnetz an, da Rechnernetze Strom benötigen und Stromnetze wiederum der IT-Steuerung bedürfen. Darüber hinaus stellen derartige Schwachstellen auch für die Geschäftstätigkeit eines Unternehmens und sogar für ganze Branchen eine Bedrohung dar. So betonen beispielsweise Glaessner et al. (2002), dass der Ausfall des operativen Geschäfts einer Bank für einen bestimmten Zeitraum beispielsweise aufgrund eines DDoS-Angriffs, welcher eine Überlast bei den anvisierten Rechnern und damit ihren Ausfall herbeizuführen versucht, die Reputation nachhaltig schädigen und damit die Bonität der Bank gefährden kann. In diesem Zusammenhang verdeutlicht Lemieux (2003), dass derartige Bedrohungen für alle Akteure in einem System ein Risiko bedeuten und sogar von der Finanzbranche auf andere Wirtschaftssektoren übergreifen können. Obgleich IT-Sicherheitsrisiken oftmals als operationelles Risiko eingestuft werden (Faisst et al. 2007), sieht sich die Wirtschaftsinformatik folglich ebenfalls in diesem Kontext mit der Herausforderung des Managements systemischer Risiken gemäß obiger Definition konfrontiert.

1 Zielsetzung und Aufbau dieser Dissertationsschrift

Ziel dieser Dissertationsschrift ist es, einen Beitrag zum Management systemischer

Risiken aus der Perspektive der Wirtschaftsinformatik zu leisten. Dazu konzentriert sich das zweite Kapitel auf die Rolle der Wirtschaftsinformatik als „Werkzeugmacher“ und beschäftigt sich am Beispiel der Finanzkrise mit dem Zusammenhang zwischen dem Einsatz spezieller EUS unter anderem für das Risikomanagement von Finanzdienstleister und den systemischen Risiken in der Finanzwirtschaft. Das dritte Kapitel fokussiert dagegen das Management systemischer Risiken im Rahmen der Softwareentwicklung und des Managements von Anwendungssystemen. Den systemischen Risiken in Bezug auf Rechnernetze und Netzwerkarchitekturen widmet sich das vierte Kapitel. Abb. I-1 strukturiert die verfolgten Ziele und zeigt den Aufbau der Arbeit.

I	**Einleitung**
Ziel I.1:	Darstellung der Zielsetzung und des Aufbaus der Arbeit
Ziel I.2:	Fachliche Einordnung und Motivation der zentralen Forschungsfragen
II	**Systemische Risiken in der Finanzwirtschaft und die Rolle der IT am Beispiel der Finanzkrise (B.1, B.2)**
Ziel II.1:	Darstellung der für die Entstehung und die Entwicklung der Finanzkrise relevanten Verbriefungsstrukturen
Ziel II.2:	Aufzeigen der Ursachen, der Zusammenhänge und des Verlaufs der Finanzkrise
Ziel II.3:	Diskussion der Verantwortung der Wirtschaftsinformatik für die Entstehung und den Verlauf der Krise
Ziel II.4:	Ableitung zukünftiger Herausforderungen für die Wirtschaftsinformatik
III	**Systemische Risiken in Anwendungssystemen (B.3, B.4)**
Ziel III.1:	Deduktion neuer formaler Zusammenhänge zwischen dem Dokumentationsgrad einer Software und der im Zuge der Softwarewartung entstehenden Fehler
Ziel III.2	Deduktion neuer formaler Hypothesen bezüglich eines optimalen Dokumentationsgrads von Softwaresystemen
Ziel III.3	Entwicklung einer Methodik für die ökonomische Bewertung und den Vergleich von alternativen, modularen Softwarearchitekturen in Bezug auf die barwertigen Auszahlungen für fachlich getriebene Änderungen im Rahmen der Softwarewartung

IV	**Systemische Risiken in Rechnernetzen im Rahmen der IT-Sicherheit (Beitrag: „*Model-based Security Analysis for Mobile Communications*“) (B.5)**
Ziel IV.1:	Durchführung einer modellbasierten Sicherheitsanalyse von mobilen Kommunikationssystemen mit der UML-Erweiterung UMLsec
Ziel IV.2:	Entwicklung einer Methodik zur Analyse von Rechnernetzen auf unsichere Verbindungen und Datenflüsse als Erweiterung von UMLsec
V	**Fazit und Ausblick**
Ziel V.1:	Zusammenfassung der zentralen Ergebnisse
Ziel V.2:	Aufzeigen künftigen Forschungsbedarfs

Abb. I-1: Aufbau der Dissertationsschrift

2 Fachliche Einordnung und fokussierte Forschungsfragen

Bezogen auf die Ziele werden nun die Beiträge fachlich eingeordnet und deren zentrale Forschungsfragen motiviert.

- **Systemische Risiken in der Finanzwirtschaft und die Rolle der IT am Beispiel der Finanzkrise (B.1, B.2)**

  B.1: Ursachen und Auswirkungen der Subprime-Krise

  Ziel dieses Beitrags ist es, auf Basis der existierenden Literatur die relevanten Instrumente, Eigenschaften, Zusammenhänge und Strukturen sowie darauf aufbauend die grundlegenden Ursachen der internationalen Finanzkrise aufzuzeigen. Dazu werden zuerst die vor der Krise vorherrschenden Verbriefungstransaktionen in ihrer grundsätzlichen Struktur dargestellt. Diese Finanzinstrumente stellen die Verbindung zwischen den Kredit- und Kapitalmärkten her und bildeten somit die zentralen, ursächlichen Strukturen für das Eintreten der im Rahmen der Krise beobachtbaren systemischen Risiken. Die Betrachtung von Ertrags- und Risikopositionen erlaubt anschließend eine ökonomische Analyse der dargestellten Verbriefungstransaktionen. Auf dieser Basis werden die Ursachen und Wirkungszusammenhänge der Krise sowie der Krisenverlauf bis Ende des Jahres 2008 herausgearbeitet. Schließlich werden die für die Marktstrukturen charakteristischen Anreizsysteme als Grundlage des Verhaltens der beteiligten Akteure beschrieben. Insbesondere die der Finanzkrise ursächlichen Strukturen sowie die Wirkungszusammenhänge und der

Verlauf der Krise sollen in einer für Leser ohne Expertenwissen in diesem Fachbereich geeigneten Form dargestellt werden, so dass dieser Beitrag als Grundlage für die anschließende Diskussion der Rolle und Verantwortung der Wirtschaftsinformatik für die Finanzkrise dienen kann. Dabei werden unter anderem die folgenden Forschungsfragen adressiert:

1. Inwieweit bildeten die Strukturen der bis 2007 vorherrschenden Verbriefungstransaktionen und die damit verbundenen Ertrags- und Risikopositionen die Grundlage für die Entstehung und den Verlauf der Finanzkrise?
2. Welche Anreizsysteme förderten insbesondere das vor der Krise beobachtbare Verhalten der beteiligten Marktakteure?

B.2: Die Verantwortung der Wirtschaftsinformatik für die Finanzmarktkrise

Die Verbriefungsstrukturen, welche dem Ausbruch und dem Verlauf der Finanzkrise zugrunde liegen, basieren zu einem wesentlichen Teil auf finanzmathematischen Modellen. Deren Verwendung wird jedoch aufgrund ihrer teilweise enormen Komplexität und der großen Quantität der benötigten Informationen erst mit der Einbettung in entsprechende Entscheidungsunterstützungssysteme (EUS) als spezielle Form von IKS praktikabel. Damit besitzen diese finanzmathematischen Modelle beispielsweise für das Risikomanagement und für die Erstellung von Ratings sowie die zugehörigen EUS einen Einfluss auf die Entstehung und den Verlauf der Krise. Zumindest gelten technische Mängel im Risikomanagement, inhärente Unzulänglichkeiten der Ratings (Rudolph 2008) sowie die Intransparenz der durch die Verbriefungsstrukturen tatsächlich induzierten Risiken (Hamerle und Plank 2010) neben anderen als zentrale Ursachen für die Finanzkrise. Nun zählt gerade die Erforschung, Konzeption, Entwicklung, Nutzung und Bewertung derartiger EUS zu den Aufgaben der Wirtschaftsinformatik. Daher möchte dieser Beitrag die Verantwortung der Wirtschaftsinformatik für die im Rahmen der Krise beobachtbaren systemischen Risiken diskutieren sowie Lehren insbesondere für die IT-basierte Entscheidungsunterstützung ziehen. Folgende Forschungsfragen stehen dabei im Fokus:

1. Inwieweit ist die Wirtschaftsinformatik als Interdisziplin und als angewandte Wissenschaft gemäß ihrem Anspruch im Zuge der Finanzkrise ihrer Verantwortung gerecht geworden und in welchem Maße haben die Versäumnisse zur Entstehung der Finanzmarktkrise beigetragen?
2. Welche Lehren und zukünftige Herausforderungen können für die Wirtschaftsinformatik und insbesondere für die IT-basierte Entscheidungsunterstützung aus

ihrer Verantwortung für die Entstehung und den Verlauf der Krise abgeleitet werden?

- **Systemische Risiken in Anwendungssystemen (B.3, B.4)**

  B.3: Technische Softwaredokumentation unter ökonomischen Gesichtspunkten – Ein formal-deduktiver Ansatz

  Ein dynamisches Marktumfeld und sich ändernde Kundenanforderungen erfordern eine kontinuierliche Anpassung der Geschäftsprozesse und damit im Rahmen der Softwarewartung auch eine regelmäßige Adaption der im Unternehmen eingesetzten Anwendungssysteme. Nachträgliche Änderungen eines Softwaresystems können jedoch fehlerhaft und inkonsistent zur ursprünglichen Struktur der Software sein (Krishnan et al. 2004). Dabei können sich die aufgrund der nachträglichen Modifikation einer Softwarekomponente entstehenden Fehler auf angrenzende Komponenten über deren Abhängigkeiten ausbreiten (Beszédes et al. 2007), so dass ein einzelner Fehler zu einer Vielzahl weiterer, über das gesamte Softwaresystem verteilter Fehler führen kann (Eaddy et al. 2008). In diesem Zusammenhang bestätigen empirische Studien, dass die Verfügbarkeit einer adäquaten technischen Programmdokumentation zu einer geringeren Fehleranfälligkeit von nachträglichen Modifikationen führt (Arisholm et al. 2006, Dzidek et al. 2008, Prechelt et al. 2002), da diese zu einer besseren Verständlichkeit der Software beiträgt und damit die Wahrscheinlichkeit für inkonsistente Modifikationen durch das Wartungspersonal reduziert. Folglich kann mittels zusätzlicher Investitionen in eine zweckmäßige Programmdokumentation während der Softwareentwicklung eine Reduktion der Auszahlungen für die Behebung von Fehlern und Inkonsistenzen im Rahmen der Softwarewartung erreicht werden. Für jedes Softwareprojekt ergibt sich demnach die Frage, in welcher Höhe eine Investition in die Erstellung einer adäquaten Programmdokumentation ökonomisch sinnvoll ist. Dieses Entscheidungsproblem wurde wissenschaftlich bisher nur ungenügend behandelt, so dass in der existierenden Literatur entsprechende Zusammenhänge fehlen, welche als Grundlage für eine befriedigende Lösung dieser Problemstellung dienen können. Daher verfolgt dieser Beitrag auf Basis der überwiegend beschreibenden und empirisch geprägten Literatur und mittels eines formal-deduktiven Ansatzes die Generierung neuer Hypothesen im Sinne einer wissenschaftlichen Erklärung insbesondere bezüglich des Dokumentationsgrads einer Software und der im Rahmen der Softwarewartung entstehenden Fehlern. Im Konkreten konzentriert sich der Beitrag auf die folgenden Forschungsfragen:

1. Welcher formale Zusammenhang besteht zwischen dem Dokumentationsgrad einer Software und der Anzahl der im Rahmen der Softwarewartung aufgrund von nachträglichen Modifikationen entstehenden Fehler?
2. Welcher optimale Dokumentationsgrad ergibt sich theoretisch aus der Gegenüberstellung der barwertigen Auszahlungen für die Dokumentation während der Softwareentwicklung und für die Fehlerbehebung im Rahmen der Softwarewartung?
3. Welche Hypothesen lassen sich bezüglich einer Abweichung vom optimalen Dokumentationsgrad deduzieren?

B.4: Monetary Valuation of Alternative Modular Software Structures Considering both Technical and Factual Dependencies – A Formal Deductive Approach

Mit bis zu 80% nimmt die Softwarewartung einen Großteil der gesamten Lebenszykluskosten von Softwaresystemen ein (Krishnan et al. 2004). Um insbesondere den Effekt der Ausbreitung von Änderungen (*change propagation*) und die damit verbundenen Auszahlungen für die Wartungen zu reduzieren, schlägt die existierende Literatur vorwiegend die Entwicklung von modularen Softwarearchitekturen mit einer möglichst losen Kopplung der Module vor (Marinescu 2002, Yu et al. 2002, Henry und Kafura 1981, Kemerer 1995). Jedoch resultiert ein Großteil der im Rahmen der Softwarewartung erforderlichen Anpassungen aus Änderungen der fachlichen Anforderungen an ein Softwaresystem (Krishnan et al. 2004). Folglich müssen für die Entwicklung und Bewertung von Softwarearchitekturen neben den technischen Abhängigkeiten von Modulen (Kopplung) ebenfalls fachliche Abhängigkeiten berücksichtigt werden. Diese werden in der Literatur bisher jedoch weitestgehend vernachlässigt, so dass keine adäquaten Ansätze für die ökonomische Bewertung von Softwarearchitekturen in Bezug auf deren Wartungsaufwand existieren. Daher verfolgt dieser Beitrag die Deduktion neuer Zusammenhänge zwischen einer modularen Softwarearchitektur und den barwertigen Auszahlungen für fachlich getriebene Änderungen während der Wartungsphase unter Berücksichtigung von sowohl technischen als auch fachlichen Abhängigkeiten. Diese Hypothesen können eine Grundlage für den Vergleich verschiedener Architekturalternativen bilden und damit die Auswahl von modularen Softwarearchitekturen mit vergleichsweise geringen Auszahlungen für die nachgelagerte Softwarewartung unterstützen. Folgende Forschungsfragen stehen dabei im Fokus:

1. Welcher formale Zusammenhang besteht zwischen einer modularen Softwarearchitektur und den barwertigen Auszahlungen für fachlich getriebene Änderungen im Rahmen der Softwarewartung unter Berücksichtigung von sowohl technischen als auch fachlichen Abhängigkeiten von Softwarefunktionalitäten?
2. Mit welcher Methodik können alternative modulare Softwarearchitekturen in Bezug auf die barwertigen Auszahlungen für fachlich getriebene Änderungen im Rahmen der Wartung miteinander verglichen werden?
3. Kann theoretisch eine Softwarearchitektur, welche ausschließlich eine möglichst lose Kopplung von Modulen realisiert, trotz des Ausbreitungseffekts von Änderungen zu höheren Auszahlungen für die Softwarewartung führen als eine Architektur, welche mit der Konsequenz einer stärkeren Kopplung die fachlichen Abhängigkeiten zwischen Modulen reduziert?

- **Systemische Risiken in Rechnernetzen im Rahmen der IT-Sicherheit (B.5)**

  B.5: Model-based Security Analysis for Mobile Communications

  Der Einsatz mobiler Endgeräte hat in den letzten Jahren ein enormes Wachstum erfahren. Aufgrund der inhärenten Schwachstellen dieser mobilen Geräte und aufgrund der hohen Komplexität der mobilen Netzwerkarchitekturen bedeutet die umfassende Verwendung dieser Endgeräte eine große Herausforderung für die IT-Sicherheit. Für den Umgang mit den für die mobile Kommunikation relevanten Risiken ist die Sicherheitsanalyse als integraler Bestandteil der Konzeption, der Entwicklung und des Managements mobiler Netzwerkarchitekturen unabdingbar. Vor diesem Hintergrund präsentiert dieser Beitrag die Ergebnisse einer modellbasierten Sicherheitsanalyse, welche mit Hilfe der UML-Erweiterung *UMLsec* (Jürjens 2004) die Überprüfung eines Teils der mobilen Kommunikationssysteme des Telekommunikationsunternehmens O_2 (Germany) GmbH & Co. OHG auf die Erfüllung der Anforderungen aus den entsprechenden Sicherheitsrichtlinien zum Ziel hatte. Als ein zentrales Ergebnis stellt der Beitrag die Entwicklung einer neuen Methodik vor, die eine automatische Analyse von Netzwerkarchitekturen auf unsichere Verbindungen und Datenflüsse ermöglicht und die als Grundlage einer entsprechenden Erweiterung der UMLsec-Notation sowie der zugehörigen UMLsec-Werkzeuge dient. Unter anderem stehen dabei die folgenden Forschungsfragen im Mittelpunkt:

1. Welche Sicherheitsanforderungen ergeben sich für die Architektur der mobilen Kommunikationsnetzwerke des Telekommunikationsunternehmens O_2 (Germany) GmbH & Co. OHG aus den unternehmensinternen Sicherheitsrichtlinien?
2. Mit welcher Methodik können Netzwerkarchitekturen (automatisch) auf unsichere Verbindungen und Datenflüsse analysiert werden?

Nach Einleitung, Zielsetzung und fachlicher Einordnung folgen in den Kapiteln II, III und IV die einzelnen Beiträge. Im Anschluss werden in Kapitel V die zentralen Ergebnisse zusammengefasst und Ansatzpunkte für künftigen Forschungsbedarf aufgezeigt.

Literatur (Kapitel I)

Arisholm E, Briand LC, Hove SE, Labiche Y (2006) The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. IEEE Transactions on Software Engineering 32(6):365-381

Baldwin CY, Clark KB (2000) Design Rules: The power of modularity. Volume 1, 1st Edition, Boston: MIT

Beszédes Á, Gergely T, Jász J, Tóth G, Gyimóthy T, Rajlich V (2007) Computation of Static Execute After Relation with Applications to Software Maintenance. In: Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM '07). IEEE, 295-304

Briand LC, Melo WL, Wüst J (2002) Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects. IEEE Transactions on Software Engineering 28(7):706-720

Buldyrev SV, Parshani R, Paul G, Stanley HE, Havlin S (2010) Catastrophic Cascade of Failures in Interdependent Networks. Nature 464:1025-1028

Dash E (2011) S.& P. Downgrades Debt Rating of U.S. for the First Time. The New York Times, verfügbar unter: http://www.nytimes.com/2011/08/06/business/us-debt-downgraded-by-sp.html, Abruf am: 09.09.2011

Davis EP (2003) Towards a typology for systemic financial instability. Economics and Finance Working papers, Brunel University, 03-20

Dzidek WJ, Arisholm E, Briand LC (2008) A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Transactions on Software Engineering 34(3):407-432

Eaddy M, Zimmermann T, Sherwood KD, Garg V, Murphy GC, Nagappan N, Aho AV (2008) Do Crosscutting Concerns Cause Defects?. IEEE Transactions on Software Engineering 34(4):497-515

Faisst U, Prokein O, Wegmann N (2007) Ein Modell zur dynamischen Investitions-rechnung von IT-Sicherheitsmaßnahmen. Zeitschrift für Betriebswirtschaft 77(5):511-538

Glaessner T, Kellermann T, McNevin V (2002) Electronic security: Risk mitigation in financial transactions. World Bank Policy Research Working Paper, verfügbar unter: http://gsmweb.udallas.edu/info_assurance/pdf/risk_mitigation.pdf, Abruf am: 09.09.2011

Gluchowski P, Gabriel R, Dittmar C (2008) Management Support Systeme und Business Intelligence – Computergestützte Informationssysteme für Fach- und Führungskräfte. 2. Auflage, Springer, Heidelberg

Gyimóthy T, Ferenc R, Siket I (2005) Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Transactions on Software Engineering 31(10):897-910

Hamerle A, Plank K (2010) Intransparenzen auf Verbriefungsmärkten – Auswirkungen auf Risikoanalyse und Bewertung. Informatik-Spektrum 33(1):27-36

Henry S, Kafura D (1981) Software Structure Metrics Based on Information Flow. IEEE Transactions on Software Engineering 7(5):510–518

Jürjens J (2004) Secure Systems Development with UML. Springer, Berlin

Kemerer CF (1995) Software complexity and software maintenance: A survey of empirircal research. Annals of Software Engineering 1(1):1–22

Krcmar H (2005) Informationsmanagement. 4. Auflage, Springer, Berlin

Krishnan MS, Mukhopadhyay T, Kriebel CH (2004) A Decision Model for Software Maintenance. Information Systems Research 15(4):396-412

Lemieux C (2003) Network Vulnerabilities and Risks in the Retail Payments System. Federal Reserve Bank of Chicago, Emerging Payments Occasional Papers Series 2003-1F, verfügbar unter: http://wwws.chicagofed.org/digital_assets/publications/risk_management_papers/sr_2003_1f.pdf, Abruf am: 09.09.2011

Marinescu R (2002) Measurement and Quality in Object-Oriented Design. Ph.D. thesis. Department of Computer Science, Politehnica University of Timisoara

McMillan R (2010) Spanish Police Take Down Massive Mariposa Botnet. PCWorld, verfügbar unter: http://www.pcworld.com/businesscenter/article/190634/spanish_police_take_down_massive_mariposa_botnet.html, Abruf am: 09.09.2011

Mertens P, Meier MC (2009) Integrierte Informationsverarbeitung 2 – Planungs- und Kontrollsysteme in der Industrie, 10. Auflage, Gabler, Wiesbaden

o.V. (2001) Milliardenschaden. Manager Magazin, verfügbar unter: http://www.manager-magazin.de/unternehmen/it/0,2828,122753,00.html, Abruf am: 09.09.2011

o.V. (2008a) US-Autoabsatz bricht wegen Finanzkrise ein. Financial Times Deutschland, verfügbar unter: http://www.ftd.de/unternehmen/industrie/:Kreditklemme-USAutoabsatz-bricht-wegen-Finanzkrise-ein/420919.html, Abruf am: 09.09.2011

o.V. (2008b) Island zahlt nicht mehr. Financial Times Deutschland, verfügbar unter: http://www.ftd.de/boersen_maerkte/aktien/anleihen_devisen/:Staatsbankrott-Island-zahlt-nicht-mehr/427188.html, Abruf am: 09.09.2011

o.V. (2009a) Volume of Subprime Mortgages Outstanding Continues to Shrink. IMF-News, verfügbar unter: http://imfpubs.wordpress.com/category/subprime/, Abruf am: 09.09.2011

o.V. (2009b) 10 Billionen Dollar - die Krise wird teuer. Süddeutsche Zeitung, verfügbar unter: http://www.sueddeutsche.de/wirtschaft/weltwirtschaft-billionen-dollar-die-krise-wird-teuer-1.163847, Abruf am: 09.09.2011

o.V. (2009c) Hypo Real Estate – Regierung: Enteignung nur als „ultima ratio“. Frankfurter Allgemeine, verfügbar unter: http://www.faz.net/artikel/C31151/hypo-real-estate-regierung-enteignung-nur-als-ultima-ratio-30010038.html, Abruf am: 09.09.2011

o.V. (2011) Schuldenkrise in Italien: Merkel drängt Berlusconi zum Sparen. Süddeutsche Zeitung, verfügbar unter: http://www.sueddeutsche.de/geld/schuldenkrise-in-italien-italien-und-die-billionen-euro-frage-1.1118630, Abruf am: 09.09.2011

Prechelt L, Unger-Lamprecht B, Philippsen M, Tichy WF (2002) Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance. IEEE Transactions on Software Engineering 28(6):595-606

Rochet JC, Tirole J (1996) Interbank Lending and Systemic Risk. Journal of Money, Credit and Banking 28(4):733-762

Romeike F, Hager P (2009) Erfolgsfaktor Risiko-Management 2.0 Methoden, Beispiele, Checklisten. Praxishandbuch für Industrie und Handel. 2. Auflage, Gabler Verlag, Wiesbaden

Rudolph B (2008) Lehren aus den Ursachen und dem Verlauf der internationalen Finanzkrise. Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung 60: 713-741

Singh Y, Kaur A, Malhotra R (2010) Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness Models. Software Quality Journal 18(1):3-35

Sommer P, Brown I (2011) Reducing Systemic Cybersecurity Risk. Beitrag zum OECD-Projekt „Future Global Shocks“, verfügbar unter: http://www.oecd.org/dataoecd/57/44/46889922.pdf, Abruf am: 09.09.2011

WKWI (Wissenschaftliche Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e. V.) (1994) Profil der Wirtschaftsinformatik. Zeitschrift Wirtschaftsinformatik 36:80–81

WKWI, GI FB WI (Wissenschaftliche Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e. V. und Fachbereich Wirtschaftsinformatik in der Gesellschaft für Informatik e. V.) (2011) Profil der Wirtschaftsinformatik. Verfügbar unter: http://wi.vhbonline.org/aktuelles-termine/, Abruf am: 09.09.2011

Yu P, Systä T, Müller H (2002) Predicting Fault-Proneness Using OO Metrics: An Industrial Case Study. In: Proceedings of the 6th European Conference on Software Maintenance and Reengineering (CSMR '02), pp. 99-107

II Systemische Risiken in der Finanzwirtschaft und die Rolle der IT am Beispiel der Finanzkrise

Am Beispiel der seit Juli 2007 virulent gewordenen Finanzkrise widmet sich dieses Kapitel den systemischen Risiken in der Finanzwirtschaft aus der Perspektive der Wirtschaftsinformatik in ihrer Rolle als „Werkzeugmacher“.

Dazu beschreibt der erste Abschnitt die der Krise zugrunde liegenden Strukturen und Merkmale des Handels mit amerikanischen Immobilienkrediten sowie die relevanten Verbriefungstransaktionen und unterzieht diese im Rahmen einer Betrachtung von Ertrags- und Risikopositionen einer ökonomischen Analyse. Darauf aufbauend werden die Ursachen, die Zusammenhänge und der Verlauf der Finanzkrise aufgezeigt.

Auf der Basis des ersten Abschnitts diskutiert der zweite Abschnitt die Rolle und Verantwortung der Wirtschaftsinformatik für die Entstehung und den Verlauf der Finanzkrise. Im Fokus stehen dabei insbesondere IT-basierte Systeme zur Entscheidungsunterstützung, welche im Allgemeinen das mittlere und obere Management bei Entscheidungsprozessen durch die Bereitstellung relevanter Informationen und die Generierung von Handlungsempfehlungen auf der Basis von Entscheidungsmodellen unterstützen und welche im speziellen Fall der Finanzkrise unter Verwendung der den Verbriefungsstrukturen zugrunde liegenden finanzmathematischen Modelle beispielsweise im Rahmen des Risikomanagements und bei der Erstellung von Ratings eingesetzt werden.

1 Beitrag: „*Ursachen und Auswirkungen der Subprime-Krise*“

Autoren:	Peter Bartmann, Prof. Dr. Hans Ulrich Buhl, Michael Hertel
	Kernkompetenzzentrum Finanz- & Informationsmanagement, Lehrstuhl für BWL, Wirtschaftsinformatik, Informations- & Finanzmanagement (Prof. Dr. Hans Ulrich Buhl) Universität Augsburg, D-86135 Augsburg
	peter.bartmann@wiwi.uni-augsburg.de, hans-ulrich.buhl@wiwi.uni-augsburg.de, michael.hertel@wiwi.uni-augsburg.de

Erschienen in:	Informatik-Spektrum, 32, 2, 2009, S. 127-145;
	B. Nietert ed., Die Eskalation der Finanz- zur Wirtschaftskrise, Fritz Knapp Verlag, Frankfurt am Main (Deutschland), 2011, S. 11-45.

Zusammenfassung:

Auf Basis der Finanzmarktkrise seit Juli 2007 untersucht dieser Beitrag die Strukturen und Merkmale des Handels mit amerikanischen Immobilienkrediten, welche zur Subprime-Krise führten. Hierbei werden die Finanzinstrumente, die eine Verbindung zwischen Kredit- und Kapitalmärkten herstellen, dargestellt und einer ökonomischen Analyse unterzogen. Anhand der Analyse von Ertrags- und Risikopositionen können asymmetrische Verteilungen identifiziert werden. Darauf aufbauend werden die Ursachen und Zusammenhänge der Subprime-Krise aufgearbeitet.

1.1 Einleitung

Kein anderes Thema hat die Finanzwelt in den vergangenen Monaten derart dominiert wie die als Subprime-Krise bekannt gewordene Spekulationsblase mit Verbriefungen amerikanischer Hypothekendarlehen. Dieser Beitrag zeigt die Faktoren auf, welche zur internationalen Finanzkrise führten. Dazu werden im zweiten Abschnitt die Verbriefungstransaktionen als die Finanzinstrumente, welche die Verbindung zwischen den Kredit- und Kapitalmärkten herstellen, detailliert aufgezeigt und anschließend im Rahmen der Betrachtung von Ertrags- und Risikopositionen einer ökonomischen Analyse unterzogen. Zusammen mit der Darstellung der makroökonomischen Rahmenbedingungen und der Verhaltensweise der beteiligten Akteure können auf dieser Basis im vierten Abschnitt die Ursachen und Wirkungszusammenhänge der Subprime-Krise herausgearbeitet werden. Die Beschreibung des Verlaufs der Krise spiegelt dabei die wesentlichen Entwicklungen bis zum 31.12.2008 wider. Der fünfte Abschnitt beschreibt schließlich die für die Marktstrukturen charakteristischen Anreizsysteme als Grundlage des Verhaltens der beteiligten Akteure.

Aufgrund der Komplexität der in der Finanzwelt vorherrschenden Strukturen und der Vielzahl theoretischer Verbriefungsoptionen kann eine vollständige Betrachtung der Realwelt jedoch nicht erfolgen. Erfasst werden sollen daher die grundlegenden Fälle. Ziel dieses Beitrags ist es, auf Basis existierender Literatur die wesentlichen Instrumente, Eigenschaften und Zusammenhänge sowie darauf aufbauend die grundlegenden Ursachen der internationalen Finanzkrise insbesondere für solche Leser darzustellen, die über kein Expertenwissen in diesem Bereich verfügen.

1.2 Verbriefungstransaktionen am US-Hypothekenmarkt

1.2.1 Grundstruktur von Verbriefungen

Am Beginn des Verbriefungsprozesses steht der Verkauf der Kreditforderungen durch die kreditvergebende Hypothekenbank (Originator). Dabei können Kreditforderungen entweder mit allen Rechten und Risiken (True Sale Transaktion) (True Sale International 2009a) oder nur die Ausfallrisiken des Forderungsportfolios mittels Kreditderivaten wie beispielsweise Credit Default Swaps (Synthetische Transaktion) an eine dritte Partei verkauft werden (Rudolph et al. 2007). Da der Krise in ihrer Entstehung in den Vereinigten Staaten überwiegend True Sale Transaktionen zugrunde liegen, werden diese für die folgende Darstellung der grundsätzlichen Verbriefungsstrukturen (vgl. Abb. II-1) unterstellt. Als Verkaufsertrag erhält der Originator entweder eine festgesetzte Provision oder einen Teil der durch die anschließende Verbriefung erzielten Zinsmarge. Die Zins-

ansprüche gehen auf den Erwerber der Kreditforderungen über.

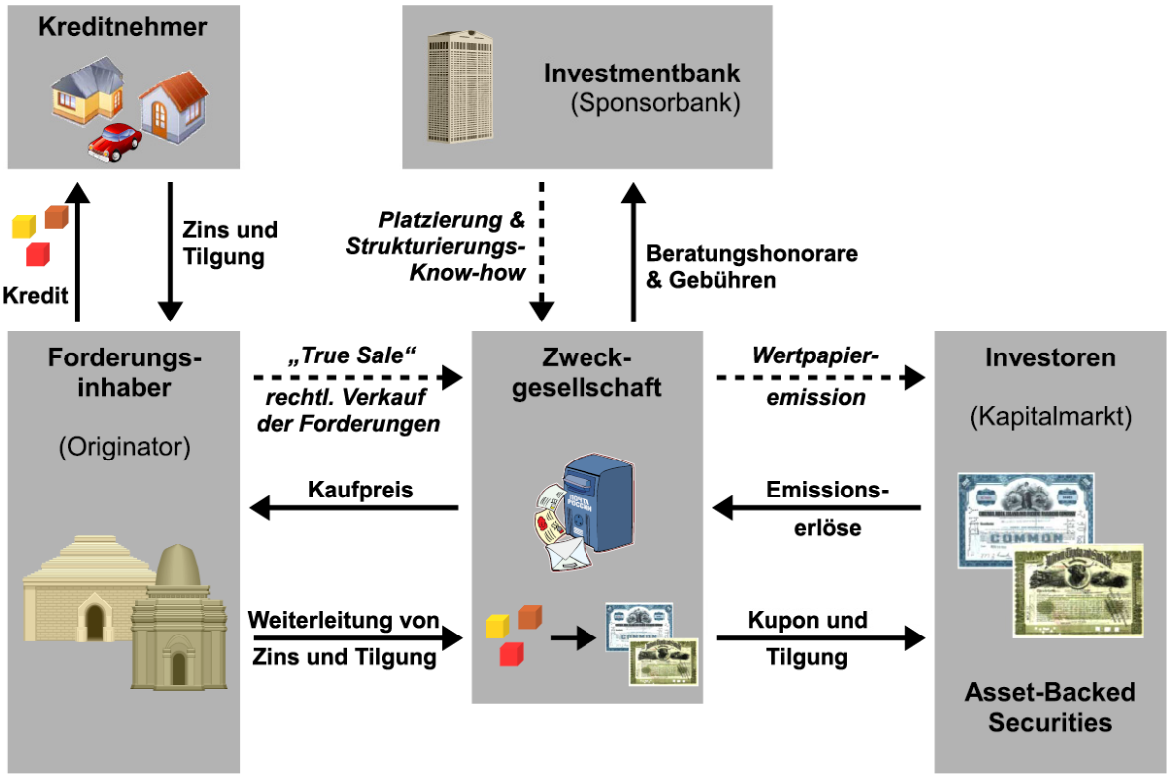

Abb. II-1: Grundstruktur von Verbriefungen

Käufer von Kreditforderungen sind in der Regel Zweckgesellschaften, die auch *Special Purpose Vehicle* (SPV) genannt und von den Originatoren selbst oder von Investmentbanken alleine für diese Transaktionen gegründet werden (Rudolph et al. 2007). Den Kauf der Kreditforderungen finanziert die Zweckgesellschaft durch die Emission von Wertpapieren, den Asset-Backed Securities (ABS)[1], am Kapitalmarkt. Diese verzinslichen Wertpapiere haben Zahlungsansprüche gegen die Zweckgesellschaft zum Gegenstand, die aus den Zins- und Tilgungsleistungen der Kreditnehmer bedient werden (vgl. Abb. II-1). Besichert sind die ABS durch die den Kreditforderungen zugrunde liegenden Vermögenswerte, welche den Investoren als Haftungsgrundlage dienen. ABS werden in der Regel *over the counter* (OTC), das heißt entweder direkt zwischen den Banken und Investoren oder über außerbörsliche Handelsplattformen, gehandelt (o.V. 2008c).

Mit der Durchführung der Transaktionen durch eine Zweckgesellschaft wird eine Trennung der Vermögenswerte vom Insolvenzrisiko der Originatoren erreicht. Die Kreditforderungen der Zweckgesellschaften sind somit nicht Teil der Haftungsmasse der Originatoren (Rudolph et al. 2007). Die Zweckgesellschaften werden in der Regel mit dem Ziel der „Insolvenzferne“ gegründet und deren Tätigkeiten auf definierte Verbriefungstransaktionen beschränkt. Da somit andere Geschäftsrisiken ausgeschlossen sind (Jortzik 2005), tragen die emittierten Wertpapiere und damit auch die Investoren ausschließlich die Ausfallrisiken der Kredite des unterliegenden Forderungsportfolios. Grundlage von ABS-Emissionen bilden in der Grundform Portfolios mit Volumen von mehreren Milliarden US-Dollar (Hemmerich 2008) aus gleichartigen, regional diversifizierten Kreditforderungen (Fender und Mitchell 2005). Trotz Homogenität dieser Forderungen kann die Bonität der Schuldner Schwankungen unterliegen.

Bei der Durchführung von Verbriefungstransaktion nehmen die auf das Kundengeschäft spezialisierten Hypothekenbanken den Dienst von Investmentbanken in Anspruch. Diese unterstützen als Sponsorbanken oder Arrangeure die Planung und Realisierung der Transaktionen (Rudolph et al. 2007). Durch diese Dienstleistungen können Investmentbanken ohne Aufbau eines eigenen Kundengeschäfts am Kreditgeschäft teilhaben. Die Strukturierung neuartiger Finanzprodukte entwickelte sich so zu einer Kernkompetenz der Investmentbanken. Im Vorfeld gewähren diese den Hypothekenbanken oftmals eine revolvierende Fazilität (warehouse line), bis das Forderungsportfolio ein ausreichend großes Volumen für den Verkauf an die Zweckgesellschaft und den Verbriefungsprozess erreicht (International Finance Corporation 2008). Daneben können die Sponsorbanken als underwriter das Risiko, dass eine ABS-Emission am Kapitalmarkt nicht vollständig platziert werden kann, gegen Gebühr übernehmen (Deutsche Bundesbank 2007).

Eine weitere Möglichkeit stellt der Verkauf der Hypothekenforderung an die von der U.S. Regierung staatlich geförderten Hypothekenbanken Fannie Mae oder Freddie Mac dar. Aufgabe dieser Banken ist die Aufrechterhaltung der Liquidität des amerikanischen Hypothekenmarkts zur Förderung privaten Wohneigentums. Um Hypothekenforderungen an Fannie Mae und Freddie Mac verkaufen zu dürfen, bedarf es der vorhergehenden Lizensierung der Hypothekenbank. Fannie Mae und Freddie Mac finanzieren

1 Unter *Asset-Backed Securities* werden generell durch Vermögenswerte besicherte Wertpapiere verstanden. Die Vermögenswerte können verschiedener Natur sein. Liegen Hypotheken zur Sicherung vor, so spricht man genauer von *Mortgage-Backed Securities*. Im Folgenden wird auch für hypothekenbesicherte Wertpapiere der Oberbegriff *Asset-Backed Securities* verwendet.

sich wie die oben beschriebenen SPVs durch die Ausgabe von ABS. Daneben haben sie Zugriff auf staatlich gedeckte, günstige Kredite (Broome und Markham 2008).

Durch den Verkauf der Forderungen entfällt für den Originator die Pflicht zur Unterlegung des Kredits mit teurem Eigenkapital und die frei gewordene Deckungsmasse kann für weitere Kreditgeschäfte genutzt werden. Die vorzeitige Monetarisierung der illiquiden Aktiva erhöht somit die finanziellen Mittel der Hypothekenbank, welche bei Immobilienkrediten über lange Zeiträume gebunden gewesen wären, und ermöglicht ein dynamisches Wachstum des Kreditgeschäfts (Hemmerich 2008). Da zusammen mit dem Kredit auch dessen Ausfallrisiken veräußert werden, befreit der Originator seine Bilanz von Risikopositionen. Das provisionsorientierte Geschäftsmodell ist auf risikolose Erträge für die Originatoren und die beteiligten Makler ausgelegt und wird aufgrund der Veräußerungsabsicht als Originate-to-Distribute bezeichnet (Deutsche Bundesbank 2007).

Für Investoren stellen ABS eine Möglichkeit der Risikostrukturierung ihres Portfolios dar (Jortzik 2005). Für die Finanzwelt bedeutete der Verkauf von Hypothekenkrediten und der anschließende Handel mit den emittierten Wertpapieren grundsätzlich eine Streuung der den Krediten inhärenten Risiken. Diese geografische Risikodiversifikation erscheint aus volkswirtschaftlicher Sicht sinnvoll, da Risikokonzentrationen ausgeglichen und die Widerstandsfähigkeit gegen lokale Verwerfungen erhöht werden können. Jedoch besteht die Gefahr einer Transmission von exogenen Schocks in das internationale Finanzsystem (Deutsche Bundesbank 2007).

1.2.2 Ausbau der Verbriefungstransaktionen

1.2.2.1. Ziele

Die Höhe der Zinsen, die eine Zweckgesellschaft an die Investoren der emittierten Wertpapiere ausbezahlt, wird durch die Ausfallrisiken der unterliegenden Kreditforderungen bestimmt. Je höher das erwartete Ausfallrisiko des Forderungsportfolios ist, desto höher ist der in den Wertpapierzinsen enthaltende Risikoaufschlag. Eine Reduzierung des Risikoaufschlags der Wertpapierzinsen kann bei gegebener Risikoaversion der Investoren einerseits durch eine Veränderung der Struktur des Forderungsportfolios, zum Beispiel durch die Erhöhung des Anteils solider Kreditforderungen, oder durch die Nutzung von Diversifikationseffekten erfolgen. Daneben existieren jedoch Mechanismen zur Bonitätsverbesserung (*Credit Enhancement*), die auf die Transaktionsstrukturen aufbauen und keine Veränderungen des Forderungsportfolios erfordern.

1.2.2.2. Tranchierung

Die *Tranchierung* (Subordination) ist der bedeutendste Mechanismus des Credit Enhancements. Dabei wird eine Emission von ABS in unterschiedliche Risikoklassen (Tranchen) derart aufgeteilt, dass Wertpapiere einer vorrangigen Tranche bei der Distribution der Zahlungen Priorität gegenüber Wertpapieren einer nachrangigen Tranche besitzen. Umgekehrt betrachtet partizipieren nachrangige Tranchen vorrangig an Ausfällen der Zins- und Tilgungszahlungen, falls die zugrundeliegenden Kredite nicht bedient werden. Erst wenn die Zins- und Tilgungszahlungen der nachrangigsten Tranche einer Emission aufgrund von Kreditausfällen vollständig ausgefallen sind, wird die nächste Tranche mit den residualen Kreditausfällen belastet (Ashcraft und Schuermann 2008). Diese Verteilungssystematik wird auch als Wasserfallprinzip bezeichnet und ist in Abb. II-2 illustriert (Rudolph und Scholz 2007).

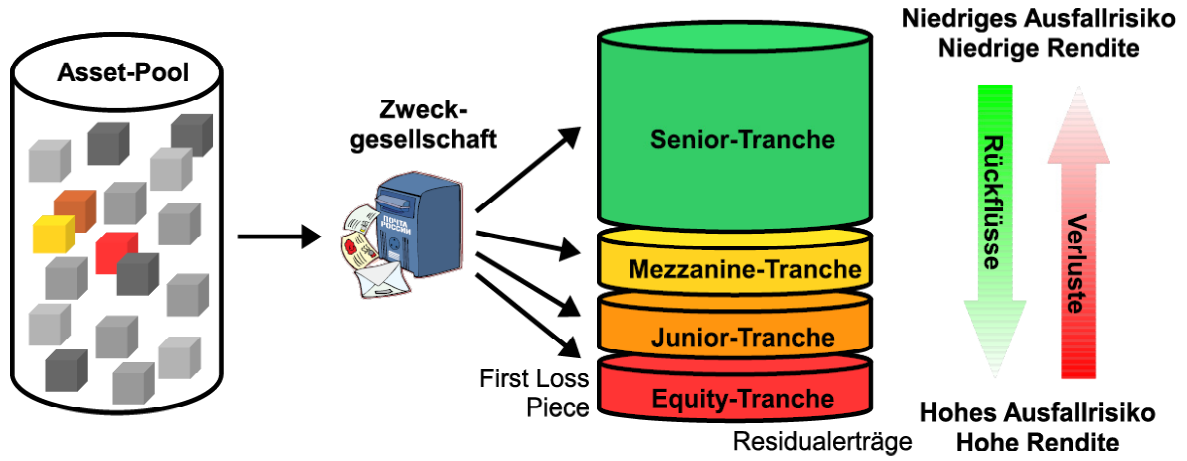

Abb. II-2: Tranchierung

Insbesondere den Investoren dienen Ratings als wesentliche Informationsquelle über die Ausfallrisiken der Tranchen von Wertpapieremissionen. Zur Erstellung solcher Ratings ermitteln externe Ratingagenturen in einem ersten Schritt mögliche Wahrscheinlichkeitsverteilungen für die Verluste im unterliegenden Forderungsportfolio. Diese Kreditrisikomodellierungen betrachten einerseits Informationen beispielsweise über die Qualität und Ausfallswahrscheinlichkeiten der einzelnen Kreditforderungen, sowie andererseits Annahmen wie die Korrelation der Ausfälle der Forderungen. Viele dieser Informationen basieren dabei auf der Extrapolation historischer Daten. Darauf aufbauend wird in einem zweiten Schritt die Distribution der Cashflows auf die einzelnen Tranchen anhand verschiedener Szenarien simuliert (Basel Committee on Banking Supervision

2005). Als Ergebnis werden schließlich die erwarteten Verluste beziehungsweise die Ausfallwahrscheinlichkeiten für die einzelnen Tranchen ermittelt und in ein alphanumerisches Rating für die Wertpapiere jeder Tranche überführt (Fender et al. 2008).

Bei der Emission von ABS werden in der Regel mehrere Tranchen gebildet, wobei die höchstrangigen für gewöhnlich als Senior-Tranchen und die nachrangigsten als Junior- bezeichnet werden. Tranchen, die zwischen den Senior- und Junior-Tranchen rangieren, werden Mezzanine-Tranchen genannt (Ashcraft und Schuermann 2008) (vgl. Abb. II-2). Je nachrangiger eine Tranche, das heißt je höher das Ausfallrisiko dieser Tranche, desto schlechter fällt das Rating aus und desto höher sind die Zinszahlungen an die Investoren der entsprechenden Papiere (Ashcraft und Schuermann 2008). Die nachrangigste Tranche (Equity-Tranche), die als erstes an den Verlusten partizipiert und daher auch als First Loss Piece bezeichnet wird, ist normalerweise mit keinem Rating versehen (True Sale International 2009a) und wird anstelle einer festen Verzinsung mit den Residualerträgen aus den Kreditforderungen bedient (vgl. Abb. II-2). Um ein möglichst gutes Rating für die höheren Tranchen zu erreichen, übersteigt das Volumen der Equity-Tranche in der Regel die erwarteten Kreditausfälle und bildet eine Sicherheit für die übrigen Tranchen (Rudolph et al. 2007). Das First Loss Piece kann beispielsweise an Anleger mit wenig Risikoaversion wie Hedge-Fonds verkauft werden. Im Gegenzug erwirbt dieser die Ansprüche an den Residualerträgen. Oftmals verbleiben das First Loss Piece sowie die Verwaltung der Kredite des verkauften Forderungsportfolios jedoch beim Originator (Hartmann-Wendels 2007), der weiterhin auch das Servicing der Kredite übernimmt. Dadurch kann der Originator im Vergleich zum Halten der Kreditforderungen sein maximales Ausfallrisiko auf das Volumen der Equity-Tranche begrenzen (Franke und Krahnen 2005, Krahnen 2005), während sich für die Kreditnehmer keine merkbaren Unterschiede durch den Kreditverkauf ergeben. Zudem können dadurch Probleme des Moral Hazard vermieden werden, die sich bei einem Verkauf der Equity-Tranche ergeben würden (Rudolph et al. 2007).

Grundsätzlich zielt die Tranchierung auf eine gezielte Nutzung von Unvollkommenheiten, die beispielsweise aufgrund asymmetrischer Informationsverteilungen bestehen (DeMarzo 2005, Rudolph und Scholz 2007), der Kapitalmärkte ab. Denn in der Regel gelingt es durch diese Risikostrukturierung, je nach Qualität der unterliegenden Kreditforderungen, einen Großteil der Wertpapiere als Senior-Tranchen zu emittieren und für diese damit ein deutlich besseres Rating als das durchschnittliche Rating des Forderungsportfolios zu erzielen. Da die Gesamtheit einer ABS-Emission inklusive der Equity-Tranche jedoch das gleiche Risiko wie das unterliegende Kreditforderungsportfolio trägt, müsste unter Annahme eines vollkommenen Kapitalmarkts (Steiner und Bruns 2000) die

Zinszahlungen aus den Kreditforderungen genau der Verzinsung der ABS entsprechen. Bei den bis 2006/2007 üblichen Risikoaufschlägen war aber mit diesem Mechanismus eine Senkung der Finanzierungskosten möglich, so dass aus den Zinszahlungen der unterliegenden Forderungen einerseits noch die Kosten der Verbriefungstransaktion inklusive der Beratungshonorare und Gebühren für die beteiligte Sponsorbank gedeckt werden konnten. Andererseits ließen strukturierte Wertpapiere mit erstklassigem Rating auch eine höhere Rendite für die Investoren erwarten als bspw. Staatsanleihen mit gleichem Rating. Für Anleger mit geringer Risikoaversion waren insbesondere die Equity-Tranchen attraktiv, da bei entsprechend niedrigen Ausfallraten der unterliegenden Kreditforderungen mit Investitionen in die Equity-Tranchen sehr hohe Renditen erzielt werden konnten.

Aufgrund der Absenkung der Diskontrate zur Abfederung der Wirtschaft nach der New Economy Blase sahen sich viele institutionelle Anleger und Banken einerseits mit nur mäßigen Ertragschancen und andererseits mit nach wie vor hohen Renditeerwartungen der Anleger konfrontiert. Dieses Dilemma versuchten zahlreiche Banken – zu denen neben Investment- und Geschäftsbanken auch viele staatliche Banken zählten – durch Investments in vermögensbesicherte Wertpapiere mit teilweise beträchtlichem Umfang zu lösen (Rudolph 2008).

1.2.2.3. Mehrfach-Tranchierung

Einer ABS-Emission unterliegt für gewöhnlich ein diversifiziertes Portfolio an homogenen Kreditforderungen (Fender und Mitchell 2005) wie beispielsweise Hypothekenkredite, Kreditkarten- oder Kraftfahrzeugdarlehen. Mit einer weiteren Form von Wertpapieren, den sogenannten Collateralized Debt Obligations (CDOs), wird die Generierung zusätzlicher Zinsmargen durch eine weitere Risikodiversifikation (Basel Committee on Banking Supervision 2005) und durch die wiederholte Nutzung von Marktunvollkommenheiten angestrebt. Dazu generiert eine weitere Zweckgesellschaft ein heterogenes Portfolio aus Wertpapieren verschiedenartiger ABS-Emissionen unterschiedlicher Tranchen[2]. Der Kauf dieser Finanztitel wird nach einer erneuten Tranchierung durch die Emission von CDOs finanziert (vgl. Abb. II-3).

Bei der Emission von CDOs kann also durch eine geeignete Kombination ein zusätzlicher Diversifikationseffekt in zwei Dimensionen geschaffen werden: Zum einen erfolgt

2 Manchen CDO-Emissionen unterliegen Portfolios, die neben ABS auch Kreditforderungen enthalten. Da dies in der Regel jedoch nur einer weiteren Diversifikation dient, spielen solche Portfolios für die Darstellung der Grundprinzipien von CDO-Emissionen nur eine untergeordnete Rolle und werden im Folgenden vernachlässigt.

eine Mischung von Wertpapieren, denen verschiedene Forderungsarten unterliegen. Zum anderen werden ABS verschiedener Risikoklassen kombiniert. Mit der daraus resultierenden Zinsmarge werden höhere Residualerträge für die Investoren der jeweiligen Equity-Tranche generiert und die anfallenden Transaktionskosten sowie Beratungshonorare der beteiligten Sponsorbanken finanziert.

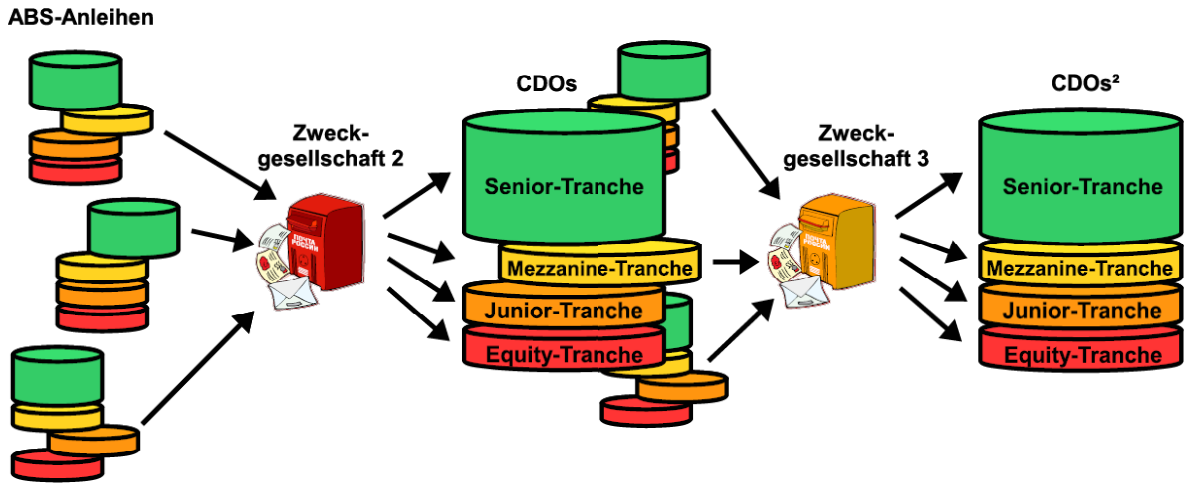

Abb. II-3: Mehrfach-Tranchierung mit CDOs und CDOs²

Dieser Diversifikationseffekt kann dabei mehrfach in Folge genutzt werden. So kann einer Wertpapieremission auch ein Portfolio aus CDOs unterlegt werden. Wertpapiere, deren Zinsen überwiegend aus den Cashflows von CDOs bedient werden, werden in der Regel als *Collateralized Debt Obligations squared* (CDOs²) bezeichnet (Whetten und Adelson 2005) (vgl. Abb. II-3). Das Ziel solcher Transaktionen ist dabei, durch eine noch umfassendere Diversifikation und mittels erneuter Tranchierung einen noch größeren Teil der Wertpapiere einer CDO²-Emission als erstklassig geratete Senior-Tranchen zu strukturieren und damit erneut eine zusätzliche Zinsmarge zu generieren.

1.2.2.4. Weitere Mechanismen des Credit Enhancements

(Mehrfach-)Tranchierungen bilden den wichtigsten Mechanismus des Credit Enhancements für die Emission von ABS, CDOs und CDOs². Darüber hinaus existieren noch weitere Möglichkeiten zur Bonitätsverbesserung, von denen einige im Folgenden ohne Anspruch auf Vollständigkeit aufgezeigt werden.

Bei der sogenannten *Übersicherung* emittiert die Zweckgesellschaft Wertpapiere zu einem geringeren Nominalwert als dem Wert des Forderungsportfolios sowie mit einer geringeren Kuponzahlung als den Zinszahlungen der unterliegenden Kreditforderungen.

Dadurch können mögliche Zahlungsausfälle der Kreditforderungen bis zur Höhe dieser Differenz kompensiert werden. Daneben kann eine Bonitätsverbesserung beispielsweise auch durch einen *Reservefonds* erreicht werden, welcher in der Regel vom Originator zum Ausgleich eventueller Zahlungsausfälle gebildet wird. Darüber hinaus kann ein Teil der erzielten Zinsmarge auch in einem sogenannten *Überlaufkonto* gesammelt und zur Deckung späterer Zahlungsausfälle verwendet werden (Rudolph et al. 2007).

1.2.2.5. Fristentransformation

Die Emission von ABS, CDOs oder CDOs² dient standardmäßig einer fristenkongruenten Finanzierung der von der Zweckgesellschaft gekauften Forderungen beziehungsweise Wertpapiere. Wird eine normale Zinsstrukturkurve, bei der die kurzfristigen Kapitalmarktzinsen unter den längerfristigen liegen (Franke und Hax 2004), sowie ein stets liquider Geld- und Kapitalmarkt unterstellt, so ergibt sich mit einer nicht-fristenkongruenten Finanzierung eine weitere Möglichkeit für die Generierung von Zinsmargen.

Dazu kauft eine weitere Zweckgesellschaft ein Portfolio aus mittel- beziehungsweise längerfristigen ABS, CDOs oder CDOs² von verschiedenen SPVs am Kapitalmarkt und finanziert den Kaufpreis revolvierend durch die Emission von sogenannten *Asset-Backed Commercial Papers* (ABCPs) mit kürzerer Laufzeit am Geldmarkt (Pfingsten 2007) (vgl. Abb. II-4). Diese Transaktionen werden dabei in der Regel über sogenannte *Conduits* abgewickelt. Conduits sind eine spezielle Form von Zweckgesellschaften, die ursprünglich für die fristenkongruente Finanzierung kurzfristiger Handelsforderungen gegründet wurden und seit einigen Jahren vermehrt als Arbitrage-Vehikel für die Emission von ABCPs verwendet werden (TSI-Arbeitsgruppe 2008). Dabei wird eine ABCP-Emission mit einer Equity-Tranche und einer großvolumigen Senior-Tranche normalerweise lediglich in zwei Tranchen aufgeteilt, da am Geldmarkt in der Regel nur Wertpapiere erstklassiger Bonität verkauft werden können (Franke und Hax 2004). Um bei der Tranchierung trotz fehlender Junior- und Mezzanine-Tranchen als zusätzliche Verlustpuffer einen Großteil der ABCP-Emission als Senior-Tranche strukturieren zu können, werden überwiegend Wertpapiere aus erstklassig gerateten Senior-Tranchen in das unterliegende Portfolio aufgenommen.

Derartige ABCPs besitzen eine Fälligkeit zwischen einem und 360 Tagen, in den meisten Fällen zwischen 30 und 50 Tagen (True Sale International 2009b). Nach der normalen Zinsstrukturkurve werden die längerfristigen ABS also mit höheren Zinsen vergütet als die kurzfristigen ABCPs. Mit der durch die Fristentransformation entstehenden Zinsarbitrage werden wiederum hohe Residualerträge für die Investoren der Equity-

Tranche generiert sowie die Transaktionskosten und die Beratungshonorare für die Sponsorbanken gedeckt. Der Mechanismus der Fristentransformation wird in Abb. II-4 verdeutlicht.

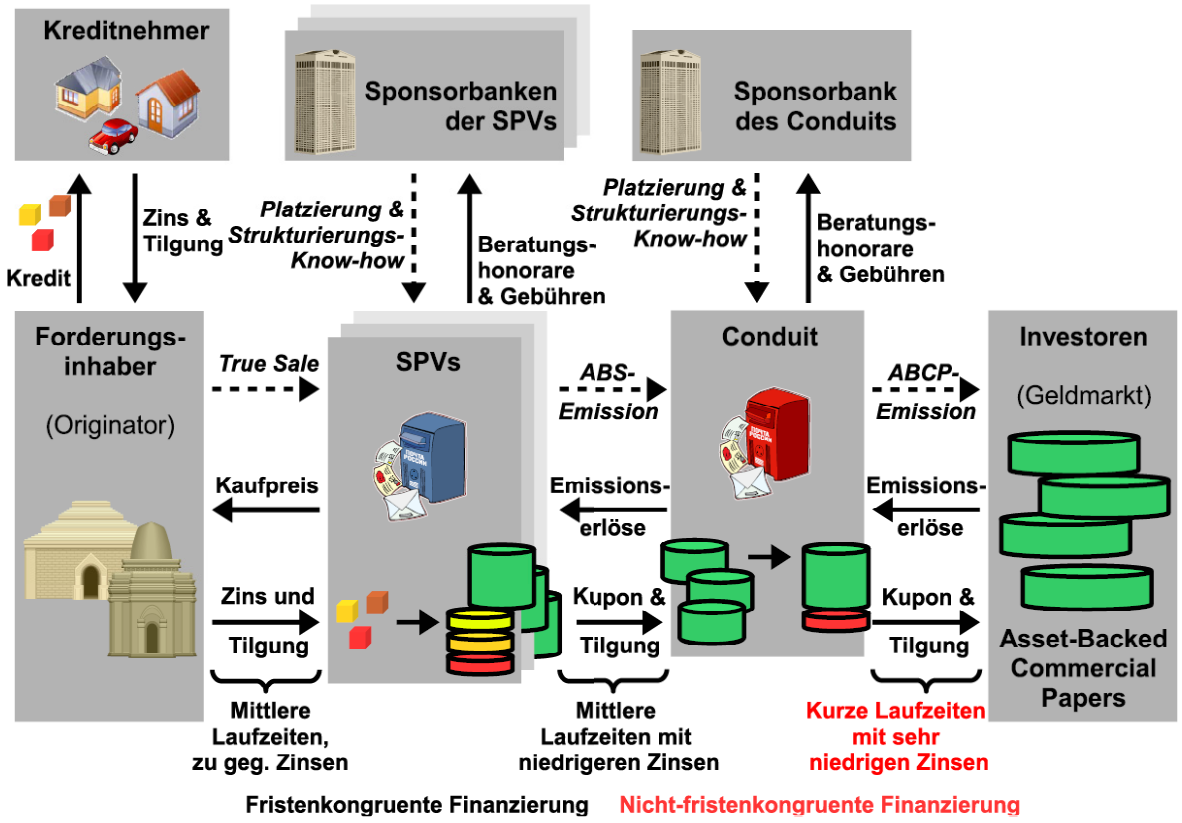

Abb. II-4: Fristentransformation

Aufgrund der revolvierenden Finanzierung würde der emittierenden Zweckgesellschaft ein laufendes Liquiditätsrisiko entstehen. Da eine ABCP-Emission der Finanzierung der Tilgungszahlungen der vorherigen Emission dient, entsteht ein kurzfristiger Liquiditätsbedarf, sofern eine ABCP-Emission nicht vollständig zum Nominalwert am Geldmarkt verkauft werden kann. Da die Zweckgesellschaften für gewöhnlich mit dem Ziel der Insolvenzferne (Jortzik 2005) auf Geschäftstätigkeiten im Rahmen der Verbriefungstransaktion beschränkt und nur mit wenig oder keinem Eigenkapital ausgestattet sind, könnten solche kurzfristigen Liquiditätsengpässe in der Regel nicht eigenständig von den Conduits ausgeglichen werden und würden in diesem Fall in der Insolvenz der Zweckgesellschaft und einem entsprechenden Ausfall der Zins- und Tilgungszahlungen an deren Investoren resultieren. Die ABCPs würden also neben den Ausfallrisiken des unterliegenden Portfolios auch das Liquiditätsrisiko der Zweckgesellschaft bei ungünsti-

gen Entwicklungen am Geldmarkt tragen und folglich mit einem entsprechend schlechtem Rating ausgezeichnet werden.

Um dennoch ein sehr gutes Rating für die großvolumige Senior-Tranche der ABCPs zu erzielen, sind spezielle Mechanismen zur Bonitätsverbesserung nötig. Dazu stellt ein externer Sicherungsgeber, der in der Regel die Sponsorbank selbst ist, die Zahlungsfähigkeit der Zweckgesellschaft mittels Vergabe einer *Liquiditätslinie* meist in Höhe des gesamten Emissionsvolumen sicher (Deutsche Bundesbank 2007). Der Sicherungsgeber übernimmt somit das gesamte Liquiditätsrisiko der Zweckgesellschaft, während das Ausfallrisiko der unterliegenden Forderungen weiterhin von den Wertpapieren getragen wird. Durch diese Form des Credit Enhancements kann das erstklassige Rating für die ABCPs gehalten werden.

Wird wiederum ein stets liquider Markt für strukturierte Wertpapiere unterstellt, so können Zweckgesellschaften, deren Portfolios aus kapitalmarktnotierten, fungiblen Wertpapieren bestehen und durch geldmarktnotierte ABCPs finanziert werden, anstelle von umfassenden Liquiditätslinien auch mit sogenannten *Marktwerttriggern* ausgestattet werden. Fällt der Marktwert oder das Rating des unterliegenden Portfolios unter einen festgelegten Schwellenwert, so darf dieses Portfolio zum Schutz der Investoren der Senior-Tranche nicht mehr durch ABCPs finanziert werden und muss in der Regel am Kapitalmarkt verkauft werden. Normalerweise wird ein marktwertbasierter Schwellenwert auf das Finanzierungsvolumen abzüglich des hälftigen Werts der Equity-Tranche festgesetzt. Das unterliegende Portfolio muss also dann verkauft werden, wenn der Wert des Portfolios am Kapitalmarkt im Vergleich zu dessen Nominalwert um mehr als die Hälfte des Volumens der Equity-Tranche sinkt. Der Verkaufserlös dient schließlich der Kompensation der Zins- und Tilgungszahlungen der letzten ABCP-Emission (Crouhy und Turnbull 2008, TSI-Arbeitsgruppe 2008).

Eine derartige Zweckgesellschaft ist in der Regel mit einer partiellen Liquiditätslinie in Höhe von 5% bis 10% des Nominalwerts des unterliegenden Portfolios ausgestattet und wird auch *Structured Investment Vehicle* (SIV) genannt (Deutsche Bundesbank 2007). Dadurch entfällt einerseits ein Großteil des Liquiditätsrisikos für den externen Sicherungsgeber. Andererseits steigt durch einen solchen Mechanismus das Risiko der Equity-Tranche, da diese nach einem Zwangsverkauf des unterliegenden Portfolios unabhängig von zukünftigen eventuell positiven Wertentwicklungen des Portfolios nicht oder nur teilweise bedient werden kann.

1.3 Ertrags- / Risikobetrachtung der Verbriefungstransaktionen

Bei näherer Analyse der Gesamtstruktur der Verbriefungstransaktionen zeigt sich eine grundsätzliche Ertrags- und Risikoverteilung, anhand derer sich einige wesentliche Ursachen der Krise aufzeigen lassen. Daher soll diese als Grundlage für die nachfolgende Diskussion der Ursachen und Auswirkungen sowie der Entwicklung der Krise zusammengefasst werden.

Der Originator kann durch den Verkauf von Kreditforderungen an eine Zweckgesellschaft im ersten Schritt seine Bilanz von den verbundenen Kreditrisiken befreien und mit den Provisionszahlungen sichere Erträge generieren. Die Ausfallrisiken werden bei der Tranchierung restrukturiert und mit der Emission von Wertpapieren an die Investoren weitergegeben. So tragen die Zweckgesellschaften keine Risiken, während die Investoren der Wertpapiere anhand unsicherer Zinszahlungen an den Ausfallrisiken und den entsprechenden Ertragschancen partizipieren. Die Sponsorbanken als Gründer der Zweckgesellschaften übernehmen bei einer fristenkongruenten Finanzierung durch ABS, CDOs oder CDOs² ebenfalls keine Risiken und generieren durch Beratungshonorare für die Unterstützung bei der Organisation und Durchführung der Verbriefungstransaktionen sichere Erträge. Erst bei einem Einbehalten der Equity-Tranche übernehmen die Hypotheken- oder Investmentbanken einen Teil des Kreditausfallrisikos, jedoch ebenfalls die Chance auf sehr hohe Renditen.

Das bei einer nicht-fristenkongruenten Finanzierung zusätzlich entstehende Liquiditätsrisiko übernehmen die Sicherungsgeber durch die Vergabe von Liquiditätslinien gegen Erhalt einer entsprechenden Prämie. In den meisten Fällen engagieren sich dabei die Sponsorbanken als Sicherungsgeber, die ihre Liquiditätslinien gemäß den Regulierungsvorschriften von Basel II unter bestimmten Bedingungen nicht, jedoch maximal zu 4% mit Eigenkapital unterlegen müssen (Hartmann-Wendels 2007). Wird ein Forderungsportfolio durch ein mit Marktwerttriggern ausgestattetes Structured Investment Vehicle revolvierend finanziert, so reduziert sich das Liquiditätsrisiko für den Sicherungsgeber erheblich, da für solche Conduits meist nur partielle Liquiditätslinien vergeben werden. Im Ausgleich übernehmen jedoch die Investoren der Equity-Tranche ein zusätzliches Kursrisiko, das im Falle eines Zwangsverkaufs des unterliegenden Portfolios bei Unterschreiten der Marktwerttrigger entsteht.

Als Basis für die Betrachtung der Krise ist darüber hinaus auch eine genauere Analyse der Risiken strukturierter Wertpapiere wie CDOs und CDOs² erforderlich. Einerseits wird durch eine wiederholte Kombination verschiedenartiger Wertpapiere in den unterliegenden Portfolios die Transparenz über Art und Qualität der letztendlich unterliegenden

Kreditforderungen zunehmend erschwert. Andererseits verändern sich dadurch auch die Faktoren, die zu Ausfällen der Wertpapiere führen können. So werden die Ausfälle einfach strukturierter Finanztitel wie ABS nur durch die Höhe der Ausfälle im unterliegenden Forderungsportfolio beeinflusst. Die Ausfälle mehrfach strukturierter Wertpapiere, denen beispielsweise ABS aus mehreren Emissionen unterliegen, basieren jedoch zusätzlich auf der Verteilung der Ausfälle der letztendlich unterliegenden Forderungen (Fender et al. 2008, Whetten und Adelson 2005). Je gleichmäßiger sich Ausfälle einer bestimmten Höhe auf die Forderungen verteilen, desto mehr ABS-Emissionen partizipieren an diesen Ausfällen und desto mehr Ausfälle können durch die nachrangigen Tranchen dieser ABS kompensiert werden. Damit sinkt die Ausfallswahrscheinlichkeit für CDO- und CDO²-Emissionen, denen die entsprechend vorrangigeren Tranchen der ABS-Emissionen unterliegen. Konzentrieren sich die Ausfälle in derselben Höhe jedoch nur auf einen kleinen Teil der unterliegenden ABS, so können die Ausfälle auch auf vorrangige ABS-Tranchen und damit ebenfalls auf die CDO und CDO²-Emissionen übergreifen. Somit besteht bei mehrfach tranchierten Finanzprodukten ein nichtlinearer Zusammenhang zwischen dem Ausfallrisiko des unterliegenden Forderungsportfolios und dem Ausfallrisiko der emittierten Wertpapiere (Basel Committee on Banking Supervision 2005).

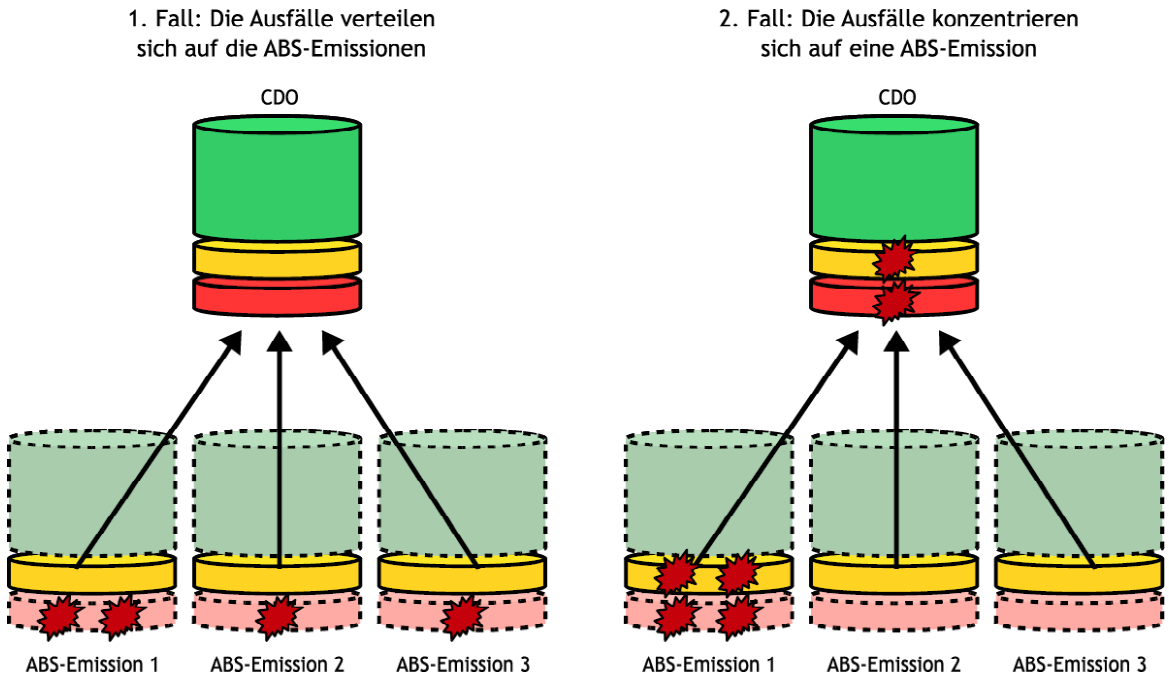

Abb. II-5: Auswirkung der Verteilung von Kreditausfällen auf das Risiko von CDO[(2)]

Das folgende, sehr vereinfachte Beispiel sowie Abb. II-5 illustrieren diesen Sachverhalt: Es wird eine CDO-Emission angenommen, die aus einer Equity-, einer Mezzanine- und einer Senior-Tranche besteht. Dieser CDO-Emission unterliegen drei Mezzanine-Tranchen aus drei verschiedenen ABS-Emissionen. Weiterhin wird angenommen, dass die Equity- und Mezzanine-Tranchen aller drei ABS-Emissionen jeweils den Ausfall von zwei unterliegenden Kreditforderungen kompensieren können. Der Ausfall einer weiteren Kreditforderung greift somit auf die nächste Tranche über. Die Equity- und Mezzanine-Tranche der CDO-Emission können jeweils den Ausfall einer der letztendlich unterliegenden Kreditforderung kompensieren. Verteilen sich nun die Ausfälle von vier Kreditforderungen, welche in Abb. II-5 durch rote Sterne dargestellt sind, gleichmäßig auf die drei ABS-Emissionen, so kompensieren die drei ABS-Equity-Tranchen die Ausfälle vollständig. Folglich partizipieren die Mezzanine-Tranchen der ABS und damit auch die gesamte CDO-Emission nicht an den Kreditausfällen. Betreffen die Ausfälle der vier Kreditforderungen jedoch nur eine der drei ABS-Emissionen, fallen sowohl die Equity- als auch die Mezzanine-Tranche der betroffenen ABS-Emission sowie folglich auch die der CDO-Emission vollständig aus.

Zudem tritt bei der Tranchierung von Wertpapieren grundsätzlich ein Hebeleffekt auf. Je niedriger eine Tranche rangiert und je geringer das Volumen dieser Tranche ist, desto höher ist die Wahrscheinlichkeit eines Ausfalls aller Zins- und Tilgungszahlungen für die Investoren dieser Tranche (Deutsche Bundesbank 2007, Fender und Mitchell 2005). So bedeutet ein Ausfall von 1% im Forderungsportfolio bereits einen Ausfall von 20% für eine Equity-Tranche, die 5% des unterliegenden Forderungsvolumens umfasst. Gleichzeitig entspricht die Wahrscheinlichkeit für einen vollständigen Ausfall einer solchen Equity-Tranche der Wahrscheinlichkeit, dass mindestens 5% der Forderungen im unterliegenden Portfolio ausfallen. Dieser Hebeleffekt kann sich durch eine mehrfache Tranchierung insbesondere bei Kombination mehrerer niederer Tranchen im unterliegenden Portfolio verstärken. Gepaart mit dem oben beschriebenen, nichtlinearen Zusammenhang zwischen dem Ausfallrisiko des unterliegenden Forderungsportfolios und dem Ausfallrisiko der emittierten Wertpapiere können sich als Konsequenz Veränderungen der Ausfallrisiken eines Forderungsportfolios verstärkt auf die Ausfallrisiken mehrfach strukturierter Wertpapiere wie CDOs und CDOs² auswirken (Whetten und Adelson 2005). Da das Rating solcher Papiere auf den erwarteten Verlusten der einzelnen Tranchen basiert (Fender et al. 2008), kann eine Erhöhung der Ausfallrisiken im unterliegenden Forderungsportfolio zu überproportionalen Herabstufungen des Ratings der emittierten Wertpapiere führen.

Ein weiterer Effekt ist mit der Korrelation der Ausfälle der Kredite im unterliegenden

Portfolio verbunden. Wie Abb. II-6 illustriert, ähnelt die typische Verlustverteilung einer CDO- beziehungsweise CDO²-Emission bei einer angestrebten kleinen Korrelation einer Binomialverteilung. Mit zunehmender Korrelation erhöht sich die Dichte an den Rändern der Verlustverteilung bei gleichbleibendem Erwartungswert (Fender und Mitchell 2005, Fender et al. 2008). Je höher die Korrelation der Ausfälle der Kredite, desto höher ist die Wahrscheinlichkeit, dass keine oder alle Kredite ausfallen. Als Konsequenz können sich die separat betrachteten Ausfallrisiken der Senior-Tranchen mit steigender Korrelation erhöhen, während die Mezzanine-Tranchen durch die höhere Wahrscheinlichkeit für eine vollständige Amortisation der unterliegenden Forderungen profitieren können (vgl. Abb. II-5) (Fender und Mitchell 2005, Fender et al. 2008, Krahnen und Wilde 2008). Die steigende Korrelation der Ausfälle der unterliegenden Assets kann also eine Verschiebung der Risiken von den nachrangigen Tranchen auf die vorrangigen Tranchen verursachen.

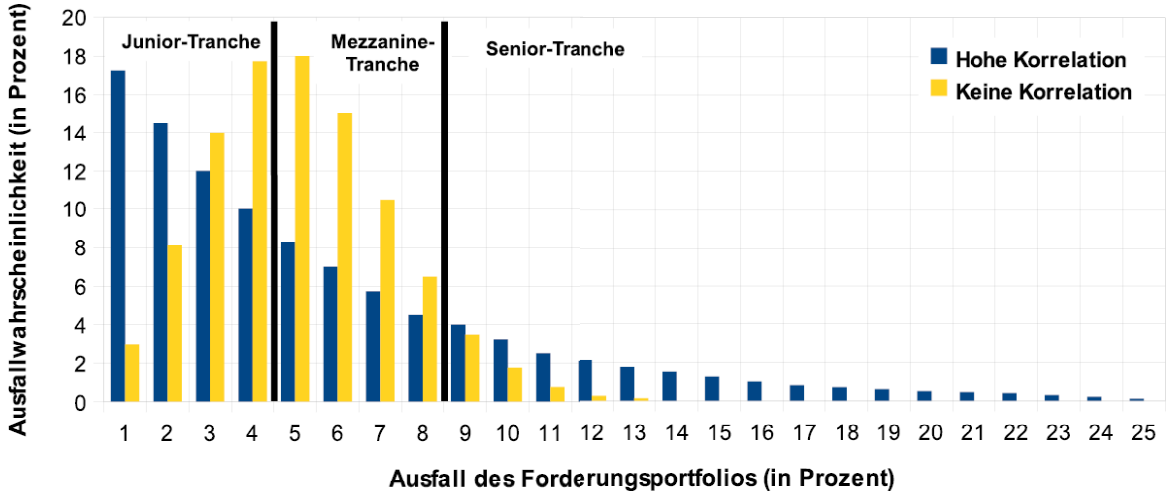

Abb. II-6: Verteilung der Ausfallswahrscheinlichkeiten (Fender und Mitchell 2005)

Daraus leitet sich insbesondere für die Risikostruktur von CDOs und CDOs², denen das Ziel einer umfassenden Diversifikation durch die Kombination verschiedener Forderungsarten mit unterschiedlichen Ausfallrisiken im unterliegenden Portfolio zugrunde liegt, eine wichtige Eigenschaft ab. So erweisen sich diese Wertpapiere mit steigender Diversifikation der unterliegenden Forderungen als zunehmend stabiler gegenüber unsystematischen Risikofaktoren, die nur wenige Assets betreffen und damit eine geringere Korrelation der Forderungen im unterliegenden Portfolio bedeuten. Dabei steigt die Stabilität gegenüber solchen unsystematischen Risikofaktoren auch mit dem Rang der Tranche. Dagegen zeigen vorrangige Tranchen von CDO- und CDO²-Emissionen eine

größere Sensibilität für systematische Risikofaktoren, die durch die Auswirkungen auf einen Großteil der Assets eine hohe Korrelation bedingen (Basel Committee on Banking Supervision 2005). Ein Anstieg der Korrelation bewirkt also bei Senior-Tranchen von CDO- und CDO²-Emissionen den größten relativen Anstieg der zu erwartenden Verluste (Krahnen und Wilde 2008).

Stellt man die beschriebene Risikostruktur mehrfach tranchierter Finanzprodukte den Ratingmethoden für solche Wertpapiere gegenüber, so zeigen sich zusätzliche Modellrisiken. In diesem Zusammenhang wird unter einem “Modellrisiko” das Risiko verstanden, dass die Ratingmodelle und damit auch das Rating selbst die Risikostruktur eines strukturierten Wertpapiers fehlerhaft beziehungsweise nicht unter Berücksichtigung aller Aspekte abbilden (Fender und Mitchell 2005). Für die Ermittlung der Ratings werden in der Regel die erwarteten Verluste beziehungsweise die Ausfallwahrscheinlichkeiten der einzelnen Tranchen separat bestimmt sowie Annahmen über die Ausfallkorrelationen der unterliegenden Forderungen zugrunde gelegt. Dementsprechend werden die Ratings der einzelnen Tranchen erheblich von den Annahmen über die Korrelationen beeinflusst (Fender und Mitchell 2005). Zudem basieren zahlreiche Daten für die Berechnung der Ratings auf der Extrapolation historischer Daten. Bis zur Entstehung der Krise lagen jedoch nur wenige historische Daten über die Entwicklung dieser neuartigen Finanzprodukte vor (Fender und Mitchell 2005). Diese Daten spiegelten keine Erfahrungswerte über das Eintreten systematischer Risikofaktoren wie Marktkrisen wider. So waren die historischen Korrelationen der Kreditausfälle sehr gering. Da die Ratings keine Unterschiede in der Sensibilität für systematische Risikofaktoren (Basel Committee on Banking Supervision 2005) und damit für Veränderungen von Korrelationen abbilden, ergibt sich hieraus ein Modellrisiko. Zudem wurden vor der Krise bei der Erstellung und Vergabe von Ratings für ABS, CDOs und CDOs² keine Markt- und Liquiditätsrisiken berücksichtigt (Basel Committee on Banking Supervision 2005, Deutsche Bundesbank 2007).

1.4 Die Krise auf den Kreditmärkten

1.4.1 Der amerikanische Immobilienmarkt als Ausgangspunkt der Krise

Den Ausgangspunkt der Finanzkrise bildeten der amerikanische Immobilienmarkt und insbesondere das Segment bonitätsschwacher Schuldner. Nach den Terroranschlägen des 11.09.2001 verfolgte die amerikanische Notenbank eine strikte Niedrigzinspolitik zur Vorbeugung der damals drohenden Rezession. Von den Möglichkeiten günstiger Refinanzierungen profitierte auch der US Hypothekenmarkt. Dabei erwiesen sich insbe-

sondere Immobilienkredite als sinnvolle Finanzinvestition für die Hypothekenbanken, da die Preise auf dem amerikanischen Immobilienmarkt während der letzten Jahre konstant, in manchen Regionen um bis zu 250% innerhalb von sechs Jahren, gestiegen waren (S&P/Case-Shiller 2009).

Im Zeitverlauf erweiterten die Hypothekenbanken die Kreditvergabe auf das Segment der einkommensschwachen Kunden mit problematischer Kredithistorie (*subprime*) (Gramlich 2007). Das hohe Ausfallrisiko dieser Schuldner sollte durch die steigenden Immobilienpreise kompensiert werden. Folglich verzichteten die Banken auf das strikte Einhalten von Kriterien der individuellen Bonitätsprüfung wie Arbeitsplatzsicherheit, Einkommensnachweise und der Darlegung von sonstigem als Sicherheit dienenden Vermögen (Hemmerich 2008). Die auf Bezieher von kleinen und mittleren Einkommen zugeschnittenen Immobilienfinanzierungen ermöglichten es weiten Teilen der amerikanischen Bevölkerung, in den Kreis der *home owner* aufzusteigen. Gerade in den Vereinigten Staaten ist dieser Status ein nach außen getragenes Zeichen des beruflichen Erfolgs und der gesellschaftlichen Anerkennung (Rohe et al. 2002). Die Bedeutung von Wohneigentum als persönliches Statussymbol hat nachhaltige Auswirkungen auf das Angebot an Immobilienfinanzierungen. Festdarlehen ohne regelmäßige Tilgungszahlungen ermöglichen die Finanzierung von größeren Immobilienobjekten in teureren Wohngegenden als Demonstration des gesellschaftlichen Aufstiegs. Eine stabile Werthaltigkeit der Immobilien ersetzt die Tilgungsleistungen bis zum Verkauf des Objekts und der Rückzahlung des Kredits. Die monatliche Belastung durch Zinszahlungen wird vermindert, indem eine kurze Dauer der Zinsbindung mit entsprechend niedrigen Zinsen vereinbart wird. Dieser Zusammenhang liegt ursächlich an der Natur der normalen Zinsstrukturkurve (Franke und Hax 2004), wonach der Kreditnehmer für langfristig geliehenes Geld einen höheren Zinssatz bezahlen muss als für kurzfristige Kredite. So stieg die Zahl der Eigenheimbesitzer von 64% (1995) auf über 69% (2005) (U.S. Census Bureau 2005). Insgesamt lag der Vergabe von Krediten an Kunden mit schlechter Bonität sowohl von Seiten des Kreditgebers wie auch des Kreditnehmers eine Spekulation auf stabile beziehungsweise steigende Immobilienpreise sowie auf niedrige respektive sinkende Zinsen zugrunde.

Im Laufe der Erschließung des Subprime-Segments entwickelten die Hypothekenbanken spezielle Finanzierungsprodukte, welche sich durch zunehmend geringere Kreditvergabestandards auszeichneten. Da diese Produkte zusätzliche Anreize für Immobilienfinanzierer mit problematischer Bonität schufen, wurden sie in der Presse auch als *NINJA-Loans* (no income, no job, no assets) bezeichnet (TSI-Arbeitsgruppe 2008). Mit *Hybrid Adjustable Rate Mortgages* (Hybrid ARMs) boten die Banken ein

Produkt an, bei dem zu Beginn ein kurzfristig fester, niedriger Zins gezahlt wurde, bevor die variable Zinsanpassung einsetzte (TSI-Arbeitsgruppe 2008, Krinsman 2007). Bei *Payment Options ARMs* waren zu Beginn nur sehr niedrige Zinsen (sogenannte *teaser-rates*) deutlich unterhalb der Zinskosten fällig, wodurch der auszahlbare Darlehensbetrag anfänglich weiter erhöht wurde (TSI-Arbeitsgruppe 2008). Das *Piggyback-Darlehen* richtete sich an Kreditnehmer gänzlich ohne Eigenkapital. Mit diesem Produkt wurde zusätzlich zum „normalen“ Darlehen mit beispielsweise 80% Beleihung der Immobilie ein zweites Darlehen angeboten, das die restlichen 20%, die normalerweise als Eigenkapital aufgebracht werden, abdeckte (TSI-Arbeitsgruppe 2008). Bei *Stated-income-loans* beziehungsweise *Self Certification* verzichtete der Kreditgeber auf die Überprüfung der Kundenangaben und damit auf die Einschätzung der Schuldendienstfähigkeit. Dieses Produkt wurde in den Medien auch als *liar loan* bezeichnet (Isidore 2007, TSI-Arbeitsgruppe 2008).

1.4.2 Der Beginn der Krise am amerikanischen Immobilienmarkt

Der Beginn der Krise liegt im Jahr 2006. Die übermäßigen Preissteigerungen vor allem in Staaten wie Kalifornien, Florida und Arizona führten zu einem landesweiten Angebotsüberhang neuer Wohnobjekte und zu einem an Dynamik verlierenden Immobilienmarkt (Hemmerich 2008). In der zweiten Hälfte des Jahres 2006 sanken die jährlichen Steigerungsraten der Immobilienpreise von 10% auf 4,7% (Krinsman 2007). Bei einem weiteren Abrutschen drohten die Kreditrisiken aufgrund der verminderten Werthaltigkeit der Wohnobjekte nicht mehr kompensiert werden zu können. Begleitet wurden diese Trends von Zinserhöhungen der amerikanischen Notenbank von 1% in 2003 auf über 5% in 2006. Da Hypothekenfinanzierungen oftmals mit einer kurzen Zinsfestschreibung gefolgt von einer variablen Verzinsung ausgestaltet waren, wurden viele Privathaushalte von den Zinserhöhungen erfasst. In Folge dessen kam es zu einem starken Anstieg der Kreditausfallraten im zweiten Halbjahr 2006 (Krinsman 2007).

Grundsätzlich sind die Hypothekenbanken bei Zahlungsschwierigkeiten der Kreditnehmer zur Verwertung der Immobilie in Form der Zwangsversteigerung berechtigt. Fällt jedoch der aktuelle Wert einer Immobilie unter den Nominalwert des ursprünglichen Kredits, kann alleine mit der Verwertung der Immobilie nur ein Teil des Darlehensbetrags getilgt werden. Da sich die Kreditnehmer des Subprime-Segments durch eine schlechte Bonität auszeichnen, kann die Hypothekenbank in der Regel auch durch Rückgriffnahme des Schuldners nicht den restlichen Darlehensbetrag decken. Für die Banken bestand somit ein Anreiz, möglichst zeitig in das Gebäude zu vollstrecken, um sich vor weiteren Preisnachlässen und wachsenden Restschulden abzusichern. Ebenso

waren die meist einflussreichen Investoren der Equity-Tranchen, wie beispielsweise Hedge Fonds oder Investmentbanken daran interessiert, die Ausfälle durch frühzeitige Zwangsversteigerungen zu begrenzen. Zudem können amerikanische Immobilienkreditverträge mit speziellen Ausstiegsklauseln ausgestattet sein, so dass für manche Kreditnehmer sogenannte *walk away* Optionen bestehen. So kann der Schuldner insbesondere bei Zahlungsschwierigkeiten die Immobilie räumen und sie zur Tilgung des Kredits an die Hypothekenbank übertragen. In einigen Bundesstaaten wie z.B. Kalifornien kann die Bank jedoch nicht den Kreditschuldner in Rückgriff nehmen, auch wenn der Verkaufserlös der Immobilie nicht zur Deckung des Darlehensbetrags ausreicht (Hemmerich 2008). Dies schaffte im Einzelfall einen zusätzlichen Anreiz für eine schnelle Zwangsversteigerung. Im Ergebnis wurde der Abwärtstrend auf dem Immobilienmarkt durch die Zunahme an Zwangsversteigerungen (Gramlich 2007) weiter beschleunigt und die Kreditverluste aufgrund der mangelnden Rückgriffmöglichkeiten der Gläubiger auf die Schuldner erhöht.

1.4.3 Die Vertrauenskrise an den Geld- und Kapitalmärkten

Aufgrund vermehrter Kreditausfälle begann im Frühjahr 2007 eine Risikoneubewertung der Hypothekengeschäfte, mit der Folge steigender Risikoaufschläge vor allem in mezzaninen CDO- und CDO²-Tranchen (Borio 2008). Diese Phase der vorsichtigen Risikokorrekturen wurde durch den Niedergang zweier Hedge-Fonds der Investmentbank *Bear Stearns* im Juli 2007 abrupt verstärkt. Diese erlitten innerhalb weniger Wochen hohe Verluste aus Investments am amerikanischen Subprime-Hypothekenmarkt. Gleichzeitig erfolgten massive Ratingherabstufungen bislang hoch eingestufter CDO- und CDO²-Tranchen, die durch deren spezielle Risikostruktur zu erklären sind: Mit dem Sinken der Immobilienpreise und dem Steigen des Zinsniveaus drohte eine stark wachsende Ausfallrate der Immobilienkredite insbesondere im Subprime-Segment. Diese Entwicklung bedeutete in erster Konsequenz ein stark ansteigendes Ausfallrisiko für eine vergleichsweise geringe Anzahl der den CDOs und CDOs² insgesamt unterliegenden Forderungen. Damit wuchs auch das Ausfallrisiko zahlreicher CDO- und CDO²-Emissionen stark an. Zugleich stiegen durch die erhöhten Ausfallrisiken der Immobilienkredite im Subprime-Segment die Risikoprämien einiger Kreditinstitute, wodurch Erhöhungen der Ausfallrisiken in anderen Segmenten der Kreditmärkte, zum Beispiel für Kreditkartendarlehen oder Kraftfahrzeugkredite, vermutet wurden (Crouhy und Turnbull 2008). So stellte sich diese Entwicklung zunehmend als systematischer Risikofaktor heraus. Dies führte schließlich zu drastischen Korrekturen der Wertpapierratings und einem massiven Vertrauensverlust in diese Finanzprodukte. Dabei erwies

sich die fehlende Transparenz insbesondere der komplexen CDO- und CDO²-Transaktionen als Katalysator des Vertrauensschwundes (Crouhy und Turnbull 2008). Da durch die mehrfache Kombination verschiedener Forderungsarten mit unterschiedlichen Ausfallrisiken die Qualität der den Emissionen letztendlich unterliegenden Forderungen für die Investoren nicht oder nur schwer nachvollziehbar war, weitete sich der Vertrauensverlust auch auf Wertpapiere aus, denen überwiegend solide Kreditforderungen unterlegt waren.

In der Folge erlitten die Hypothekenmarktpapiere nach einer langen Phase der Wertstabilität massive Werteinbrüche (vgl. Abb. II-7*)*. Da diese Papiere in der Regel OTC gehandelt werden, konnte das fehlende Vertrauen nicht durch Anpassungsprozesse des Marktes mit daraus resultierenden Marktpreisen aufgefangen werden. Die Papiere wurden nicht mehr gehandelt, so dass rationale Marktpreise nicht ermittelbar waren. Daher waren die bilanziellen Verluste vieler Investoren, zu denen auch zahlreiche Investment- und Geschäftsbanken sowie staatliche Banken zählten (Rudolph 2008), lange Zeit nicht bestimmbar, denn die Bewertung dieser Papiere erfolgte gemäß der *International Financial Reporting Standards* (IFRS) nach dem beizulegenden Zeitwert (Institut der Wirtschaftsprüfer 2007).

Abb. II-7: Kursverluste am Beispiel eines ABS-Index zwischen Januar 2007 und Januar 2009

Diese Entwicklung hatte darüber hinaus auch erhebliche Auswirkungen auf die kurzfris-

tigen ABCPs. Der Wertverlust der von den SIVs gehaltenen Portfolios aus kapitalmarktnotierten Hypothekenmarktpapieren war teilweise so groß, dass die von den Marktwerttriggern gesetzten Schwellenwerte deutlich unterschritten wurden und keine Finanzierung durch Neuemission kurzfristiger ABCPs erfolgen durfte. In diesem als *trigger event* bezeichneten Vorfall mussten die Zweckgesellschaften ihr Portfolio am Kapitalmarkt veräußern, um aus den Verkaufserlösen die Investoren der auslaufenden ABCPs bedienen zu können (Crouhy und Turnbull 2008). Aufgrund des fehlenden Vertrauens in die Qualität sank jedoch die Nachfrage nach Hypothekenmarktpapieren bei stets steigendem Angebot, was einen erneuten Preisverfall nach sich zog und weitere trigger events auslöste. So verloren viele Wertpapiere, die noch Anfang 2007 mit einem „AAA“-Rating ausgezeichnet waren, bis Ende 2008 durchschnittlich bis zu 70% ihres Nominalwerts. ABS mit einem „BBB“-Rating verzeichneten durchschnittlich sogar einen Wertverlust von über 95% (siehe Abb. II-7). Dieser sich selbst verstärkende Verkaufsdruck und Wertverfall, der durch die eigentlich zur Stabilisierung eingesetzten Marktwerttrigger aufrecht erhalten wurde, führte schließlich zu einem Zusammenbrechen des Marktes (TSI-Arbeitsgruppe 2008). So konnten viele Wertpapiere nicht mehr oder nur mit erheblichen Wertverlust verkauft werden, wobei mit den Verkaufserlösen in den meisten Fällen nur ein Bruchteil der Zins- und Tilgungszahlungen für die Senior-Tranchen der ABCP-Emissionen gedeckt werden konnte. Durch diese Entwicklung griff die Vertrauenskrise auf den gesamten Markt für kurzfristige ABCPs über. Einige Sponsorbanken kauften die emittierten ABCPs zum Nominalwert auf und überführten diese samt den Kursverlusten und dem Risiko der zugrundeliegenden Kreditforderungen in ihre Bilanzen. Ziel dieser Maßnahmen war einerseits der Schutz der Investoren vor den Wertverlusten. Andererseits versuchten die Sponsorbanken das Vertrauen der Investoren zu erhalten. Einige der Sponsorbanken waren auch an Verbriefungstransaktionen anderer und bis dato von der Finanzkrise nicht betroffenen SPVs beteiligt, die diese dadurch vor den Konsequenzen eines weiteren Vertrauensverlusts der Investoren schützen wollten.

Trotz dieser Maßnahmen brach der Markt für kurzfristige Wertpapiere ein. Die für die Finanzierung notwendigen Neuemissionen revolvierender ABCPs fanden aufgrund der erhöhten Risikoaversion und wachsender Sorge bezüglich der Qualität der zugrunde liegenden Kreditforderungen keine Käufer mehr (Borio 2008). Ohne die Liquidität des Geldmarktes drohten die Zweckgesellschaften auszutrocknen, so dass Liquiditätslinien in Anspruch genommen werden mussten. Dadurch schlugen die Risiken, welche im Rahmen des Verbriefungsprozesses auf die Zweckgesellschaften transferiert wurden, in die Bankbilanzen der Sponsoren zurück (Deutsche Bundesbank 2007). Viele Sicherungsgeber konnten jedoch das Volumen der in Anspruch genommenen Liquiditätslinien

aufgrund der sehr geringen Pflicht zur Eigenkapitalunterlegung meist nicht aus eigener Kraft stemmen. So verkündete beispielsweise die mittelständische *IKB Deutsche Industriebank* bereits Ende Juli 2007 massive Liquiditätsprobleme und Gewinnwarnungen infolge vergebener Liquiditätsgarantien für ihr SPV *Rhineland Funding*. Auch die damalige *Landesbank Sachsen (Sachsen LB)* geriet im August 2007 aufgrund von Liquiditätsgarantien für ihr Conduit *Ormond Quay* in eine existenzbedrohende Schieflage. Infolge dessen stellte kurz darauf die *Sparkassen-Finanzgruppe* der *Sachsen LB* eine Kreditlinie in Höhe von 17,3 Mrd. Euro zur Verfügung (Sachsen LB 2007). Im Dezember 2007 wurde schließlich die endgültige Übernahme der *Sachsen LB* durch die *Landesbank Baden-Württemberg* (*LBBW*) beschlossen.

Derartige Vorgänge verstärkten das bereits vorherrschende Misstrauen in der gesamten Bankenwelt. Intransparenz und die resultierende Ungewissheit über mögliche Risiken, beispielsweise aufgrund von Direktinvestments der Banken in die Papiere oder der indirekten Risikobeteiligung anhand von zugesicherten Liquiditätslinien, ließen die Liquidität an den globalen Finanzmärkten versiegen (Deutsche Bundesbank 2007). Der generelle Vertrauensschwund im Finanzsystem ermöglichte ein Übergreifen der Krise von den Subprime-Märkten auf den gesamten Finanzsektor, da die Liquiditätsversorgung der Banken untereinander am Interbankengeldmarkt versiegte. Die resultierende Zurückhaltung bei der Vergabe neuer Kredite und steigende Risikoaufschläge von teilweise 300 bis 400 Basispunkten (entspricht 3 - 4%; vgl. Abb. II-8) drohten nun das allgemeine Wirtschaftswachstum zu bremsen.

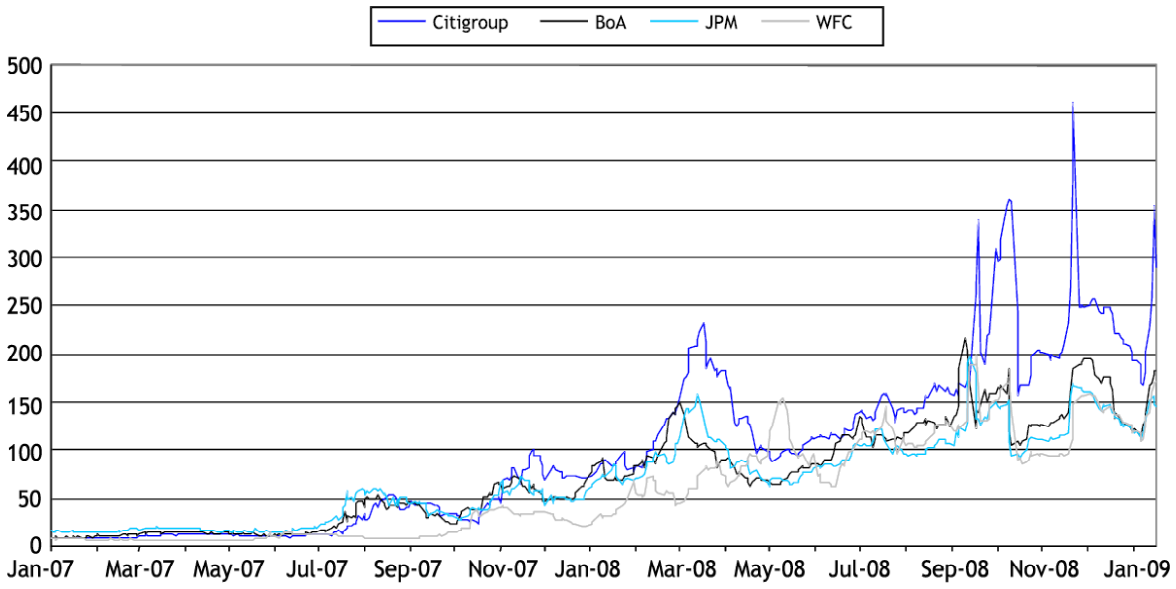

Abb. II-8: Anstieg der Refinanzierungszinsen zwischen Januar 2007 und Januar 2009

Um den Interbankengeldmarkt zu beleben und ein Übergreifen der Krise auf die Realwirtschaft zu begrenzen, reagierten die Notenbanken mit kurzfristigen Liquiditätsspritzen in großem Umfang (Crouhy und Turnbull 2008). Ende 2007 stellte beispielsweise die *Europäische Zentralbank* den Banken Liquidität in Höhe von knapp 350 Mrd. Euro zur Verfügung (o.V. 2007). Dennoch führten diese Entwicklungen zu existenzbedrohenden Schieflagen einiger Finanzinstitute. So konnte sich beispielsweise die britische Hypothekenbank *Northern Rock*, die in hohem Maße auf den Interbankengeldmarkt angewiesen war, nicht mehr refinanzieren. Aufgrund dessen verloren zahlreiche Anleger das Vertrauen und zogen im September 2007 ihre Einlagen bei der Bank ab. Im Februar 2008 folgte schließlich die Verstaatlichung von *Northern Rock*.

Infolge der Finanzkrise war der Kapitalmarkt einer generellen Neubewertung von Risiken ausgesetzt. Da von dieser Umorientierung übergreifend alle Anlage- und Bonitätsklassen weltweit betroffen waren, brachen auch Verbriefungsgeschäfte ein, denen keine Kredite im Subprime-Segment unterlagen. Als hauptsächliche Ursache und Problem zeigte sich dabei das fehlende Vertrauen insbesondere zwischen den Banken aufgrund der mangelnden Transparenz (Franke und Krahnen 2007). Daneben führten die umfassenden Kurseinbrüche der strukturierten Wertpapiere bei zahlreichen Investoren zu massiven Verlusten. Dazu zählten neben Investmentbanken, Hedge Fonds und Hypothekenbanken beispielsweise auch Rentenfonds und Versicherungsgesellschaften, die überwiegend in Wertpapiere der Senior- und Mezzanine-Tranchen investiert hatten. Schließlich zählten auch die Ratingagenturen zu den Geschädigten der Krise, da viele Banken und Investoren das Vertrauen in die Qualität der Ratings verloren.

Einen vorläufigen Höhepunkt erreichte die Krise am 22.01.2008, als Meldungen von erneuten Milliardenabschreibungen zu den größten Kurseinbrüchen an den internationalen Börsen seit dem 11.09.2001 führten. Das steigende Volatilitätsniveau auf den Kapitalmärkten und fortlaufende Rekordabschreibungen aufgrund von Investments im amerikanischen Immobilienmarkt veränderten die Finanzwelt nachhaltig. Zahlreiche Banken, Großinvestoren und Anleiheversicherer wurden im Zuge der Krise unmittelbar von der Insolvenz bedroht. In Deutschland sind bspw. die Notlagen der Landesbanken *WestLB* und *BayernLB* sowie die zum Schutz dieser und weiterer Banken konstruierten Rettungsschirme in Höhe mehrerer Milliarden Euro bekannt geworden.

In den USA begann im März 2008 eine komplette Neuordnung des Investmentbanking, als das Kreditinstitut *JPMorgan Chase* mit staatlicher Unterstützung die notleidende Investmentbank *Bear Stearns* übernahm. Im September 2008 musste die Investmentbank *Merrill Lynch*, die bis Juli 2008 Abschreibungen in Höhe von mehr als 40 Mrd.

Dollar insbesondere aufgrund eines 30 Mrd. Dollar Investments in hochrangige CDO-Tranchen verzeichnete, an die *Bank of America* notverkauft werden (o.V. 2008b). Keine Rettung gab es dagegen für die durch Investments in niedere ABS-Tranchen stark angeschlagene Investmentbank *Lehman Brothers*, deren Insolvenz am 15.09.2008 zu einem Kollaps des Interbankengeldmarkts führte. Mit dieser Entwicklung griff die Krise auch auf Banken über, die überwiegend in solide Anlagen investierten. So drohte beispielsweise der *Hypo Real Estate Holding*, deren deutsch-irische Tochter *Depfa Bank* in langfristige Staatsanleihen mit guter Bonität investierte und diese Anlagen kurzfristig am Interbankengeldmarkt refinanzierte, aufgrund der mangelnden Liquidität auf dem Interbankengeldmarkt und den daraus resultierenden Refinanzierungsschwierigkeiten die Insolvenz (Hesse 2008). Auch bei den isländischen Banken *Kaupthing Bank*, *Glitnir* und *Landsbanki* führten die Folgen der Insolvenz von *Lehman Brothers* zu existenziellen Liquiditätskrisen. Nach der Verstaatlichung dieser Banken wurde im Oktober 2008 die faktische Zahlungsunfähigkeit des Staates Island bekannt. Zu diesem Zeitpunkt überstieg die Bilanzsumme der Banken das Bruttoinlandsprodukt Islands um das Neunfache (Osman 2008). Als letzte verbliebene Investmentbanken gaben *Goldman Sachs* und *Morgan Stanley* Ende September 2008 freiwillig ihren Sonderstatus als Investmentbanken auf und begaben sich unter die Aufsicht der US Notenbank, um so Zugang zu Zentralbankgeld zu erhalten und durch Übernahmen weiterer Geschäftsbanken neue Einkommensquellen erschließen zu können (Bräuer und Schrörs 2008).

Des Weiteren litten Unternehmen der Realwirtschaft unter mangelnden Finanzierungsquellen und vermehrten Umsatzrückgängen. Wie aus Abb. II-8 ersichtlich verzeichnen die Refinanzierungszinsen vieler Banken seit der zweiten Jahreshälfte 2007 einen starken Anstieg. Dies führte zu einer Erhöhung der Kreditzinsen und verteuerte für Finanzdienstleister, Unternehmen der Realwirtschaft sowie Privatpersonen die Aufnahme von Fremdkapital. Folglich reduzierten sich das Investitionsvolumen vieler Unternehmen und damit letztendlich auch die Ausgaben für Konsumgüter. So erfuhr beispielsweise die Automobilbranche aufgrund der engen Verflechtung mit den Finanzmärkten und der sinkenden Nachfrage bereits im vierten Quartal 2008 enorme Absatzschwierigkeiten. Jedoch waren auch branchenübergreifend Unternehmen jeglicher Größe von den extremen Marktbedingungen betroffen. Insgesamt wird für das Jahr 2009 mit einem Rückgang des Bruttoinlandsprodukts um bis zu 3% gerechnet. Die vom amerikanischen Subprime-Markt ausgehende Krise entwickelte sich zum makroökonomischen Risikofaktor, der nicht nur die Finanzwelt neu ordnete, sondern die globale Wirtschaft in eine Rezession stürzt, deren Ausmaß aktuell nicht abgeschätzt werden kann.

1.5 Anreizstrukturen

Das teilweise fragwürdige Verhalten einiger wichtiger Marktakteure, welches entscheidend zur Entstehung und zum Verlauf der Krise beigetragen hat, kann unter anderem durch einige Anreizsysteme, welche für die zugrunde liegenden Verbriefungsstrukturen charakteristisch sind, erklärt werden (Ashcraft und Schuermann 2008). Diese Anreizstrukturen resultieren dabei einerseits aus der oben beschriebenen Ertrags-/Risikoverteilung sowie andererseits aus einer asymmetrischen Informationsverteilung zwischen vielen Beteiligten. Im Folgenden sollen die wichtigsten Anreizstrukturen skizziert werden.

Nachhaltige Anreizsysteme sind in der Regel gekennzeichnet durch ein symmetrisches Verhältnis von Chancen und Risiken. Dadurch entstehen Beziehungen, die sich für beide Parteien im Erfolgsfall als win-win und andernfalls als lose-lose darstellen. Die Strukturen, die zur globalen Finanzkrise führten, zeichneten sich hingegen dadurch aus, dass diese Verteilung divergierte und win-lose Beziehungen ermöglicht wurden. Dies wird besonders deutlich anhand der Beziehungsentkopplung von Kreditgeber und Schuldner. Während das traditionelle Kreditverhältnis langfristig ausgelegt ist, waren die Hypothekenbanken mit der Möglichkeit, vergebene Kredite gegen eine sichere Prämie zur Verbriefung an SPVs zu verkaufen, zunehmend an kurzfristigen Provisionserträgen interessiert. Da die Hypothekenbanken mit dem Verkauf auch die Risiken der Kreditforderungen transferierten, entstand zudem der Anreiz, die Kreditvergabe unter anderem dadurch zu steigern, dass sie auf einkommensschwache Gesellschaftsschichten mit schlechter Bonität ausgeweitet wurde. Dabei konnten die Hypothekenbanken einerseits von den meist nur sehr eingeschränkten Erfahrungen der einkommensschwachen Kunden im Bereich der Finanzierung sowie andererseits von der damaligen Niedrigzinspolitik der amerikanischen Notenbank profitieren. Zwischen Originator und Käufer der Kreditforderungen bestand also ein klassisches Problem des Moral Hazard (Rudolph 2008).

Eine ähnlich problematische Anreizstruktur zeigt sich in der Beziehung zwischen Emittent und Investor. Da sich viele Investoren auf die Ratings verlassen haben, kann grundsätzlich davon ausgegangen werden, dass die Sponsorbanken als Gründer der emittierenden SPVs ein fundierteres Verständnis über die Chancen und Risiken der komplexen und teilweise intransparenten Wertpapiere besaßen als die Investoren. Nun konnten die SPVs mit dem Verkauf der Wertpapiere die Risiken der unterliegenden Kreditforderungen an die Investoren übertragen und die Sponsorbanken in ihrer Funktion als Arrangeure sichere Erträge erzielen. Daher sank ebenfalls für die Emitten-

ten der Anreiz, ein entsprechendes Qualitätsniveau der unterliegenden Kreditforderungen und damit auch der Wertpapiere sicherzustellen. In Extremfällen konnten sich die win-lose Beziehungen sogar dahingehend steigern, dass die Sponsorbanken von einer Krise profitieren konnten, indem sie zwar die Wertpapiere bewarben und an Investoren veräußerten, jedoch selbst gegen diese Papiere Wetten eingingen. So wurde *Goldman Sachs* Ende 2007 vorgeworfen, an der Krise mehr zu verdienen, als es bei einem Ausbleiben der Krise möglich gewesen wäre (Kelly 2007).

Darüber hinaus erweist sich auch das Verhältnis zwischen Investor und Ratingagentur als problematisch. Einerseits stützten sich insbesondere die Investoren bei ihren Investitionsentscheidungen überwiegend auf die Ratingurteile (Rudolph 2008). Andererseits beziehen die Ratingagenturen jedoch von den Sponsorbanken ihre Vergütung für das Erstellen der Ratings der strukturierten Wertpapiere (Jäger und Voigtländer 2008). Entsprechend diesem Dilemma dürfte es eher dem Anreiz der Ratingagenturen entsprochen haben, die Kundeninteressen der Sponsorbanken für ein möglichst gutes Rating der Wertpapiere zu befriedigen und damit hohe Gebühren zu vereinnahmen, anstatt ein objektives Rating im Interesse der Investoren zu erstellen.

Letztendlich haben nahezu alle Beteiligten des Systems Strukturen geschaffen, die ihren Anreizsystemen größtenteils entsprachen und sich wechselseitig bedingten. Diese zeichneten sich jedoch durch eine asymmetrische Verteilung von Chancen und Risiken aus und ermöglichten win-lose-Szenarien. So zerbrach beispielweise durch den Verkauf und die Verbriefung der Kreditforderungen die Beziehung zwischen Schuldnern und originären Gläubigern, wodurch Hilfsmaßnahmen bei Zahlungsproblemen der Kreditnehmer faktisch unmöglich wurden. Berücksichtigt man weiterhin, dass vor Ausbruch der Krise viele Marktakteure wie beispielweise Investment-/Sponsorbanken nicht unter der staatlichen Bankenaufsicht standen und somit weniger restriktiven Vorgaben unterlagen als bspw. Geschäftsbanken (Broome und Markham 2008), muss auch der Politik eine Mitverantwortung an der Krise angelastet werden. So verhinderte die US-Regierung im letzten Jahrzehnt viele Initiativen zu einer Regulierung dieses Sektors im Interesse der Wall Street.

Obwohl sicherlich das fragwürdige Verhalten vieler Akteure hauptsächlich auf die beschriebenen Anreizstrukturen zurückzuführen ist, kann ein rein opportunistisches Verhalten vermutlich nicht als einzige Ursache für die Entstehung und den Verlauf der Krise angeführt werden. Darüber hinaus scheinen nahezu alle Marktteilnehmer die tatsächlichen Risiken der Wertpapiere und Verbriefungsstrukturen fehlerhaft bewertet zu haben. Viele Sponsorbanken investierten auch in teilweise beträchtlichem Umfang in die

strukturierten Finanztitel (Rudolph 2008). Offensichtlich glaubten die Akteure, die komplexen Strukturen und Risiken beherrschen und steuern zu können.

1.6 Ausblick

Ohne Kenntnis genauer Zahlen sprach *Jochen Sanio*, Präsident der deutschen *Bundesanstalt für Finanzdienstleistungsaufsicht*, bereits Anfang August 2007 in Anspielung auf den Zusammenbruch des Bankensektors während der Großen Depression von der „schlimmsten Finanzkrise seit 1931“. Schätzungen belegen das gewaltige Ausmaß, welches die Subprime-Krise erreichen könnte. Der *Internationale Währungsfonds* bezifferte schon im September 2008 die durch die Finanzkrise verursachten Abschreibungsverluste im Finanzsektor auf insgesamt 1.300 Mrd. Dollar, wobei jedoch die gesamtwirtschaftlichen Verluste kaum bestimmbar seien (Lipsky 2008). Laut einer Studie des Hedge-Fonds *Bridgewater Associates* können die weltweiten Verluste des Finanzsektors unter ungünstigen Bedingungen noch auf 1.600 Mrd. Dollar steigen (o.V. 2008a). Diese Zahl beeindruckt im Hinblick auf die Tatsache, dass das gesamte Eigenkapital aller amerikanischen Finanzinstitute lediglich 1.200 Mrd. Dollar (Sinn 2008) beträgt, sowie, dass das kumulierte Volumen der im Jahr 2007 ausstehenden Subprime-Kredite mit 940 Mrd. Dollar und der im Jahr 2006 ausstehenden Subprime-Kredite mit 1.240 Mrd. Dollar beziffert wurde. Daneben ist zu vermuten, dass sich in Anbetracht des hohen Verschuldungsgrades der amerikanischen Bevölkerung die Verluste nicht auf den Hypothekenkreditmarkt beschränken, sondern auf andere Kreditmärkte wie Autofinanzierungen oder Kreditkarten übergreifen werden. Besonders deutlich wird dieses Verlustpotenzial in Anbetracht der 900 Mrd. Dollar an ausstehenden Kreditkartenschulden, welche im Gegensatz zu Hypotheken nicht durch unterliegende Vermögenswerte besichert sind (Koch 2008).

Grundsätzlich bilden sowohl der kurzfristige Liquiditätsbedarf der Banken, die beispielsweise Liquiditätslinien für Zweckgesellschaften vergeben haben, als auch die teilweise massiven Wertverluste der Wertpapiere die direkten Effekte der Krise. Die Bewertungsverluste führen primär zu Abschreibungen bei den Investoren, die nur im Falle eines Verkaufs der Wertpapiere tatsächlich realisiert werden. So kann davon ausgegangen werden, dass bei einer Erholung der Werte vermögensbesicherter Wertpapiere nur ein Teil der vorgenommenen Abschreibungsverluste letztendlich realisiert wird. Größere Auswirkungen sind durch die indirekten Folgen der Subprime-Krise zu erwarten. Einerseits wird durch die vorgenommenen Abschreibungen das Eigenkapital vieler Banken reduziert. Dies schränkt die Möglichkeit zur Vergabe neuer Kredite ein, da diese grundsätzlich mit Eigenkapital unterlegt werden müssen. Andererseits bewegen sich, wie aus

Abb. II-8 ersichtlich, die Refinanzierungszinsen vieler Banken seit der zweiten Jahreshälfte 2007 auf einem sehr hohen Niveau. Sollte sich diese Entwicklung in der Zukunft über einen längeren Zeitraum weiter fortsetzen oder sogar verstärken, ist mit einer noch deutlicheren Einschränkung der Kreditvergabemöglichkeiten der Banken zu rechnen. Die Banken müssen bereits heute und auch in Zukunft die Erwartungen der Investoren an vergleichsweise hohe Renditen erfüllen. Aufgrund dieser Erwartungen und der hohen Refinanzierungskosten werden die Banken zunehmend zu einer kurzfristigeren und damit günstigeren Aufnahme von Fremdkapital gezwungen sein. Folglich wird sich in diesem Fall der Anstieg der Zinsen für langfristige Darlehen fortsetzen, wodurch Investitionen der Realwirtschaft noch teurer werden oder sogar unterbleiben, der Konsum zunehmend zurückgeht und die Produktion weiter eingeschränkt werden muss. Eine solche Entwicklung würde die entstandene Rezession in noch unkalkulierbarem Maße verstärken.

Jedoch darf in der Diskussion nicht vergessen werden, dass die Strukturen des US-Hypothekenkreditmarktes zu den Wachstumstreibern der letzten Jahre gehörten. Die günstigen Bedingungen für Kreditvergaben stellten einen Investitionsanreiz für Unternehmen dar und ermöglichten den Anstieg der Konsumgüternachfrage. Besonders von diesem Wachstum profitierte jedoch die Finanzbranche selbst. So stiegen die Gewinne in den Jahren vor 2007 durchschnittlich um jährlich 19%. Es ist also fraglich, ob langfristig die Wohlfahrtsverluste durch die Subprime-Krise von den positiven Auswirkungen der vorangegangenen Wachstumsphase kompensiert werden können. Unfraglich ist dagegen, dass sich die Verteilung der Gewinne stark auf einige wenige Akteure in der Finanzbranche fokussierte, während sich die Verluste weltweit an den Kapitalmärkten und darüber hinaus in die Realwirtschaft streuen konnten (Hoffmann 2008).

Literatur (Kapitel II.1)

Ashcraft AB, Schuermann T (2008) Understanding the Securitization of Subprime Mortgage Credit. Federal Reserve Bank of New York Staff Reports No. 318

Basel Committee on Banking Supervision (2005) Credit Risk Transfer. Verfügbar unter: http://www.bis.org/publ/joint13.pdf, Abruf am: 16.01.2009

Borio C (2008) The Financial Turmoil of 2007–?: A Preliminary Assessment and Some Policy Considerations. Verfügbar unter: www.bis.org/publ/work251.pdf, Abruf am: 16.01.2009

Bräuer S, Schrörs M (2008) Wall Street beerdigt Kasino-Ära. Financial Times Deutschland (23.09.2008)

Broome L, Markham J (2008) Regulation of Bank Financial Service Activities. West Group, St. Paul MN

Crouhy M, Turnbull SM (2008) The Subprime Credit Crisis of 07. Verfügbar unter: http://ssrn.com/abstract=1112467, Abruf am: 16.01.2009

DeMarzo PM (2005) The Pooling and Tranching of Securities: A Model of Informed Intermediation. Review of Financial Studies 18:1–35

Deutsche Bundesbank (2007) Finanzstabilitätsbericht 2007. Verfügbar unter: http://www.bundesbank.de/download/volkswirtschaft/finanzstabilitaetsberichte/finanzstabilitaetsbericht2007.pdf, Abruf am: 16.01.2009

Fender I, Mitchell J (2005) Structured Finance – Complexity, Risk and the Use of Ratings. BIS Quarterly Review June 2005:67–79

Fender I, Tarashev N, Zhu H (2008) Credit Fundamentals, Ratings and Value-at-Risk: CDOs versus Corporate Exposures. BIS Quarterly Review March 2008:87–101

Franke G, Hax H (2004) Finanzwirtschaft des Unternehmens und Kapitalmarkt. 5. Auflage, Springer, Berlin

Franke G, Krahnen JP (2005) Default Risk Sharing between Banks and Markets: The Contribution of Collateralized Debt Obligations. Verfügbar unter: http://www.nber.org/papers/w11741.pdf, Abruf am: 16.01.2009

Franke G, Krahnen JP (2007) Finanzmarktkrise – Ursachen und Lehren. Frankfurter Allgemeine Zeitung (24.11.2007)

Gramlich EM (2007) America’s Second Housing Boom. Verfügbar unter: http://www.urban.org/UploadedPDF/311418_Second_Housing_Boom.pdf, Abruf am: 16.01.2009

Hartmann-Wendels T (2007) Bestehen Aufsichtsdefizite im Bankensektor? Wirtschaftsdienst 87:640–643

Hemmerich F (2008) Vom US-Immobilienmarkt zur internationalen Finanzkrise. WISU – Das Wirtschaftsstudium 37(4):514–520

Hesse M (2008) Mit Anlauf in den Abgrund. Süddeutsche Zeitung (05.10.2008)

Hoffmann C (2008) Schnäppchen mit Risiko. Süddeutsche Zeitung (08.05.2008)

Institut der Wirtschaftsprüfer (2007) Positionspapier des IDW zu Bilanzierungs- und Bewertungsfragen im Zusammenhang mit der Subprime-Krise. Verfügbar unter: www.idw.de/idw/download/Subprime-Positionspapier.pdf?id=424920, Abruf am: 16.01.2009

International Finance Corporation (2008) Warehouse Line of Credit. Verfügbar unter: http://www.ifc.org/ifcext/gfm.nsf/AttachmentsByTitle/HF-WHL/$FILE/HF-WHL.pdf, Abruf am: 16.01.2009

Isidore C (2007) „Liar loans“: Mortgage woes beyond subprime. Verfügbar unter: http://money.cnn.com/2007/03/19/news/economy/next_subprime, Abruf am: 16.01.2009

Jäger M, Voigtländer M (2008) Hintergründe und Lehren aus der Subprime-Krise. IW-Trends 35(3):1–14

Jortzik S (2005) Semi-analytische und simulative Kreditrisikomessung synthetischer Collateralized Debt Obligations bei heterogenen Referenzportfolios. Verfügbar unter: http://webdoc.sub.gwdg.de/diss/2006/jortzik/jortzik.pdf, Abruf am: 16.01.2009

Kelly K (2007) How Goldman profited from subprime meltdown. The Wall Street Journal. Verfügbar unter: http://articles.moneycentral.msn.com/Investing/Extra/HowGoldmanProfitedFromSubprimeMeltdown.aspx, Abruf am: 16.01.2009

Koch M (2008) Der Kaufrausch ist beendet. Süddeutsche Zeitung (14.11.2008)

Krahnen JP, Wilde C (2008) Risk Transfer with CDOs. Verfügbar unter: http://www.finance.uni-frankfurt.de/wp/1652.pdf, Abruf am: 16.01.2009

Krahnen JP (2005) Der Handel von Kreditrisiken: Eine neue Dimension des Kapitalmarktes. Perspektiven der Wirtschaftspolitik 6:499–519

Krinsman AN (2007) Subprime mortgage meltdown – how did it happen and how will it end? The Journal of Structured Finance 13(2):13–19

Lipsky J (2008) The Global Economy and Financial Crisis. Verfügbar unter: http://www.imf.org/external/np/speeches/2008/092408.htm, Abruf am: 16.01.2009

o.V. (2007) Die EZB öffnet die Geldschleusen. Frankfurter Allgemeine Zeitung (19.12.2007)

o.V. (2008a) Kosten der Subprimekrise – 1 600 000 000 000 Dollar. Financial Times Deutschland (06.07.2008)

o.V. (2008b) Merrill Lynch saniert Bilanz mit milliardenschwerer Abschreibung. Financial Times Deutschland (29.07.2008)

o.V. (2008c) Viermetz bricht Lanze für die Börsen. Börsen-Zeitung (08.05.2008)

Osman Y (2008) Island warnt vor eigenem Bankrott. Financial Times Deutschland (08.10.2008)

Pfingsten A (2007) Wider den Regulierungsreflex. Wirtschaftsdienst 87:635–640

Rohe WM, Van Zandt S, McCarthy G (2002) Home ownership and access to opportunity. Home Studies 17(1):51–61

Rudolph B, Hofmann B, Schaber A, Schäfer K (2007) Kreditrisikotransfer – Moderne Instrumente und Methoden. Springer, Berlin

Rudolph B, Scholz J (2007) Pooling und Tranching im Rahmen von ABS-Transaktionen. Bank Archiv 55(7):538–548

Rudolph B (2008) Lehren aus den Ursachen und dem Verlauf der internationalen Finanzkrise. Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung 60: 713–741

S&P/Case-Shiller® (2009) U.S. National Home Price Index. Verfügbar unter: http://www2.standardandpoors.com/spf/pdf/index/CSHomePrice_History_112766.xls, Abruf am: 16.01.2009

Sachsen LB (2007) Ad-hoc Mitteilung vom 17.08.2007. Verfügbar unter: http://www.sachsenbank.de/sb/1000012130-de.html, Abruf am: 16.01.2009

Sinn H-W (2008) Wenn Banken mit Zitronen handeln. Börsen-Zeitung (26.04.2008)

Steiner M, Bruns C (2000) Wertpapiermanagement. Schäffer-Pöschel, Stuttgart

True Sale International (2009a) Asset Backed Securities – Was versteht man unter True Sale. Verfügbar unter: http://www.tsi-gmbh.de/abs-im-ueberblick/wasistabs/wasisttruesale.html, Abruf am: 16.01.2009

True Sale International (2009b) Glossar, Commercial Paper. Verfügbar unter: http://www.tsi-gmbh.de/abs-research-und-literatur/glossar.html#C, Abruf am: 16.01.2009

TSI-Arbeitsgruppe (2008) Die Krise auf den Kreditmärkten – Ursache, Wirkung und Folgerungen. Verfügbar unter: http://www.tsi-gmbh.de/fileadmin/tsi_downloads/ABS_Research/Aktuelle_Positionen_und_Research/20080507_TSI_Positionspapier.pdf, Abruf am: 16.01.2009

U.S. Census Bureau (2005) Housing Vacancies and Homeownership. Verfügbar unter: http://www.census.gov/hhes/www/housing/hvs/hvs.html, Abruf am: 16.01.2009

Whetten M, Adelson M (2005) CDOs-Squared Demystified. Verfügbar unter: http://www.securitization.net/pdf/Nomura/CDO-Squared_4Feb05.pdf, Abruf am: 16.01.2009

2 Beitrag: „*Die Verantwortung der Wirtschaftsinformatik für die Finanzmarktkrise*“

Autor:	Peter Bartmann
	Kernkompetenzzentrum Finanz- & Informationsmanagement, Lehrstuhl für BWL, Wirtschaftsinformatik, Informations- & Finanzmanagement (Prof. Dr. Hans Ulrich Buhl) Universität Augsburg, D-86135 Augsburg
	peter.bartmann@wiwi.uni-augsburg.de

Erschienen in:	Informatik-Spektrum, 32, 2, 2009, S. 146-152;
	B. Nietert ed., Die Eskalation der Finanz- zur Wirtschaftskrise, Fritz Knapp Verlag, Frankfurt am Main (Deutschland), 2011, S. 46-59.

Zusammenfassung:

Entscheidungsunterstützungssysteme (EUS) zählen zu den zahlreichen Errungenschaften der Wirtschaftsinformatik. Diese speziellen Anwendungssysteme unterstützen Führungskräfte aus dem mittleren und oberen Management bei Entscheidungsprozessen durch die Bereitstellung relevanter Informationen und die Generierung von Handlungsempfehlungen auf der Basis von Entscheidungsmodellen. Auch bei der Entstehung und Entwicklung der seit Juli 2007 vorherrschenden Finanzmarktkrise spielen diese eine Rolle. So stützen sich nahezu alle Marktakteure bspw. im Rahmen des Risikomanagements auf entsprechende Modelle und Anwendungssysteme. Vor diesem Hintergrund möchte dieser Beitrag die Rolle und Verantwortung der Wirtschaftsinformatik diskutieren sowie Lehren aus der Finanzmarktkrise für die IT-basierte Entscheidungsunterstützung ziehen.

2.1 Einleitung

Spätestens seit der Insolvenz der damals viertgrößten US-amerikanischen Investmentbank Lehman Brothers Inc. am 15.09.2008 (o.V. 2008a) hat sich die seit Juli 2007 virulent gewordene Finanzmarktkrise, deren Ausgangspunkt die Vergabe und Verbriefung zweitklassiger amerikanischer Hypothekendarlehen (Subprime) bilden, zur weltweiten Finanzmarktkrise entwickelt. Mit dem Übergreifen der Krise auf die Automobilbranche im Jahr 2008 (o.V. 2008b) und der Zahlungsunfähigkeit des Staates Island (o.V. 2008c) drohen die anhaltenden Turbulenzen an den Finanzmärkten eine globale Rezession und Weltwirtschaftskrise noch unkalkulierbareren Ausmaßes auszulösen.

Die Frage nach der Hauptverantwortung für die Krise ist dabei letztendlich nur sehr schwer zu beantworten. Nach Ausbruch der Krise entstand aufgrund der Stellungnahmen von vielen involvierten und betroffenen Marktteilnehmern der Eindruck, dass eine regelrechte Diffusion der Verantwortung erfolgte. So sehen sich viele Akteure als Opfer der für die Krise ursächlichen Strukturen und des Handelns der anderen Akteure. Nach Meinung des Autors kann in der Tat ein Haupt- oder sogar Alleinverantwortlicher nicht identifiziert werden. Nichtsdestotrotz haben alle Akteure wie z.B. Kreditnehmer, Hypothekenbanken, Investmentbanken, Investoren, Ratingagenturen, Regierungen und Notenbanken ihren Teil zu der Entstehung und dem Verlauf der Krise beigetragen. Alle Beteiligten des Systems haben mit dem Auf- und Ausbau von komplexen und oftmals intransparenten Verbriefungstransaktionen Strukturen geschaffen, die ihren individuellen Anreizsystemen größtenteils entsprachen und sich wechselseitig bedingten (Bartmann et al. 2009). Ob und inwieweit die Erschaffung dieser Strukturen auf ein unseriöses Profitstreben oder opportunistisches Verhalten der beteiligten Akteure zurückzuführen ist, bleibt spekulativ. Zumindest haben jedoch offensichtlich nahezu alle Marktteilnehmer die durch diese Strukturen induzierten Risiken fehlerhaft bewertet. Der Autor ist überzeugt, dass gerade aufgrund dieses Sachverhalts keiner der beteiligten Akteure auch nur partiell von seiner Verantwortung entbunden werden kann.

Die geschaffenen Verbriefungsstrukturen basieren nun zu einem großen Teil auf finanzmathematischen Modellen. Aufgrund der teilweise enormen Komplexität dieser Modelle und der großen Quantität der benötigten Informationen wird bspw. die Erstellung der Ratings oder das Risikomanagement der Akteure überwiegend mit IT-Unterstützung durch Einsatz von entsprechenden Entscheidungsunterstützungssystemen (EUS) betrieben. Damit spielen solche Modelle und EUS ebenfalls eine entscheidende Rolle für die Entstehung und den Verlauf der Finanzmarktkrise. Nun zeichnet gemäß ihrem Selbstverständnis gerade die Wirtschaftsinformatik u. a. für die Erfor-

schung, Konzeption, Entwicklung und Nutzung solcher Anwendungssysteme verantwortlich. Daher möchte dieser Beitrag die Frage diskutieren, inwieweit die Wirtschaftsinformatik als Interdisziplin und als angewandte Wissenschaft in diesem Zusammenhang ihrer Verantwortung gerecht geworden ist und in welchem Maße die Versäumnisse zur Entstehung der Finanzmarktkrise beigetragen haben.

Dazu skizziert der zweite Abschnitt die für die Diskussion relevanten Schwerpunkte und Aufgaben der Wirtschaftsinformatik. Auf dieser Grundlage soll im dritten Abschnitt die Verantwortung der Wirtschaftsinformatik für die Entstehung der Krise diskutiert und anschließend die zukünftigen Herausforderungen für die Wirtschaftsinformatik abgeleitet werden.

2.2 Schwerpunkte und Aufgaben der Wirtschaftsinformatik

Im Jahr 1994 veröffentlichte die Wissenschaftliche Kommission Wirtschaftsinformatik (WKWI) im Verband der Hochschullehrer für Betriebswirtschaft e.V. das Profil der Wirtschaftsinformatik (WKWI 1994). Dieses Profil formuliert „*Informations- und Kommunikationssysteme (IKS) in Wirtschaft und Verwaltung*“ als zentralen Gegenstand der Wirtschaftsinformatik (WKWI 1994, S. 80). Unter IKS werden dabei „*soziotechnische Systeme, die menschliche und maschinelle Komponenten (Teilsysteme) als Aufgabenträger umfassen, die voneinander abhängig sind, ineinander greifen und / oder zusammenwirken*“ verstanden (WKWI 1994, S. 80). Im deutschsprachigen Raum hat sich dieses Verständnis der Wirtschaftsinformatik dabei weitestgehend durchgesetzt (Fink et al. 2005). So benennen beispielsweise Mertens et al. (1998, S. 1) die Aufgaben der Wirtschaftsinformatik als die „*Konzeption, Entwicklung, Einführung, Wartung und Nutzung von Systemen der computergestützten Informationsverarbeitung im Betrieb*“. Auch Hansen und Neumann (2001) sowie Scheer (1997) verwenden eine vergleichbare Definition.

Im Mittelpunkt der Wirtschaftsinformatik steht gemäß der WKWI (1994, S. 80) „*die Unterstützung bei der Erfüllung betrieblicher Aufgaben*“ mittels IKS. Dabei ist der primäre Zweck dieser Systeme, einerseits mit Hilfe von Informationen die betrieblichen Prozesse zu steuern und andererseits die Informationsnachfrage von Aufgabenträgern zu befriedigen. „*Art und Umfang der Informationsnachfrage ergeben sich aus den in Wirtschaft und Verwaltung zu erfüllenden Aufgaben*“ (WKWI 1994, S. 80). Dementsprechend ist das Ziel der Konstruktion und Anwendung von IKS „*die optimale Bereitstellung von Information und die Unterstützung von Kommunikation nach wirtschaftlichen Kriterien*“ (WKWI 1994, S. 80).

Insbesondere in diesem Kontext ist das Informationsmanagement (IM) von zentraler Bedeutung, welches Voß und Gutenschwager (2001, S. 1) als „*die wirtschaftliche (effiziente) Planung, Beschaffung, Verarbeitung, Distribution und Allokation von Informationen als Ressource zur Vorbereitung und Unterstützung von Entscheidungen (Entscheidungsprozessen) sowie die Gestaltung der dazu erforderlichen Rahmenbedingungen*“ definieren. Mertens et al. (2005) benennen dabei das IM als Teilgebiet und einen der Schwerpunkte der Wirtschaftsinformatik. Obgleich Voß und Gutenschwager (2001) im Gegensatz dazu das IM in die Betriebswirtschaftslehre einordnen, beschreiben auch sie eine enge Verzahnung und Überschneidung der Aufgaben der Wirtschaftsinformatik und der des IM. So bilden „*die Planung und der Einsatz möglicher Funktionalitäten der IT im Zuge der Entscheidungsfindung*“ einen zentralen Schwerpunkt des IM (Voß und Gutenschwager 2001, S. 87). Krcmar (2005, S. 49) stellt eine noch detailliertere Definition des IM dar und fasst dieses als das „*Management der Informationswirtschaft, der Informationssysteme, der Informations- und Kommunikationstechniken sowie der übergreifenden Führungsaufgaben*“ zusammen. Der zentrale Gegenstand der Informationswirtschaft ist dabei die Ressource Information als Grundlage für Entscheidungen und damit auch als wesentlicher Produktionsfaktor im betrieblichen Leistungserstellungsprozess. Als übergeordnetes Ziel der Informationswirtschaft wird die „*Herstellung des informationswirtschaftlichen Gleichgewichts im Unternehmen*“ (Krcmar 2005, S. 51), also die effiziente Informationsversorgung durch Kongruenz von Informationsangebot, -nachfrage und -bedarf unter Berücksichtigung von Wirtschaftlichkeitsaspekten, formuliert. Insbesondere bei der Entwicklung des Informationsangebots im Rahmen einer unternehmerischen Informationswirtschaft liegt der Schwerpunkt auf dem Einsatz von IKS.

Laut Mertens et al. (2005, S. 4) liegt ein langfristiges Ziel der Wirtschaftsinformatik in der „*sinnhaften Vollautomation*“ eines Betriebs. Gemäß diesem sind alle Aufgaben, die Maschinen nach betriebswirtschaftlichen Maßstäben besser erledigen können, von Menschen auf Anwendungssysteme zu übertragen. Unter Berücksichtigung dieses Ziels und angesichts der rasant zunehmenden Menge verfügbarer Informationen erscheint der stark wachsende Einsatz von IT zur Selektion, Verarbeitung und Bereitstellung von Informationen nach wirtschaftlichen Kriterien nicht nur logisch, sondern sogar zwingend notwendig.

Grundsätzlich lassen sich Anwendungssysteme als Teil von IKS in Administrations- und Dispositionssysteme sowie in Planungs- und Kontrollsysteme unterteilen (Mertens et al. 2005). Während die ersten beiden auch unter dem Begriff operative Systeme zusammengefasst werden, haben Planungs- und Kontrollsysteme „*die Aufgabe, Entschei-*

dungsträgern Informationen und Entscheidungsvorschläge zu präsentieren, die ihnen bei der Unternehmensplanung und -kontrolle helfen“ (Mertens und Meier 2009, S. 1). Zu diesen Systemen zählen die Entscheidungsunterstützungssysteme (EUS), welche neben der reinen Bereitstellung von Informationen auch automatisch Entscheidungshilfen und Handlungsempfehlungen generieren. Mit dem Einsatz solcher Systeme im Unternehmen soll eine effektive Unterstützung im Planungs- und Entscheidungsprozess durch eine Verbesserung des Urteilsvermögens des Managers und damit auch der Entscheidungsqualität erreicht werden (Gluchowski et al. 2008). Zentraler Bestandteil der EUS sind die integrierten, formalen und meist betriebswirtschaftlichen (Entscheidungs-) Modelle, welche sich meist auf bestimmte Probleme oder auf bestimmte Klassen von Aufgaben beziehen (Gluchowski et al. 2008, Mertens und Meier 2009). So können beispielsweise logische Modelle, welche What-if- und How-to-achieve-Fragestellungen beantworten oder bestimmte Optimierungsprobleme lösen, in EUS Verwendung finden.

Auf diesem Gebiet kann die Wirtschaftsinformatik bereits wertvolle Beiträge in Forschung und Praxis verzeichnen. So verwenden bspw. Unternehmen solche Anwendungssysteme vielfach für eine teilweise bis vollständige Automatisierung von operativen Routineentscheidungen und setzen auch zunehmend auf strategischer Ebene immer komplexere und umfassendere EUS ein. Beispiele hierfür finden sich u. a. im Risikomanagement, in der Logistik oder im Customer Relationship Management (CRM). Mit einem sinnvollen Einsatz von EUS können betriebliche Entscheidungsprozesse hinsichtlich Geschwindigkeit und Qualität erheblich verbessert werden, was zu zuverlässigeren Reaktionen auf Markt- und Umweltereignisse führt und Wettbewerbsvorteile schafft. Dies gilt insbesondere auch für den Finanzdienstleistungssektor. So zeigt eine Studie des Beratungsunternehmens McKinsey & Company (Ackermann et al. 2007), dass sich sehr erfolgreiche Banken insbesondere durch einen nahe am Geschäftsbetrieb ausgerichteten Einsatz von IT und durch eine sinnvollen Steuerung ihrer IKS von anderen, weniger erfolgreichen Banken abheben.

2.3 Die Verantwortung und Versäumnisse der Wirtschaftsinformatik im Rahmen der Finanzmarktkrise

Gerade beim Einsatz von EUS für die Unterstützung strategischer Entscheidungen birgt eine (Teil-) Automatisierung jedoch auch Gefahren, da in diesen Situationen im Wesentlichen die Urteilskraft und die Erfahrungen der Führungskraft gefordert sind (Mertens und Meier 2009). Das Potenzial für Probleme liegt überwiegend in den Entscheidungsmodellen und deren Integration in die EUS. Prinzipiell können Entscheidungsmodelle

unabhängig davon, wie innovativ und Mehrwert stiftend diese sind, nur unter vereinfachenden Annahmen einen Teil der realen Komplexität abbilden. Ist – wie oftmals der Fall – mindestens eine Annahme nicht erfüllt, so ist auch die Interpretation der Modellaussagen für eine reale Gegebenheit nicht mehr uneingeschränkt zulässig. Damit ergibt sich ein Modellrisiko, dass also Modelle die relevanten Aspekte der Realität nicht nur unvollständig, sondern fehlerhaft abbilden. Die Gefahr besteht nun in einer möglichen, unreflektierten Modellgläubigkeit der Entscheidungsträger in die EUS und in die integrierten Entscheidungsmodelle. Aufgrund der meist stark theoretischen Prägung sowie der teilweise hohen Komplexität der Modelle haben Führungskräfte vermutlich oftmals nur ein eingeschränktes oder sogar kein Verständnis über deren Funktionsweise, Aussagekraft und Limitationen. Zudem werden EUS überwiegend dafür eingesetzt, aus einer großen Menge die für eine Entscheidung relevanten Informationen zu selektieren, mittels der Modelle zu aggregieren und diese der Führungskraft beispielsweise in Form von wenigen und einfachen Handlungsempfehlungen bereitzustellen. Durch diese Form der Unterstützung wird vermutlich der Anreiz für die Entscheidungsträger, die komplexen Modelle nicht kritisch zu hinterfragen, sondern diesen unreflektiert zu vertrauen, zusätzlich verstärkt. Eine Koinzidenz der immanenten Einschränkungen der Modelle mit einer solchen Modellgläubigkeit birgt nun die Gefahr unternehmerischer Fehlentscheidungen. Daher sind vor allem für modelltheoretische Grenzbereiche eine kritische Prüfung und differenzierte Bewertung der Aussagekraft, Limitationen und Anwendbarkeit erforderlich. Zudem gilt es, Modelle kritisch auf Ihre Robustheit zu prüfen, also inwieweit sie auch bei einer Schwankung der angenommenen Parameter noch näherungsweise korrekte Aussagen liefern.

Insbesondere die komplexen, für die eingangs erwähnte Krise ursächlichen Finanzstrukturen basieren fast ausschließlich auf finanzmathematischen (Entscheidungs-) Modellen, deren Berechnungen wiederum auf zahlreichen Annahmen und der Extrapolation historischer Daten basieren (Bartmann et al. 2009). So werden bspw. die Ratings von Finanzdienstleistern und von strukturierten Wertpapieren wie Asset-Backed Securities (ABS), Collateralized Debt Obligations (squared) (CDOs(2)) und Asset-Backed Commercial Papers (ABCPs) unter Verwendung solcher Modelle berechnet. Aber auch die Anwendungssysteme für das Risiko- und Portfoliomanagement vieler Marktteilnehmer stützen sich auf entsprechende Modelle u. a. zur Bewertung der Ausfallrisiken solcher Finanztitel. Obwohl insbesondere bei mehrfach strukturierten Wertpapieren meist nicht mehr nachvollzogen werden kann, welche Kredite mit welchen Risiken den Anleihen letztendlich unterliegen, glaubte ein Großteil der beteiligten Banken und Investoren offensichtlich, mit Hilfe dieser Modelle die tatsächlichen Ausfall-

risiken der komplexen Finanztitel berechnen, bewerten und steuern zu können. Viele der beteiligten Investmentbanken sowie internationale und regionale, private und staatliche Banken investierten teilweise in beträchtlichem Umfang in die Wertpapiere (Rudolph 2008). Der Verlauf der Krise zeigt jedoch, dass diese Einschätzung fehlerhaft war.

Insbesondere mehrfach strukturierte Wertpapiere wie CDOs(2) weisen spezielle Risikostrukturen und Abhängigkeiten auf, die von den eingesetzten Modellen mit ihren vereinfachenden Annahmen nicht vollständig erfasst werden. Unter ungünstigen Umständen können diese die tatsächlichen Risiken der letztendlich unterliegenden Kreditforderungen sogar potenzieren (Bartmann et al. 2009). Auch Rudolph (2008) stellt neben zahlreichen anderen Ursachen für die Krise fest, dass die Modelle der Ratingagenturen die tatsächlichen Risiken nur unzureichend abgebildet haben. So werden die Ratings von strukturierten Finanzprodukten zwar als nützlich eingestuft, jedoch können diese „*wegen inhärenter Unzulänglichkeiten die Risiken dieser Produkte nicht voll erfassen*“ (Rudolph 2008, S. 720). Gleichzeitig wird darauf hingewiesen, dass gerade aufgrund der Komplexität dieser Transaktionen die Investoren dazu neigen, sich bei diesen Anlagen stärker als bei anderen Wertpapieren auf die Ratingurteile der Agenturen zu verlassen. Darüber hinaus benennt Rudolph (2008) auch technische Mängel im Risiko- und Portfoliomanagement der Investoren als eine Ursache für die Krise. Zugleich weist er jenseits dieser Defizite auf eine mangelhafte Risikokultur hin, die zu Fehlern im Risikomanagement geführt hat. So haben sich viele Banken und Investoren zu stark auf das quantitative Risikomanagement konzentriert und dabei die qualitativen, weichen Aspekte vernachlässigt. „*Da die rechenbaren quantitativen Elemente […] auf Restriktionen ausgerichtet sind, die mit Hilfe von Vergangenheitsdaten arbeiten, befasst sich das Risikomanagement zu wenig mit Szenarien, die bislang unbekannte ökonomische Entwicklungen betreffen*“ (Rudolph 2008, S. 728). So wurde bspw. vielfach angenommen, gewisse zukünftige Marktdaten aus historischen Daten schätzen zu können. Für die o. g. neuartigen Produkte existieren jedoch nur wenige historische Daten und keine Erfahrungswerte über Marktkrisen. Solange eine Marktsituation vorherrschte, die den bisherigen Erfahrungen – nämlich steigende Immobilienpreise und sinkende Zinsen – weitestgehend entsprach, konnten zuverlässige Bewertungen generiert werden. Das aus einem Anstieg des Zinsniveaus und einem Fallen der Immobilienwerte tatsächlich resultierende und um ein vielfaches höhere Risiko für ABS, CDOs(2) und ABCPs konnten die Modelle jedoch nicht prognostizieren. Nahezu alle Modelle setzen auch stets liquide Märkte für vermögensbesicherte Wertpapiere voraus. Eine kritische Überprüfung, inwieweit die eingesetzten Modelle in einer Marktsituation, in der niemand mehr den Wertpapieren vertraut, noch zuverlässige Ergebnisse berechnen können, fand offenbar

nicht statt. Insgesamt zeigten die Modelle eine nur sehr geringe Robustheit.

Nach Meinung des Autors stellt das Ausbleiben einer ausreichenden kritischen Prüfung der verwendeten Modelle das hauptsächliche Versäumnis der Wirtschaftsinformatik im Rahmen der Finanzmarktkrise dar. Das mögliche Argument, die ursächlichen Modelle seien rein finanzmathematischer Natur und damit nicht originär der Wirtschaftsinformatik zuzuordnen, greift hier zu kurz. Denn es sind die von Wirtschaftsinformatikern konzipierten EUS, die auf Basis dieser Modelle Informationen selektieren und Handlungsempfehlungen generieren. Diese Anwendungssysteme bilden als Teil von IKS den primären Gegenstand der Wirtschaftsinformatik. Daher und insbesondere aufgrund der interdisziplinären Ausrichtung zwischen Betriebswirtschaftslehre und Informatik kommt der Wirtschaftsinformatik zumindest eine Mitverantwortung für die fachliche Anwendungslogik der EUS zu. Auch die WKWI (1994, S. 81) verlangt aufgrund des sozio-technischen Erkenntnisgegenstands der Wirtschaftsinformatik, „*dass bei der Auswahl der anzuwendenden Methoden / Werkzeuge nicht nur Fragen der technischen Effizienz, sondern auch die ökonomische und soziale Einsetzbarkeit […] beachtet wird*“. Ein Abweisen der Verantwortung für Aufgaben mit Überschneidungen zu anderen Disziplinen bzw. Aufgabenbereichen sowie für die eigene Beteiligung an der Gestaltung von Marktstrukturen würde diesem Anspruch der Wirtschaftsinformatik klar widersprechen. Mit vergleichbaren Argumenten versuchten sich viele der beteiligten Marktteilnehmer auf ihre originären Tätigkeitsbereiche zu berufen und aus der Mitverantwortung für die Finanzmarktkrise freizusprechen. Gesteht man diesen eine solche Verantwortungsdelegation jedoch nicht zu, so muss konsequenterweise auch der oben formulierte Anspruch an die Wirtschaftsinformatik betont werden. Dieser Anspruch trifft nach Meinung des Autors dabei in gleichem Maße für Praktiker und Wissenschaftler der Wirtschaftsinformatik sowie für Informatiker und Betriebswirte mit Schwerpunkt auf eben dieser Schnittstelle zu.

Ein weiteres Problem liegt in der Verbreitung der verwendeten Modelle. Nutzt ein Großteil der Marktteilnehmer dieselben Modelle, kann dies implizit zu homogenen Erwartungen und gleichgerichtetem Handeln führen. Die Folge ist eine Verstärkung normaler Marktzyklen, die unter anderen Umständen gedämpft würden. Tatsächlich stützten sich nahezu alle involvierten Akteure auf dieselben oder zumindest auf sehr ähnliche Modelle bspw. zur Bewertung und Steuerung der Verbriefungstransaktionen und der strukturierten Wertpapiere. Dieser Umstand führte bei einem Großteil der Marktteilnehmer zu sehr ähnlichen Erwartungen an die Performance der vermögensbesicherten Wertpapiere und damit zu einer stark steigenden Nachfrage dieser Finanztitel. Da jedoch nahezu alle Beteiligten den gleichen Modellen vertrauten, übersahen viele Banken und Investoren

die tatsächlichen Risiken, die die Wertpapiere bei ungünstigen Marktentwicklungen in sich bergen. So konnte die Blase um vermögensbesicherte Wertpapiere entstehen und letztendlich auch platzen.

Auch in diesem Punkt muss sich die Wirtschaftsinformatik nach Meinung des Autors ein Versäumnis eingestehen. Denn der Sachverhalt, dass Entscheidungsmodelle und damit auch EUS bei breiter Verwendung die Grundlage für das Entstehen und letztendlich auch für das Platzen von Marktblasen sein können, ist bei der Konzeption, Entwicklung und Nutzung der EUS offenbar nicht oder nur ungenügend beachtet worden.

2.4 Zukünftige Herausforderungen der Wirtschaftsinformatik

Aus ihrem Selbstverständnis und der soeben skizzierten Mitverantwortung für die Entstehung der Finanzmarktkrise lassen sich einige zentrale, zukünftige Herausforderungen der Wirtschaftsinformatik ableiten. Insgesamt kommt der Wirtschaftsinformatik insbesondere als Interdisziplin zwischen Betriebswirtschaftslehre und Informatik bei der Erforschung und Gestaltung von EUS sowie bei der Integration teilweise fachfremder Modelle eine besondere Verantwortung zu. Sie muss noch stärker als bisher eine Schnittstellenfunktion zwischen den angrenzenden Disziplinen wahrnehmen.

Für die Wirtschaftsinformatik als Formalwissenschaft (WKWI 1994) gilt es daher, insbesondere Entscheidungsmodelle vor einer Integration in EUS stets einer kritischen Prüfung bzgl. Aussagekraft, Limitationen, Anwendbarkeit etc. zu unterziehen. Die Annahmen müssen sorgfältig und explizit herausgearbeitet werden und auf deren Basis entsprechende Limitationen der Modelle sowie deren Robustheit im Rahmen einer kritischen Würdigung abgeleitet werden. Da den Wirtschaftsinformatikern dafür im Einzelfall die Detailkenntnisse fehlen, ist in diesem Kontext eine noch engere und intensivere Zusammenarbeit mit den „Ursprungsdisziplinen“ erforderlich.

In diesem Rahmen muss die Wirtschaftsinformatik darüber hinaus insbesondere mit ihrem Verständnis als Ingenieurwissenschaft (WKWI 1994) verstärkt Theorien, Konzepte und Methoden entwickeln, die eine derartige Gestaltung von EUS ermöglichen, dass die Versuchung blinder Modellgläubigkeit weitestgehend reduziert wird. Denn ein blindes Vertrauen von Entscheidungsträgern in EUS und die darin integrierten Modelle birgt die Gefahr schwerwiegender Fehlentscheidungen. So können innovative Modelle zur Entscheidungsunterstützung und entsprechende Anwendungssysteme für ein erfolgreiches unternehmerisches Handeln zwar notwendig, aber niemals hinreichend sein. Die Verantwortung muss stets bei den Entscheidungsträgern liegen und dies erfordert EUS, die einen differenzierten und reflektierten Umgang mit den generierten Handlungs-

empfehlungen fördern. Da sich Entscheidungsträger offensichtlich bei zunehmend komplexeren Modellen vermehrt auf die Modellergebnisse verlassen und diese nicht kritisch hinterfragen, darf bei der Gestaltung von EUS und der Integration von Entscheidungsmodellen nicht nur die Komplexitätsbeherrschung im Vordergrund stehen, sondern auch eine sinnvolle und angemessene Komplexitätsreduktion. Auch wenn dies in manchen Fällen nicht mit dem Ziel einer Vollautomation vereinbar sein mag, muss die situative Identifikation und Definition eines sinnvollen Automatisierungsgrades verstärkt in den Fokus der Wirtschaftsinformatik gerückt werden. In diesem Kontext kann es auch als Aufgabe der Wirtschaftsinformatik gesehen werden, insbesondere für Entscheidungsträger mit einem nicht ausreichenden Expertenverständnis für die komplexen Sachverhalte und Modelle der EUS – zu denen im Rahmen der Finanzmarktkrise bspw. einige Führungskräfte und Aufsichtsräte der deutschen Landesbanken sowie einige zuständige Politiker zählen dürften – zusätzliche Risikomanagement- und Frühwarnsysteme zu entwickeln, die auf einem deutlich niedrigeren und verständlicheren Komplexitätsniveau die fundamentalen und existenziellen Risiken eines Unternehmens überwachen. Solche Systeme könnten bspw. jeweils das Verhältnis von Eigenkapital zu dem Investitionsvolumen einer Anlageform berechnen und bei Überschreiten einer kritischen Größe den Führungskräften entsprechend eindeutige Warnungen ausgeben.

Die kritische Prüfung von Entscheidungsmodellen und die Gestaltung von EUS müssen zudem unter Berücksichtigung eines Szenarios erfolgen, in dem Akteure gezielt nach Schwächen und Limitationen der Systeme suchen und diese zu ihrem Vorteil ausnutzen. Ob und inwieweit im Rahmen der Finanzmarktkrise einige wenige Marktteilnehmer die o. g. Defizite der Modelle und EUS kannten und gezielt zu ihrem Vorteil ausnutzten, bleibt zwar spekulativ, jedoch angesichts der Tatsache, dass einige Akteure von der Krise zeitweise sogar profitierten und dass die Ermittlungen des FBI gegen zahlreiche Banken und Finanzdienstleister bereits zu ersten Festnahmen führten (o.V. 2008d), ein begründeter Verdacht.

Im Sinne der Wirtschaftsinformatik als Realwissenschaft (WKWI 1994) zeichnet sie ebenfalls für die Evaluation von EUS und den darin umgesetzten Modellen unter Realweltbedingungen verantwortlich. Erkenntnisse aus einer kritischen Untersuchung der praktischen Anwendbarkeit von Modellen und deren Robustheit unter Berücksichtigung der Modellannahmen muss an die „Ursprungsdisziplinen“ und die bereits vorhandenen Anwender zurück gespielt werden. Dabei darf nicht nur die isolierte Evaluation einzelner EUS im Fokus stehen. Auch das Zusammenwirken mit anderen IKS sowie die strukturellen Auswirkungen durch den Einsatz der EUS müssen Gegenstand der Betrachtung sein.

Nur mit diesem Anspruch kann die Wirtschaftsinformatik seine vielleicht nicht auf den ersten Blick ersichtliche, aber dennoch immanente Verantwortung wahrnehmen und einen Beitrag zur Vermeidung zukünftiger Krisen leisten. Der interdisziplinären Ausrichtung der Wirtschaftsinformatik wohnt nicht nur eine Stärke inne, sondern auch eine entsprechende Verantwortung.

Literatur (Kapitel II.2)

Ackermann J, Yeung MA, van Bommel E (2007) Better IT management for banks. TheMcKinseyQuarterly, http://www.mckinseyquarterly.com/Better_IT_management_for_banks_2028, Abruf am: 05.01.2009

Bartmann P, Buhl HU, Hertel M (2009) Ursachen und Auswirkungen der Subprimekrise. Informatik-Spektrum 32(2):127-145

Fink A, Schneidereit G, Voß S (2005) Grundlagen der Wirtschaftsinformatik. 2. Auflage, Physica-Verlag, Heidelberg

Gluchowski P, Gabriel R, Dittmar C (2008) Management Support Systeme und Business Intelligence – Computergestützte Informationssysteme für Fach- und Führungskräfte. 2. Auflage, Springer, Berlin

Hansen HR, Neumann G (2001) Wirtschaftsinformatik. 8. Auflage, UTB, Stuttgart

Krcmar H (2005) Informationsmanagement. 4. Auflage, Springer, Berlin

Mertens P, Bodendorf F, König W, Picot A, Schumann M (1998) Grundzüge der Wirtschaftsinformatik. 5. Auflage, Springer, Berlin

Mertens P, Bodendorf F, König W, Picot A, Schumann M, Hess T (2005) Grundzüge der Wirtschaftsinformatik. 9. Auflage, Springer, Berlin

Mertens P, Meier MC (2009) Integrierte Informationsverarbeitung 2 – Planungs- und Kontrollsysteme in der Industrie. 10. Auflage, Gabler, Wiesbaden

o.V. (2008a) Lehman Brothers muss Konkurs beantragen. Handelsblatt, http://www.handelsblatt.com/unternehmen/banken-versicherungen/lehman-brothers-muss-konkurs-beantragen;2040059, Abruf am: 05.01.2009

o.V. (2008b) US-Autoabsatz bricht wegen Finanzkrise ein. Financial Times Deutschland, http://www.ftd.de/unternehmen/industrie/:Kreditklemme-US-Autoabsatzbricht-wegen-Finanzkrise-ein/420919.html, Abruf am: 05.01.2009

o.V. (2008c) Island zahlt nicht mehr. Financial Times Deutschland, http://www.ftd.de/boersen_maerkte/aktien/anleihen_devisen/:Staatsbankrott-Island-zahlt-nicht-mehr/427188.html, Abruf am: 05.01.2009

o.V. (2008d) FBI nimmt Ex-Bear-Stearns-Fondsmanager fest. Handelsblatt, http://www.handelsblatt.com/unternehmen/banken-versicherungen/fbi-nimmtex-bear-stearns-fondsmanager-fest;1445748, Abruf am: 05.01.2009

Scheer A-W (1997) Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse. 7. Auflage, Springer, Berlin

Rudolph B (2008) Lehren aus den Ursachen und dem Verlauf der internationalen Finanzkrise. Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung 60: 713–741

Voß S, Gutenschwager K (2001) Informationsmanagement. Springer, Berlin

WKWI (Wissenschaftliche Kommission Wirtschaftsinformatik im Verband der Hochschullehrer für Betriebswirtschaft e. V.) (1994) Profil der Wirtschaftsinformatik. Wirtschaftsinformatik 36:80–81

III Systemische Risiken in Anwendungssystemen

Das folgende Kapitel befasst sich mit systemischen Risiken im Rahmen des Managements von Anwendungssystemen. Als spezielle Form eines systemischen Risikos steht dabei insbesondere die Ausbreitung von Fehlern aufgrund der nachträglichen Änderung einer Softwarekomponente auf angrenzende Komponenten über deren Abhängigkeiten beziehungsweise in einem allgemeineren Sinne die Ausbreitung von Änderungen im Fokus.

Als eine Ursache für die Fehleranfälligkeit von Wartungsmaßnahmen gilt das Fehlen einer adäquaten technischen Programmdokumentation. Der erste Beitrag widmet sich daher der Erstellung einer zweckmäßigen Programmdokumentation als präventive Maßnahme zur Reduzierung der Auswirkungen der Fehlerausbreitung. Ziel dieses Beitrags ist insbesondere die Deduktion neuer formaler Zusammenhänge zwischen dem Dokumentationsgrad einer Software und der Anzahl der Fehler aufgrund nachträglicher Änderungen sowie die Ableitung neuer Hypothesen bezüglich des Dokumentationsgrads, welcher die barwertigen Auszahlungen für die Erstellung einer technischen Programmdokumentation während der Softwareentwicklung und für die Fehlerbehebung während der Wartung minimiert.

Eine alternative Möglichkeit zur Reduzierung des Ausbreitungseffekts von Änderungen ist die Entwicklung einer modularen Softwarearchitektur mit einer möglichst losen Kopplung der Softwaremodule. Gleichzeitig erscheint es jedoch insbesondere für die nachgelagerte Wartung von Vorteil, wenn alle Funktionalitäten, welche bei der Änderung einer fachlichen Anforderung mit hoher Wahrscheinlichkeit gemeinsam der Modifikation bedürfen, auf möglichst wenige Module verteilt sind. Vor diesem Hintergrund beschäftigt sich der zweite Beitrag mit der Bewertung alternativer modularer Softwarearchitekturen unter der Berücksichtigung von sowohl technischen als auch fachlichen Abhängigkeiten.

1 Beitrag: „*Technische Softwaredokumentation unter ökonomischen Gesichtspunkten – Ein formal-deduktiver Ansatz*“

Autor:	Peter Bartmann
	Kernkompetenzzentrum Finanz- & Informationsmanagement, Lehrstuhl für BWL, Wirtschaftsinformatik, Informations- & Finanzmanagement (Prof. Dr. Hans Ulrich Buhl) Universität Augsburg, D-86135 Augsburg
	peter.bartmann@wiwi.uni-ausburg.de

Erschienen in:	Zeitschrift für Betriebswirtschaft, 82, 3, 2012, S. 243-274

Zusammenfassung:

Der Softwaredokumentation wird offensichtlich ein nur sehr geringer Stellenwert eingeräumt. Dabei bestätigen empirische Studien, dass mit der Verfügbarkeit einer adäquaten technischen Programmdokumentation eine Reduktion der Fehleranfälligkeit von nachträglichen Änderungen im Rahmen der Softwarewartung einhergeht. Für jedes Softwareprojekt ergibt sich demnach die Frage, in welcher Höhe eine Investition in die Erstellung einer technischen Programmdokumentation ökonomisch sinnvoll ist. Dieses Entscheidungsproblem wurde wissenschaftlich bisher stark vernachlässigt. Da entsprechende Zusammenhänge empirisch vermutlich nur schwer aufgedeckt werden können, stellt der Beitrag einen neuartigen deduktiven Ansatz vor, welcher die barwertigen Auszahlungen für die Erstellung einer technischen Programmdokumentation und für die Fehlerbehebung gegenüberstellt. Grundlage des Modells bildet das Phänomen der Fehlerausbreitung von einer Komponente über deren Abhängigkeiten zu angrenzenden Komponenten. Auf dieser Basis kann folgende Hypothese abgeleitet werden: Auch bei einer beliebig niedrigen Dokumentationsqualität oder einem beliebig hohen Kalkulationszinssatz ist es stets vorteilhaft, eher zu viel als vergleichsweise zu wenig zu dokumentieren. Diese Hypothese erweist sich dabei auch gegenüber der Änderung einiger Annahmen als robust.

1.1 Einleitung

Zahlreiche Unternehmen setzen IT zur Unterstützung ihrer Geschäftsprozesse ein. Da ein dynamisches Marktumfeld und sich ändernde Kundenanforderungen kontinuierliche Anpassungen der Geschäftsprozesse verlangen, ändern sich im Laufe des Betriebs ebenfalls die Anforderungen an die eingesetzte Software. Folglich müssen Softwaresysteme auch nach Inbetriebnahme im Rahmen der Softwarewartung laufend an neue Anforderungen angepasst werden. Dabei nimmt die Wartung von Softwaresystemen laut Zarnekow et al. (2004, S. 181) bereits bei einer Betriebsdauer von fünf Jahren knapp 80% der gesamten Lebenszykluskosten ein. Auch Krishnan et al. (2004, S. 396) beziffern die anteiligen Kosten für die Softwarewartung von in Betrieb befindlichen Altsystemen (*legacy systems*) auf 50% - 80% der gesamten Lebenszykluskosten. Die Softwarewartung stellt also einen erheblichen Kostenfaktor für Unternehmen und für die öffentliche Verwaltung dar.

Nachträgliche Modifikationen eines Softwaresystems im Rahmen der Wartung können fehlerhaft und inkonsistent zur ursprünglichen Struktur der Software sein (Krishnan et al. 2004, S. 396). Die Änderung einer Softwarekomponente kann dazu führen, dass angrenzende Komponenten nicht mehr fehlerfrei mit dieser interagieren. In der Folge sind zusätzliche Modifikationen in den angrenzenden Komponenten erforderlich, welche wiederum neue Änderungen bedingen und so weiter (Beszédes et al. 2007, S. 296). Inkonsistente Modifikationen sowie die Ausbreitung der dadurch entstehenden Fehler können demnach einen erheblichen Einfluss auf die Kosten für die Softwarewartung ausüben.

Ein wesentlicher Einflussfaktor auf Fehler und Inkonsistenzen durch nachträgliche Modifikationen ist das Fehlen einer adäquaten technischen Programmdokumentation (Dzidek et al. 2008, S. 424). Nachträgliche Änderungen von Softwaresystemen werden oftmals einige Monate oder sogar Jahre nach deren Entwicklung und in vielen Fällen von anderen Mitarbeitern als den eigentlichen Entwicklern umgesetzt. Daher nimmt die Analyse und das Verstehen der anzupassenden Software meist einen beträchtlichen Teil des Wartungsaufwands ein (Tan und Mookerjee 2005, S. 238). Für die Verständlichkeit von Softwaresystemen spielt nun die (technische) Programmdokumentation zum Beispiel in Form von externen Dokumenten, Quellcode-Kommentaren oder einer begreiflichen Bezeichnung von Methoden und Variablen eine unentbehrliche Rolle, da diese Informationen über den Zweck und die Ziele des vorliegenden Codes sowie über die dahinterliegenden Konzepte bereitstellen (Rajlich 2009, S. 4). So kann insbesondere

das Fehlen einer adäquaten technischen Programmdokumentation zu einem schlechteren Verständnis der Software durch das Wartungspersonal führen und damit die Ursache für inkonsistente und fehlerhafte Modifikationen bilden (Dzidek et al. 2008, S. 424). Zusätzliche Investitionen in eine zweckmäßige technische Programmdokumentation während der Softwareentwicklung können dagegen zu einer besseren Verständlichkeit der Software beitragen und die Wahrscheinlichkeit für inkonsistente Modifikationen während der Wartung reduzieren.

Demgegenüber wird der Softwaredokumentation im Allgemeinen jedoch offenbar nur ein sehr geringer Stellenwert beigemessen. Softwareprojekte sind oftmals durch einen hohen Zeit- und Kostendruck gekennzeichnet und mit dem Versuch, entsprechende Projektziele einzuhalten oder deren Überschreitung zu reduzieren, werden während der Entwicklung häufig Abstriche bei der Dokumentation vorgenommen (Van Vliet 2008, S. 14). Eine Schätzung aus dem Jahr 2008 beziffert den Anteil schlecht dokumentierter Software weltweit auf ca. 80% (Van Vliet 2008, S. 466). Darüber hinaus haben sich in den letzten Jahren die Ansätze der agilen Softwareentwicklung wie *Scrum* in der Praxis etabliert, welche eine lauffähige Software höher gewichten als eine umfangreiche Dokumentation (Hruschka 2003, S. 398). Sie folgen unter anderem der Maxime, dass zum Beispiel für die nachgelagerte Wartung, Anpassung und Weiterentwicklung der Software nur so viel wie nötig und so wenig wie möglich in die Dokumentation investiert wird. Mit derartigen Maximen versuchen die agilen Ansätze den bürokratischen Aufwand weitestgehend zu reduzieren und gelten aufgrund dessen als leichtgewichtig und flexibel. Dennoch scheint auch im Rahmen solcher Ansätze ein erheblicher Bedarf an einer technischen Programmdokumentation zu bestehen. Laut einer Umfrage gehören bei agilen Vorgehensmodellen unter anderem Quellcode-Kommentare zu den wichtigsten Informationsquellen für das Verständnis von zu wartender Software (de Souza et al. 2005, S. 73-74). Außerdem existieren Hinweise, dass das Fehlen einer adäquaten technischen Programmdokumentation ebenfalls im Rahmen agiler Entwicklungsmethoden das Verstehen einer Software erheblich erschweren und damit auch die Wahrscheinlichkeit für fehlerhafte Modifikationen erhöhen kann (Hanssen et al. 2009, S. 488).

Hinsichtlich der Dokumentation von Softwaresystemen besteht aus ökonomischer Sicht also ein Trade-off. Einerseits bedeutet die Erstellung einer umfassenderen technischen Programmdokumentation bei der Softwareentwicklung in der Regel höhere Auszahlungen während der Entwicklungsphase. Andererseits bestätigen jedoch mehrere wissenschaftliche Studien, dass die Verfügbarkeit einer adäquaten technischen Programmdokumentation insbesondere bei objektorientierten Softwaresystemen zu einer

geringeren Fehleranfälligkeit von nachträglichen Modifikationen führt (Arisholm et al. 2006, S. 365, Dzidek et al. 2008, S. 424, Prechelt et al. 2002, S. 595). Folglich kann über die Bereitstellung einer zusätzlichen technischen Programmdokumentation auch eine Reduktion der Auszahlungen für die Behebung von Fehlern und Inkonsistenzen im Rahmen der Softwarewartung erreicht werden. Für jedes Softwareprojekt ergibt sich demnach die Frage, in welcher Höhe eine Investition in die Erstellung einer zweckmäßigen technischen Programmdokumentation ökonomisch sinnvoll ist. Dieses Entscheidungsproblem wurde wissenschaftlich bisher nur ungenügend behandelt. Darüber hinaus identifizieren empirische Studien neben der technischen Programmdokumentation mit der Anzahl der Abhängigkeiten einer Softwarekomponente zu anderen Komponenten einen weiteren Einflussfaktor auf die Fehleranfälligkeit von nachträglichen Änderungen (Briand et al. 2002, S. 714, Gyimóthy et al. 2005, S. 909, Singh et al. 2010, S. 31). Folglich können entsprechende Zusammenhänge, welche die unterschiedlichen Einflussfaktoren berücksichtigen und als Grundlage für eine befriedigende Beantwortung dieser Problemstellung dienen können, empirisch vermutlich nur schwer aufgedeckt werden. Vor diesem Hintergrund stellt der vorliegende Beitrag mittels eines formal-deduktiven Ansatzes die barwertigen Auszahlungen für die Erstellung einer technischen Programmdokumentation während der Entwicklung den barwertigen Auszahlungen für die Fehlerbehebung während der Softwarewartung gegenüber. Dabei liegt dem Modell unter anderem das Phänomen zugrunde, dass sich Fehler aufgrund der nachträglichen Modifikation einer Komponente auch auf angrenzende Komponenten über deren Abhängigkeiten ausbreiten können (Beszédes et al. 2007, S. 296). Ziel ist die Generierung neuer, empirisch prüfbarer Hypothesen im Sinne einer wissenschaftlichen Erklärung auf Basis der existierenden, überwiegend beschreibenden und empirisch geprägten Literatur.

Dazu wird in den folgenden beiden Abschnitten der Forschungsgegenstand abgegrenzt und die angewandte Forschungsmethodik begründet, bevor der vierte Abschnitt einen Überblick der relevanten Literatur gibt. Anschließend werden im fünften Abschnitt die grundlegenden Annahmen getroffen und das Modell aus diesen abgeleitet. Die Analyse des Modells erfolgt im sechsten Abschnitt. Nach einer kritischen Würdigung folgt eine Diskussion des theoretischen und praktischen Beitrags. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick.

1.2 Abgrenzung des Forschungsgegenstands

Im Fokus des vorliegenden Beitrags stehen Anwendungssoftwaresysteme, für deren

Implementierung höhere, insbesondere objektorientierte Programmiersprachen Verwendung finden. Derartige Softwaresysteme setzen sich in der Regel aus verschiedenen Komponenten wie beispielsweise Funktionen, Methoden, Klassen oder Modulen zusammen. Diese Komponenten stehen dabei über verschiedenartige Abhängigkeiten wie zum Beispiel Methodenaufrufe, Datenaustausch oder Cross-Cutting Concerns in Beziehung zueinander. Eben solche Abhängigkeiten spielen für das Verstehen und die Anpassung von Software im Rahmen der Wartung eine zentrale Rolle.

Grundsätzlich ist die Analyse und das Verstehen des existierenden Codes durch das Wartungspersonal vorausgehender Bestandteil aller nachträglichen Änderungen im Rahmen der Softwarewartung (Robillard et al. 2004, S. 889, Rostkowycz et al. 2004, S. 94). Bevor eine Änderung umgesetzt werden kann, müssen die zuständigen Entwickler die zu modifizierenden Stellen des Softwaresystems identifizieren und analysieren. Dazu muss eine abstrakt beschriebene Änderungsanforderung mit den relevanten Softwarekomponenten in Verbindung gebracht werden (*concept location*) (Rajlich 2009, S. 3-8, Rostkowycz et al. 2004, S. 94-95). Als einen der ersten Schritte sucht das Wartungspersonal beispielsweise über Stichwörter potenziell betroffene Komponenten des Softwaresystems, um diese bezüglich ihrer Funktionalität und ihrer Interaktionen mit angrenzenden Komponenten zu untersuchen und zu verstehen (*concept recognition* (Rajlich 2009, S. 3)). Anschließend navigieren die Entwickler zu einigen der angrenzenden Komponenten, welche wiederum bezüglich Funktionalität und Interaktionen analysiert sowie auf deren Relevanz für die umzusetzende Modifikation geprüft werden, bevor sie als Ausgangspunkt für die weitere Analyse dienen können (Ko et al. 2006, S. 982-983). Insbesondere im Rahmen dieser Analyse von Komponenten und deren Interaktionen sind zusätzliche Informationen über die im Code implementierten Funktionalitäten und Konzepte für die Verständlichkeit unentbehrlich (Rajlich 2009, S. 4). Diese können beispielsweise als zusätzliche Dokumente, Kommentare im Quellcode oder als eindeutige und begreifliche Namen von Methoden, Variablen etc. vorliegen. All diese Formen zusätzlicher Informationen werden fortan als *Dokumentation* bezeichnet. Dabei bezieht sich diese Definition ausschließlich auf die technische Programmdokumentation einer Software, welche für die Wartungsaktivitäten benötigt wird. Andere Formen der Softwaredokumentation wie beispielsweise die Benutzerdokumentation werden nicht betrachtet.

Besitzt das Wartungspersonal ein nicht ausreichendes Verständnis über die Funktionalität der Softwarekomponenten und über deren Abhängigkeiten, können Komponenten derart modifiziert werden, dass einige der angrenzenden Komponenten nicht mehr

fehlerfrei mit der geänderten Komponente interagieren würden. Dies kann zusätzliche Änderungen in den angrenzenden Komponenten erfordern, welche wiederum neue Änderungen verlangen und so weiter (Beszédes et al. 2007, S. 296). Eine Situation, in der eine Komponente aufgrund der inkonsistenten Modifikation einer benachbarten Komponente mit dieser nicht mehr fehlerfrei interagieren würde und einer entsprechenden Änderung bedarf, wird im Folgenden als *Fehler* in der betroffenen Komponente bezeichnet. Diese Definition bezieht sich dabei ausschließlich auf logische oder semantische Fehler im Quellcode (*fault*, *bug*) (Radatz 1990), welche bei der Ausführung des Softwareprogramms die Ursache für Versagensfälle (*failure*) wie beispielsweise Laufzeitfehler oder ein nicht spezifikationskonformes Verhalten bilden können. Das Auftreten von Versagensfällen an sich sowie andere Fehlerarten wie zum Beispiel Fehler in der Bedienung werden nicht betrachtet. Nun können insbesondere schwer ersichtliche und ungenügend dokumentierte Abhängigkeiten wie beispielsweise Cross-Cutting Concerns die Verständlichkeit einer Software beträchtlich einschränken und die Ursache für solche Fehler bilden (Eaddy et al. 2008, S. 498). Mit der Verfügbarkeit einer adäquaten technischen Programmdokumentation, welche Informationen über die Funktionalitäten und Interaktionen von Komponenten bereitstellt, kann sich die Wahrscheinlichkeit für derartige Fehler erheblich reduzieren (Dzidek et al. 2008, S. 417). Allerdings bedeutet das Bereitstellen einer zusätzlichen technischen Programmdokumentation einen höheren Aufwand für deren Erstellung. Dieser Trade-off bildet den Forschungsgegenstand des vorliegenden Beitrags.

Als Grundlage für das vorgestellte Modell wird der Lebenszyklus einer Software grob in die Phase der initialen Entwicklung und in die nachgelagerte Wartungsphase unterteilt. Vereinfachend wird im Folgenden die Erstellung der technischen Programmdokumentation auf die Entwicklungsphase beschränkt. Bei einer Dokumentation während der Entwicklung wird unterstellt, dass nur ein vernachlässigbar geringer Aufwand für das Verstehen der Software erforderlich ist. Dagegen bedeutet eine nachträgliche Dokumentation im Rahmen der Softwarewartung einen zumindest einmal notwendigen und mit nachträglichen Modifikationen vergleichbaren Aufwand für die Analyse und das Verstehen der existierenden Software (Rostkowycz et al. 2004, S. 92). Eben dieser Analyseprozess kann jedoch mittels einer zuvor bereitgestellten technischen Programmdokumentation unterstützt werden. Daher wird der Fokus dieses Beitrags auf den Effekt der technischen Programmdokumentation, welche während der Entwicklung erstellt wird und damit bereits zu Beginn der Wartungsphase zur Verfügung steht, auf die Fehleranfälligkeit von Modifikationen gelegt und eine nachträgliche Dokumentation während

der Wartung nicht betrachtet.

Eine scharfe Trennung zwischen Entwicklung und Wartung für ein gesamtes Softwaresystem dürfte nur bei sequenziellen Vorgehensmodellen wie dem Wasserfallmodell gelingen (Rajlich 2006, S. 68). Iterative, inkrementelle und agile Vorgehensmodelle untergliedern die Softwareentwicklung gewöhnlich in kurze, sich wiederholende Zyklen, womit die Entwicklung und die Wartung eines gesamten Softwaresystems zumindest teilweise ineinandergreifen. Daher wird der Fokus dieses Beitrags primär auf die sequenzielle Softwareentwicklung beschränkt, welche eine klare Trennung zwischen Entwicklung und Wartung erlaubt. Dennoch erscheint eine Untersuchung der Zusammenhänge zwischen der technischen Programmdokumentation und der Fehleranfälligkeit von nachgelagerten Modifikationen im Rahmen iterativer Vorgehensmodelle lohnenswert. Nachfolgende Iterationen haben eine Erweiterung der in vorausgehenden Iterationen entwickelten Softwarekomponenten zum Gegenstand, welche auch eine Modifikation von bereits existierenden Komponenten erfordern kann. Dabei kann für nachgelagerte Modifikationen insbesondere bei größeren Softwaresystemen zumindest in einem gewissen Umfang ebenfalls eine entsprechende Analyse des zu modifizierenden Codes notwendig sein (Rajlich 2006, S. 69-70), welche durch die Bereitstellung einer adäquaten technischen Programmdokumentation unterstützt werden kann. Eine Erweiterung des vorgestellten Modells, welche die Spezifika iterativer Vorgehensmodelle adressiert, wird in den Fokus zukünftiger Forschung gerückt.

Da die Erstellung der technischen Programmdokumentation und die Fehlerbehebung zu verschiedenen Zeitpunkten anfallen, werden für die Untersuchung des oben beschriebenen Trade-off die Barwerte der Auszahlungen für die Dokumentation während der Entwicklung und für die Fehlerbehebung während der Softwarewartung gegenübergestellt. Damit bezieht sich der vorliegende Beitrag primär auf kommerzielle Software und vernachlässigt Open Source Software. Dennoch scheint auch für diese ein hoher Bedarf an technischer Programmdokumentation zu bestehen (Sharif und Buckley 2008, S. 6). Das vorgestellte Modell wäre jedoch nur mit entsprechenden Änderungen, beispielsweise dass statt den Auszahlungen die für die Dokumentation und Fehlerbehebung erforderliche Zeit betrachtet wird, auf Open Source Software anwendbar.

Darüber hinaus beschränkt sich der Fokus auf Unternehmen, die sowohl die Entwicklung als auch die Wartung eines Softwaresystems durchführen. An dieser Stelle sei darauf hingewiesen, dass das vorgestellte Modell nur für diesen spezifischen Fall Gültigkeit besitzt und eine Verallgemeinerung der Ergebnisse ohne eine entsprechende

Modellerweiterung unzulässig ist. Dieser Fokus schließt einerseits alle Fälle ein, in denen Unternehmen Individualsoftware für die Eigenverwendung oder für Kunden entwickeln und im Rahmen des Betriebs auch warten. Andererseits betrifft dies auch die Anbieter von Standardsoftware, da diese in der Regel sowohl für die Entwicklung als auch für die Wartung und Weiterentwicklung verantwortlich sind. Nicht betrachtet werden dagegen alle Situationen wie zum Beispiel das Outsourcing der Wartung an spezielle Dienstleister (Sneed 2008, S. 1-4) oder das Application Outsourcing (Riedl und Kepler 2003, S. 8), in denen die Entwicklung und die Wartung einer Software von zwei unterschiedlichen Parteien durchgeführt werden. Der Umstand, dass im Jahr 2010 gegenüber dem weltweiten Investitionsvolumen in Software in Höhe von 395 Mrd. US-Dollar (Statista 2010b) lediglich 27 Mrd. US-Dollar für Application Outsourcing ausgegeben wurden (Statista 2010a), darf an dieser Stelle als Indiz für die im Vergleich vermutlich höhere ökonomische Relevanz der betrachteten Fälle angeführt werden.

1.3 Forschungsmethodik

Folgt man dem Forschungsrahmen für den Bereich des Operations Management von Meredith et al. (1989, S. 301-304), sollen sich alle Forschungsaktivitäten in einen kontinuierlichen, sich wiederholenden Forschungskreislauf aus Beschreibung (*description*), Erklärung (*explanation*) und Prüfung (*testing*) eingliedern. Dabei geht die Beschreibung den erklärenden und prüfenden Forschungsarbeiten voraus und versucht, noch nicht betrachtete Begebenheiten zu untersuchen oder bestimmte Aspekte existierender Forschungsgegenstände im Detail zu beleuchten. Darauf aufbauend sollen Beiträge die zugrunde liegenden Zusammenhänge und Kausalitäten ableiten, welche die beschriebenen Forschungsgegenstände und Begebenheiten erklären können. Neben der reinen Erklärung von Phänomenen kann dabei auch der Anspruch auf die Entwicklung von Gestaltungsempfehlungen im Sinne normativer Ansätze erhoben werden. Die anschließende Prüfung der erklärenden Beiträge erfolgt durch Untersuchungen, inwieweit diese Beiträge entsprechende Phänomene aus der Realität zuverlässig prognostizieren können (*prediction*). Die dabei gewonnen Erkenntnisse bilden die Basis für die schrittweise Verbesserung der erklärenden Beiträge sowie für bessere Handlungsempfehlungen zur Lösung der Probleme in der Praxis.

Wie bereits erwähnt und im folgenden Abschnitt detailliert dargestellt kann auf Basis der existierenden Literatur aus den Bereichen Softwarewartung und *Program Comprehension* die Verfügbarkeit einer adäquaten technischen Programmdokumentation als Einflussfaktor auf die Fehleranfälligkeit von nachgelagerten Änderungen einer Software

angenommen werden. Vor diesem Hintergrund muss für jedes Softwareprojekt aus ökonomischer Sicht entschieden werden, wie viel während der Entwicklung in die Erstellung einer zweckmäßigen technischen Programmdokumentation investiert wird, um die Anzahl der Fehler während der Softwarewartung und damit die Auszahlungen für deren Behebung zu reduzieren. Die im Rahmen der bisherigen Literatur bestätigten Zusammenhänge genügen jedoch nicht für eine befriedigende Beantwortung dieser Problemstellung. Dazu müssen die Auszahlungen für die Erstellung einer technischen Programmdokumentation mit den Auszahlungen für die Fehlerbehebung in Verbindung gebracht werden. Hierbei gilt es, auch andere Einflussfaktoren auf die Anzahl der durch nachträgliche Änderungen eingeführten Fehler zu berücksichtigen. So bestätigen Studien, dass Softwarekomponenten mit einer größeren Anzahl von Abhängigkeiten zu anderen Komponenten auch eine höhere Fehleranfälligkeit im Rahmen der Softwarewartung aufweisen (Briand et al. 2002, S. 714, Gyimóthy et al. 2005, S. 909, Singh et al. 2010, S. 31). Ziel des vorliegenden Beitrags ist somit die Ableitung neuer Zusammenhänge im Sinne einer wissenschaftlichen Erklärung, welche die Grundlage für die Lösung der erläuterten Problemstellung bilden können. Da diese vermutlich nicht unmittelbar aus Beobachtungen in der Realität ableitbar sein dürften, eignet sich hierzu ein deduktiver Ansatz.

Gemäß dem Forschungskreislauf von Meredith et al. (1989, S. 301-304) müssen sich solch deduktiv abgeleitete Zusammenhänge grundsätzlich einer kritischen Prüfung unterziehen, inwieweit sie die erklärten Phänomene prognostizieren können. Dabei können insbesondere formale Modelle beispielsweise empirisch gewonnene Erkenntnisse in eine vom Betrachter unabhängige Form überführen, welche sich für diese nachgelagerte Prüfung der abgebildeten Zusammenhänge eignet (Meredith et al. 1989, S. 308). Ziel ist daher die Entwicklung eines formalen Modells, welches in der existierenden Literatur bisher ungenügend betrachtete aber entscheidungsrelevante Zusammenhänge abbildet. Im Vordergrund steht dabei jedoch weniger die Bereitstellung eines unmittelbar in der Praxis anwendbaren Modells, sondern vielmehr die Deduktion neuer, empirisch prüfbarer Hypothesen. Dennoch erheben die abgeleiteten Zusammenhänge den Anspruch, nach einer entsprechenden Prüfung im Rahmen zukünftiger Forschungsarbeiten und bei einer hinreichend starken Bestätigung als Grundlage für normative Aussagen beziehungsweise Handlungsempfehlungen für die Praxis dienen zu können.

1.4 Überblick der existierenden Literatur

Der Zusammenhang zwischen der Verfügbarkeit einer adäquaten technischen

Programmdokumentation und der Fehleranfälligkeit von nachträglichen Änderungen im Rahmen der Softwarewartung ist bereits durch mehrere Studien empirisch bestätigt. So zeigen Prechelt et al. (2002), dass mit einer expliziten Dokumentation der in einer Software verwendeten Entwurfsmuster (*design patterns*) eine Reduktion der Fehleranfälligkeit von nachträglichen Modifikationen sowie der dafür benötigten Zeit einhergeht. Für eine technische Programmdokumentation in Form von UML-Diagrammen stellen Arisholm et al. (2006) einen analogen Zusammenhang fest. Einen signifikanten Effekt der technischen Programmdokumentation auf die Zeit, welche für die Umsetzung der Änderungen zuzüglich der nachträglichen Anpassung der UML-Diagramme benötigt wird, können sie jedoch nicht bestätigen. Hingegen zeigen Rostkowycz et al. (2004), dass eine nachträgliche, inkrementell während der Softwarewartung erstellte Programmdokumentation zu einer Reduzierung des Wartungsaufwands führt und sich dadurch langfristig auszahlen kann. Dzidek et al. (2008) betrachten ebenfalls eine technische Programmdokumentation in Form von UML-Diagrammen und deren Effekte auf die Softwarewartung. Interessanterweise führt die Verwendung der technischen Programmdokumentation zu einer Verminderung insbesondere der Fehler, welche auf ein Unverständnis existierender Interaktionen von verschiedenen Softwarekomponenten beruhen.

Darüber hinaus bestätigen zahlreiche Studien ebenfalls einen Zusammenhang zwischen der Anzahl der Abhängigkeiten einer Softwarekomponente zu angrenzenden Komponenten und deren Fehleranfälligkeit. So berichten beispielsweise Aggarwal et al. (2009), Singh et al. (2010), Subramanyam und Krishnan (2003) sowie Yu et al. (2002), dass bei Klassen mit einer höheren Anzahl an gekoppelten Klassen auch eine höhere Fehleranfälligkeit beobachtbar ist. Olague et al. (2007) können diesen Zusammenhang bestätigen, weisen jedoch für die Komplexität einer Klasse einen noch stärkeren Zusammenhang zu deren Fehleranfälligkeit nach. Dagegen betonen Briand et al. (2002), Gyimóthy et al. (2005) und Binkley und Schach (1998), dass gemäß deren Ergebnissen die jeweils betrachtete Kopplungsmetrik zu den besten Indikatoren für die Fehleranfälligkeit einer Klasse gehört. Eaddy et al. (2008) betrachten Cross-Cutting Concerns als spezielle Art von Abhängigkeiten und bestätigen auch für diese einen Zusammenhang mit der Fehleranfälligkeit.

Bezüglich anderer Einflussfaktoren auf die Fehleranfälligkeit finden sich in der existierenden Literatur widersprüchliche Aussagen. Während beispielsweise Aggarwal et al. (2009), Subramanyam und Krishnan (2003) und Singh et al. (2010) einen Zusammenhang zwischen der Größe von Klassen und deren Fehleranfälligkeit aufzeigen, können

Eaddy et al. (2008) sowie Fenton und Ohlsson (2000) diesen Zusammenhang nicht bestätigen. Auch bezüglich der Komplexität von Klassen besteht Uneinigkeit. Laut Olague et al. (2007) und Singh et al. (2010) sind Komplexitätsmetriken gute Indikatoren für die Fehleranfälligkeit einer Klasse. Zimmermann und Nagappan (2008) sowie Fenton und Ohlsson (2000) können dagegen keinen signifikanten Effekt der Komplexität einer Klasse auf deren Fehleranfälligkeit zeigen.

Zusammenfassend kann also festgehalten werden, dass die Größe und Komplexität einer Softwarekomponente als Einflussfaktoren auf deren Fehleranfälligkeit empirisch nicht zweifelsfrei bestätigt sind. Gesichert sind dagegen der Zusammenhang zwischen der Fehleranfälligkeit einer Softwarekomponente und der Anzahl ihrer Abhängigkeiten zu angrenzenden Komponenten sowie ein Zusammenhang zwischen der Verfügbarkeit einer adäquaten technischen Programmdokumentation und der Fehleranfälligkeit nachträglicher Änderungen.

Hinweise, warum die Bereitstellung einer zusätzlichen technischen Programmdokumentation zu einer Reduzierung von inkonsistenten Modifikationen im Rahmen der Softwarewartung führt, können Beiträge aus dem Bereich *Program Comprehension* liefern. So beschreibt Rajlich (2009) die sechs fundamentalen Vorgänge des Verstehens von Softwaresystemen. Angelehnt an die entsprechenden Theorien aus der Linguistik und Mathematik charakterisiert er dazu ein Code-Fragment über seine *Extension* (der tatsächliche Code; z.B. *y = x * 0,19*), seine *Intension* (der Zweck des Codes bzw. die darin implementierte, abstrakte Funktionalität; z.B. die Berechnung der auf einen Nettopreis *x* entfallenden Mehrwertsteuer *y*) und über den zugeordneten *Namen* (z.B. „Mehrwertsteuerberechnung“). Die sechs Vorgänge des Verstehens von Software beschreiben dabei die gegenseitige, paarweise Zuordnung dieser drei Merkmale. Während beispielsweise der Vorgang *concept location* die Lokalisation der Code-Fragmente, welche eine gegebene Funktionalität implementieren, bezeichnet (Identifikation der entsprechenden Extension zu einer gegebenen Intension), entspricht der Vorgang *concept recognition* dem Erkennen des Zwecks beziehungsweise der abstrakten Funktionalität eines gegebenen Code-Fragments (Identifikation der zu einer gegebenen Extension gehörigen Intension). Analog dazu definiert Rajlich (2009, S. 3) die Vorgänge *naming* (Zuordnung eines Namens zu einer gegebenen Intension), *definition* (Identifikation der einem gegebenen Namen zugehörigen Intension), *annotation* (Zuordnung eines Namens zu einer gegebenen Extension) und *traceability* (Identifikation der einem gegebenen Namen zugehörigen Extension). Das Verstehen eines Code-Fragments bedeutet, dass das Wartungspersonal alle sechs Vorgänge bei diesem Fragment vollziehen kann.

Auf dieser Basis begründet Rajlich (2009, S. 4) die zentrale Rolle der technischen Programmdokumentation für das Verstehen und die nachträgliche Modifikation von Software. Eine zweckmäßige technische Programmdokumentation beispielsweise in Form von begreiflichen Bezeichnern, Quellcode-Kommentaren oder separaten Dokumenten stellt Informationen über den Zweck von Code-Fragmenten und den darin implementierten Funktionalitäten (Intension) sowie die zugeordneten Namen bereit und macht damit den Code (Extension) erst verständlich. Für die Umsetzung einer Änderung muss das Wartungspersonal die betroffenen Funktionalitäten aus der abstrakten Änderungsanforderung selektieren und beispielsweise über die zugeordneten Namen die entsprechenden, zu modifizierenden Code-Fragmente lokalisieren. Dabei ist auch umgekehrt die Analyse verschiedener Code-Fragmente auf die darin implementierten Funktionalitäten und deren Interaktionen erforderlich. Eben diese Vorgänge des Verstehens können durch die Verfügbarkeit einer adäquaten technischen Programmdokumentation erheblich unterstützt werden.

Als Ergebnis ihrer explorativen Studie beschreiben Ko et al. (2006), wie Entwickler unbekannten Code für die Umsetzung nachträglicher Änderungen im Rahmen der Softwarewartung analysieren. Nachdem sie ein potenziell relevantes Code-Fragment identifiziert haben, navigieren die Entwickler über deren Abhängigkeiten zu angrenzenden Code-Fragmenten, um die Funktionalität des ursprünglichen Fragments sowie deren Interaktionen mit der Umgebung zu verstehen und um gegebenenfalls weitere betroffene Code-Stellen zu suchen. Diese Navigation nimmt dabei durchschnittlich 35% der für die Umsetzung einer Änderung benötigten Zeit ein und kann durch eine adäquate technische Programmdokumentation, welche zusätzliche Informationen über den Zweck und die Verwendung des Codes bereitstellt, unterstützt werden. Ebenfalls im Rahmen einer explorativen Studie arbeiten Robillard et al. (2004) verschiedene Charakteristika der Analyse von Softwaresystemen heraus, bezüglich derer sich bei der Umsetzung von nachträglichen Änderungen erfolgreiche von nicht erfolgreichen Entwicklern unterscheiden. So nehmen insbesondere erfolglose Entwickler für die Umsetzung einer Änderungsanforderung Modifikationen an lediglich einer Stelle im Code vor, obwohl die Modifikation mehrerer Code-Fragmente für eine konsistente und dem existierenden Design besser entsprechende Änderung erforderlich gewesen wäre. Zudem orientieren sich erfolgreiche Entwickler bei der Suche nach relevanten Code-Fragmenten an der Struktur der Software, also unter anderem an den Abhängigkeiten zwischen verschiedenen Code-Fragmenten, und vermeiden eine unstrukturierte Inspektion des Codes. Als weitere Beobachtung beschreiben die Autoren, dass für eine Änderung relevante Code-

Fragmente oftmals nicht als solche erkannt werden, sofern die Entwickler diese Code-Fragmente nicht gezielt inspizieren, sondern zufällig oder versehentlich darauf stoßen. Diese Beobachtungen erlauben die Schlussfolgerung, dass insbesondere das solide Verständnis von Abhängigkeiten zwischen verschiedenen Softwarekomponenten eine wichtige Rolle für die Vermeidung inkonsistenter Modifikationen im Rahmen der Softwarewartung spielt, und dass eine adäquate technische Programmdokumentation das Verstehen dieser Abhängigkeiten erheblich unterstützen kann.

Ferner existieren zahlreiche Beiträge, die verschiedene Ansätze zur Prognose von Fehlerhäufigkeiten (*fault*), des Auftretens von Versagensfällen (*failure*) oder allgemeiner von Softwarezuverlässigkeit vorschlagen. Grottke (2005) stellt einen umfassenden Überblick der existierenden Ansätze zur Prognose von Softwarezuverlässigkeit dar. Dabei bilden sogenannte Softwarezuverlässigkeitswachstumsmodelle die bedeutendste Modellklasse. Derartige Modelle berechnen aus dem Verlauf der während der Testphase aufgetretenen Versagensfälle eine Prognose für die Zuverlässigkeit während des Betriebs. Die auftretenden Versagensfälle werden dabei mithilfe stochastischer Prozesse modelliert. Als weitere Klasse werden statistische Modelle dargestellt, welche die aktuelle Fehleranzahl oder die Zuverlässigkeit einer Software mittels Stichproben schätzen. Eine dritte Klasse von Modellen versucht den Fehlergehalt der Software auf Basis von Informationen über das Softwareprodukt und seine Erstellung vorherzusagen. Erstaunlicherweise berücksichtigt keiner der vorgestellten Ansätze die Verfügbarkeit einer technischen Programmdokumentation als Einflussfaktor auf die Anzahl der Fehler aufgrund nachträglicher Änderungen. Zwar zeigen beispielsweise Takahashi und Kamayachi (1989, S. 83-84) einen signifikanten Zusammenhang zwischen dem Umfang der Designdokumentation und der Anzahl der Fehler, welche zu Beginn der Testphase während der Entwicklung in der Software verbleiben. Auf dieser Basis schlägt der Beitrag ein lineares Regressionsmodell zur Prognose der Fehleranzahl vor. Jedoch liegt der Fokus dabei nicht auf der Entstehung von Fehlern während der Softwarewartung.

Die Recherche nach Beiträgen, welche ökonomische Modelle für die Berechnung der Kosten zur Vermeidung von Fehlern im Rahmen der Softwarewartung beinhalten, führte zu keinem Ergebnis. Beiträge mit entsprechenden Modellen aus anderen Kontexten, welche als Grundlage für eine Adaption in den Kontext der Softwarewartung dienen können, brachte die Suche ebenfalls nicht hervor. An dieser Stelle setzt der vorliegende Beitrag an und stellt ein neu entwickeltes, formal-deduktives Modell vor, welches die barwertigen Auszahlungen für die Erstellung einer technischen Programmdokumentation und für die Fehlerbehebung gegenüberstellt sowie die Generierung neuer

Hypothesen ermöglicht.

1.5 Modell

1.5.1 Grundlegende Annahmen

Im Folgenden wird die Größe einer Software SW in der Anzahl ihrer Komponenten gemessen. Eine Softwarekomponente kann dabei als Funktion, Methode, Klasse, Modul oder ähnliches verstanden werden. Diese Komponenten stehen über gegenseitige Abhängigkeiten wie beispielsweise Methodenaufrufe, verschiedene Formen des Datenaustausches, Schnittstellen oder Cross-Cutting Concerns in Beziehung zueinander. Jede Komponente, die über derartige Abhängigkeiten eine Verbindung zu einer bestimmten Komponente besitzt, wird als deren *angrenzende Komponente* bezeichnet. Da auf Basis der bisherigen Literatur ein Zusammenhang zwischen der Fehleranfälligkeit einer Komponente und der Anzahl ihrer angrenzenden Komponenten gesichert erscheint (Briand et al. 2002, S. 714, Gyimóthy et al. 2005, S. 909, Singh et al. 2010, S. 31) und sich Fehler bei nachträglichen Änderungen insbesondere über Abhängigkeiten auf angrenzende Komponenten ausbreiten können (Beszédes et al. 2007, S. 296), dient im Folgenden die Anzahl der angrenzenden Komponenten einer Komponente als Maß der strukturellen Komplexität von SW. In der Realität dürften sich verschiedene Komponenten in der Anzahl ihrer angrenzenden Komponenten deutlich voneinander unterscheiden. Dennoch wird im Folgenden vereinfachend angenommen, dass jede Komponente von SW gleich viele angrenzende Komponenten besitzt. Dabei sollte für diesen Wert die durchschnittliche Anzahl der angrenzenden Komponenten pro Komponente gewählt werden. Vergleichbar zu MacCormack et al. (2006, S. 1020) kann dieser Wert als Maß für die durchschnittliche Anzahl der von der Modifikation einer Komponente potenziell direkt betroffenen Stellen in SW dienen. An dieser Stelle sei zudem vorweggenommen, dass der konkrete Wert keine Bedeutung für die im Rahmen der Modellanalyse deduzierte Hypothese besitzt.

Im Gegensatz zur Anzahl der Abhängigkeiten ist der Einfluss der Größe und Komplexität einer Komponente auf deren Fehleranfälligkeit nicht gesichert. So können beispielsweise Fenton und Ohlsson (2000, S. 812) entsprechende Zusammenhänge nicht bestätigen. Daher werden diese im Folgenden vernachlässigt.

(A1) Die Entwicklung von SW findet in $t = 0$ statt. Die Größe von SW wird in der Anzahl ihrer Komponenten m $(m \in \mathbb{N} | m \geq 2)$ gemessen. Die (durchschnittliche) Anzahl der angrenzenden Komponenten, zu denen eine Abhängigkeit besteht, ist für jede Komponente gleich und wird als a $(a \in \mathbb{R}^{+} | a \geq 1)$ bezeichnet. Die Größe und Komplexität der Komponenten selbst werden nicht berücksichtigt.

Gemäß Dzidek et al. (2008, S. 417) geht mit der Verfügbarkeit einer adäquaten technischen Programmdokumentation insbesondere eine Reduktion der Fehler einher, die auf ein Unverständnis der Abhängigkeiten verschiedener Softwarekomponenten beruhen. Zudem zeigen Eaddy et al. (2008, S. 498), dass Cross-Cutting Concerns als schwer ersichtliche und meist ungenügend dokumentierte Abhängigkeiten die Verständlichkeit einer Software erheblich einschränken und die Ursache für Fehler bilden können. Daher wird im Folgenden die Dokumentation von Abhängigkeiten zwischen jeweils zwei Komponenten betrachtet. Diese wird als technische Programmdokumentation mit zusätzlichen Informationen sowohl über den Zweck und die Funktionalitäten der relevanten Code-Fragmente in beiden Komponenten als auch über deren Abhängigkeit beziehungsweise deren Interaktion verstanden. Die *Dokumentation einer Abhängigkeit* gibt also Aufschluss darüber, welche Funktionalität(en) zwei Code-Fragmente in unterschiedlichen Komponenten umsetzen und wie diese Code-Fragmente in ihrer Funktionalität zusammenwirken. Sie kann dabei zum Beispiel in Form von Quellcode-Kommentaren, Diagrammen, externen Dokumenten oder einer begreiflichen Bezeichnung von Variablen, Methoden, Klassen etc. vorliegen.

Weiterhin wird angenommen, dass die Dokumentation einer Abhängigkeit eine Auszahlung in stets gleicher Höhe bedeutet. In der Realität dürften sich die Auszahlungen für die Dokumentation einer Abhängigkeit je nach Beschaffenheit der Abhängigkeiten sowie der relevanten Code-Fragmente in den jeweils angrenzenden Komponenten voneinander unterscheiden. Auch der Umfang und die Qualität der zu erstellenden technischen Programmdokumentation werden sich vermutlich auf die Höhe der Auszahlungen auswirken. Daher sollte für diese Auszahlungen ein durchschnittlicher Wert verwendet werden. Wie im sechsten Abschnitt gezeigt werden kann, spielt auch dieser konkrete Wert für die im Rahmen der Modellanalyse abgeleitete Hypothese keine Rolle. Da die technische Programmdokumentation zur Verbesserung der Verständlichkeit von SW zusätzliche Informationen über die in den relevanten Code-Fragmenten realisierten Funktionalitäten (Rajlich 2009, S. 4) sowie über deren Zusammenwirken bereitstellen soll, wird zudem ein linearer Zusammenhang zwischen der Anzahl der dokumentierten Abhängigkeiten und den dafür anfallenden Auszahlungen angenommen.

Wie bereits in Abschnitt 2 ausgeführt wird die Dokumentation auf die Entwicklung von SW beschränkt und eine nachträgliche Dokumentation während der Wartung nicht betrachtet.

(A2) Für jede Abhängigkeit zwischen zwei angrenzenden Komponenten können zusätzliche Informationen über die Funktionalitäten der jeweils relevanten Code-Fragmente in den beiden Komponenten sowie über deren Zusammenwirken bereitgestellt werden. Die Dokumentation einer Abhängigkeit führt in $t = 0$ *zu einer (durchschnittlichen) Auszahlung in Höhe von* c_d $(c_d \in \mathbb{R}^+)$*. Der relative Anteil der dokumentierten Abhängigkeiten wird als Dokumentationsgrad* r_d $(r_d \in \mathbb{R} | 0 \leq r_d \leq 1)$ *bezeichnet. Die dokumentierten Abhängigkeiten verteilen sich gleichmäßig auf alle Komponenten in SW.*

Die Softwarewartung wird auf den Zeitraum nach Fertigstellung und Inbetriebnahme von SW beschränkt. Die Anpassung an geänderte Anforderungen erfolgt dabei ausschließlich mittels Modifikationen an einem bestimmten, gleichbleibenden Anteil der existierenden Komponenten in jeder Periode während des Betriebs. In der Realität wird zur Anpassung einer Software an alle sich ändernden Anforderungen die Modifikation eines pro Periode jeweils unterschiedlichen Anteils der existierenden Komponenten erforderlich sein. Jedoch unterliegt die Softwarewartung oftmals einer festen Budgetrestriktion (Krishnan et al. 2004, S. 396) oder einem Unternehmen steht nur eine feste Anzahl von Mitarbeitern für die entsprechenden Änderungen zur Verfügung. Daher können in der Regel auch nicht alle neuen Anforderungen umgesetzt werden. Aufgrund solcher Restriktionen ist meist nur die Änderung eines pro Periode jeweils ähnlich großen Anteils der vorhandenen Komponenten realisierbar. Daher erscheint eine solche Annahme vertretbar. Da sich dieser Anteil dennoch in einem gewissen Umfang von Periode zu Periode unterscheiden wird, sollte bei der Anwendung des Modells ein durchschnittlicher Wert verwendet werden.

Außerdem erfolgt die Anpassung einer Software in der Praxis oftmals nicht nur über die Änderung bestehender Komponenten, sondern auch über entsprechende Erweiterungen (Krishnan et al. 2004, S. 396). Als Vereinfachung beschränkt sich das vorliegende Modell jedoch ausschließlich auf die Auswirkungen des Dokumentationsgrads auf die Fehleranfälligkeit von Änderungen der existierenden Komponenten. Dementsprechend werden Erweiterungen von SW um neue Komponenten oder um zusätzliche Abhängigkeiten vernachlässigt. Wie im Rahmen der kritischen Würdigung diskutiert, dürften solche Erweiterungen von SW die Fehleranfälligkeit von nachgelagerten Wartungs-

maßnahmen jedoch sogar noch erhöhen.

(A3) *Der Betrieb und die Wartung von SW erfolgen von $t = 1$ bis einschließlich $t = n$ ($n \in \mathbb{N}$). Während des Betriebs wird SW jeweils zu Beginn einer jeden Periode an geänderte Anforderungen angepasst. Dies geschieht pro Periode ausschließlich über die Änderung eines jeweils gleichen (durchschnittlichen) Anteils r_c ($r_c \in \mathbb{R}^+$) der existierenden Komponenten. Die Anzahl der Komponenten m und die Anzahl der Abhängigkeiten a bleiben während der Wartung konstant.*

Abb. III-1 veranschaulicht den Zusammenhang, der für die Entstehung von Fehlern aufgrund der Modifikation einer Komponente zur Anpassung an geänderte Anforderungen angenommen wird. Dabei wird insbesondere der Effekt berücksichtigt, dass die Modifikation einer Komponente inkonsistent zu den angrenzenden Komponenten sein kann. Aufgrund der daraus resultierenden Fehler können Änderungen in den angrenzenden Komponenten erforderlich sein, welche wiederum neue Änderungen bedingen und so weiter (Beszédes et al. 2007, S. 296).

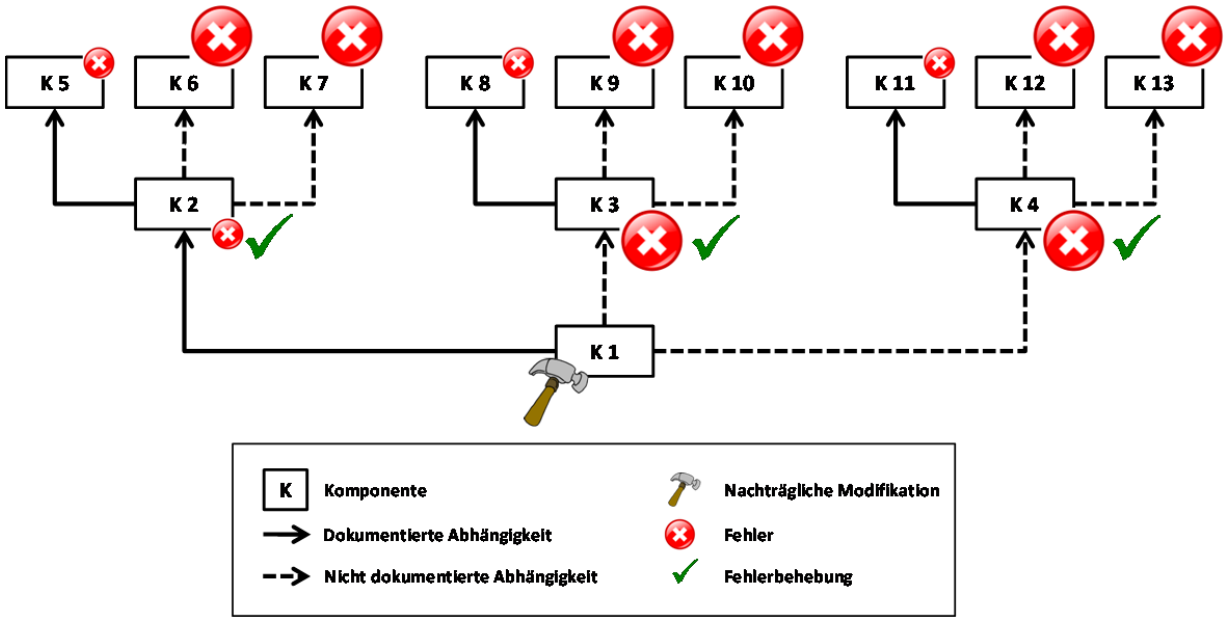

Abb. III-1: Entstehung und Ausbreitung von Fehlern aufgrund nachträglicher Änderungen

In diesem vereinfachten Beispiel besitzt jede Komponente drei angrenzende Komponenten, wobei jeweils die Abhängigkeit zu einer angrenzenden Komponente dokumentiert ist und die Abhängigkeiten zu den beiden anderen Komponenten nicht dokumentiert

sind. Bei einer Modifikation von Komponente K 1 wird unterstellt, dass der zuständige Entwickler die beiden nicht dokumentierten Abhängigkeiten zu den Komponenten K 3 und K 4 unter Umständen nicht vollständig versteht. Als Folge verändert der Entwickler K 1 derart, dass K 3 und K 4 mit einer gewissen Wahrscheinlichkeit nicht mehr konsistent und fehlerfrei mit K 1 interagieren. Dementsprechend entstehen mit einer bestimmten Wahrscheinlichkeit Fehler in den Komponenten K 3 und K 4. Hingegen steht für die Abhängigkeit zwischen K 1 und der angrenzende Komponente K 2 eine entsprechende technische Programmdokumentation zur Verfügung. Abhängig vom Umfang und der Qualität der zusätzlichen Informationen wird angenommen, dass der zuständige Entwickler ein besseres Verständnis über diese Komponenten und deren Zusammenwirken gewinnen kann (Rajlich 2009, S. 4) und dementsprechend mit einer geringeren Wahrscheinlichkeit einen Fehler in K 2 verursacht (Dzidek et al. 2008, S. 424). Im zweiten Schritt werden die entstandenen Fehler behoben. Dazu müssen die Komponenten K 2, K 3 und K 4 mit den entsprechenden Wahrscheinlichkeiten ebenfalls modifiziert werden. Aufgrund desselben Effekts entstehen wiederum mit einer bestimmten (bedingten) Wahrscheinlichkeit Fehler in den Komponenten K 6, K 7, K 9, K 10, K 12 und K 13 sowie mit einer entsprechend geringeren (bedingten) Wahrscheinlichkeit Fehler in K 5, K 8 und K 11. Analog dazu hat eine wiederholte Fehlerbehebung eine fortlaufende Fehlerausbreitung zur Folge (Beszédes et al. 2007, S. 296). Aus Gründen der Übersichtlichkeit ist diese nicht mehr in Abb. III-1 dargestellt.

Wie bereits im vierten Abschnitt dargestellt ist ein reduzierender Effekt einer adäquaten technischen Programmdokumentation auf die Fehlerwahrscheinlichkeit von nachträglichen Änderungen empirisch mehrfach bestätigt (Arisholm et al. 2006, S. 365, Dzidek et al. 2008, S. 424, Prechelt et al. 2002, S. 595). Dennoch ist eine Situation denkbar, in der die verfügbare technische Programmdokumentation beispielsweise aufgrund einer sehr geringen Qualität keinerlei Mehrwert für das Verstehen von SW bietet. Dieser extreme Fall wird im Folgenden jedoch vernachlässigt. Vorausgesetzt wird eine hinreichend hohe Dokumentationsqualität, so dass zumindest ein beliebig kleiner Effekt der technischen Programmdokumentation auf die Fehlerwahrscheinlichkeit angenommen werden kann.

Die Annahme, dass bei der Änderung einer Komponente und bei der Fehlerbehebung pro nicht dokumentierte Abhängigkeit jeweils mit der stets gleichen Wahrscheinlichkeit neue Fehler in den angrenzenden Komponenten entstehen, erscheint auf den ersten Blick nicht mit der Realität vereinbar. So können sich die Komponenten einer Software zum Beispiel erheblich in der Beschaffenheit und Komplexität ihrer Abhängigkeiten

unterscheiden. Dementsprechend dürften für verschiedene Komponenten auch die Wahrscheinlichkeiten für die Entstehung neuer Fehler merklich voneinander abweichen. Ähnliches gilt für die Annahme, dass sich die Wahrscheinlichkeit für die Entstehung neuer Fehler bei allen dokumentierten Abhängigkeiten gleichermaßen reduziert. Vergleichbar zu der Anzahl der angrenzenden Komponenten a in Annahme (A1) sollten daher bei der Verwendung des Modells beispielsweise aus Erfahrungen in vergleichbaren Softwareprojekten gewonnene Durchschnittswerte herangezogen werden. Für eine Schätzung der Fehlerwahrscheinlichkeiten kann außerdem auf die Arbeiten von Abdelmoez et al. (2004) und von Abdelmoez et al. (2005) zurückgegriffen werden. Die konkreten Werte spielen jedoch für die in Abschnitt 6 abgeleitete Hypothese ebenfalls keine Rolle. Zeichnet sich ein Softwaresystem dennoch durch eine starke Heterogenität aus, sollte das Modell nicht auf die gesamte Software angewendet werden. Stattdessen sollte die Anwendung jeweils separat auf verschiedene Teile von SW erfolgen, für die ein bestimmtes Maß der Homogenität und damit ähnliche Wahrscheinlichkeiten für die Entstehung neuer Fehler angenommen werden können.

Um die vollständigen Auswirkungen der Fehlerausbreitung monetär abbilden zu können, wird angenommen, dass alle Fehler im Rahmen der Wartung unmittelbar erkannt und behoben werden. Die Korrektur eines Fehlers bedeutet eine Auszahlung in stets gleicher Höhe. Nach der Korrektur werden alle angrenzenden Komponenten getestet, wobei für das Testen einer Abhängigkeit wiederum eine Auszahlung in fester Höhe anfällt. Da sich in der Realität diese Auszahlungen je nach Schwere des Fehlers durchaus unterscheiden dürften, sollten auch für diese Parameter entsprechende Durchschnittswerte herangezogen werden.

(A4) Fehler in SW, die während der Entwicklung eingebaut und bei Inbetriebnahme noch nicht behoben wurden, werden nicht berücksichtigt. Die Änderung einer Komponente führt pro nicht dokumentierter Abhängigkeit mit einer (durchschnittlichen) Wahrscheinlichkeit e $(e \in \mathbb{R} | 0 \leq e \leq 1)$ *zu einem Fehler in der angrenzenden Komponente. Bei einer dokumentierten Abhängigkeit reduziert sich die Wahrscheinlichkeit für einen Fehler in der angrenzenden Komponente (durchschnittlich) um* ε $(\varepsilon \in \mathbb{R} | 0 < \varepsilon \leq e)$*. Für die Behebung eines Fehlers gilt der gleiche Effekt. Alle eingebauten Fehler werden in derselben Periode erkannt und behoben. Die Behebung eines Fehlers bedeutet eine (durchschnittliche) Auszahlung in Höhe von* c_b $(c_b \in \mathbb{R}^+)$ *für die Fehlerkorrektur in der Komponente sowie pro angrenzende Komponente eine (durchschnittliche) Auszahlung in Höhe von* c_s $(c_s \in \mathbb{R}^+)$ *für das Testen der Abhängigkeit.*

Schließlich werden ein risikoneutraler Entscheider sowie ein gegebener Kalkulationszinssatz unterstellt.

(A5) Der Entscheider ist risikoneutral. Es gilt der Kalkulationszinssatz i $(i > 0)$.

Ziel ist die Modellierung des ökonomischen Trade-off bezüglich der technischen Programmdokumentation von SW. Dazu werden im Folgenden die barwertigen Auszahlungen für die Erstellung der technischen Programmdokumentation und für die Fehlerbehebung als bewertungsunabhängige Finanzstromgrößen aus den Annahmen abgeleitet.

Tab. III-1: Notationen

Variable	
r_d	Dokumentationsgrad: Relativer Anteil der dokumentierten Abhängigkeiten
Parameter	
m	Anzahl der Komponenten in SW
a	Anzahl der angrenzenden Komponenten (bzw. Abhängigkeiten) pro Komponente
r_c	Relativer Anteil der zu einem Zeitpunkt t modifizierten Komponenten
e	Wahrscheinlichkeit für die Entstehung eines Fehlers in der angrenzenden Komponente bei einer nicht dokumentierten Abhängigkeit
ε	Subtrahend, um den sich die Wahrscheinlichkeit für die Entstehung eines Fehlers in der angrenzenden Komponente bei Verfügbarkeit einer technischen Programmdokumentation reduziert
θ	Erwartungswert der Fehleranzahl aufgrund der Modifikation einer Komponente
c_d	Auszahlung für die Dokumentation einer Abhängigkeit
c_b	Auszahlung für die Korrektur eines Fehlers
c_s	Auszahlung für das Testen einer Abhängigkeit bzw. einer angrenzenden Komponente
i	Kalkulationszinssatz
n	Anzahl der betrachteten Perioden

1.5.2 Funktion der barwertigen Auszahlungen für Dokumentation und Fehlerbehebung

Die Auszahlungen CF_d für die Dokumentation der Abhängigkeiten ergeben sich als Produkt der Anzahl aller existierenden Abhängigkeiten in SW, des Dokumentationsgrads r_d und der Auszahlungen c_d für die Dokumentation einer einzelnen Abhängigkeit. Die Anzahl aller existierenden Abhängigkeiten setzt sich wiederum aus der Anzahl der Komponenten m und der Anzahl a der pro Komponente angrenzenden Komponenten zusammen:

$$CF_d(r_d) = m \cdot a \cdot r_d \cdot c_d$$

Da die Dokumentation der Abhängigkeiten während der Entwicklung von SW in $t = 0$ stattfindet, entspricht $CF_d(r_d)$ den barwertigen Auszahlungen für die Erstellung der technischen Programmdokumentation.

Die Auszahlungen CF_r für die Fehlerbehebung während der Wartung von SW setzen sich aus den jeweiligen Auszahlungen $CF_{r,t}$ für die Fehlerbehebung zu den Zeitpunkten $t = 1$ bis $t = n$ zusammen. $CF_{r,t}$ ergibt sich wiederum als Produkt der Anzahl der zum jeweiligen Zeitpunkt t modifizierten Komponenten $r_c \cdot m$, der Anzahl der entstehenden Fehler θ bei der Modifikation einer Komponente und der Auszahlungen für die Behebung eines Fehlers. Dabei bestehen die Auszahlungen für die Behebung eines Fehlers gemäß (A4) aus den Auszahlungen c_b für die Korrektur des Fehlers sowie den Auszahlungen $c_s \cdot a$ für das Testen aller Abhängigkeiten zu den angrenzenden Komponenten:

$$CF_{r,t} = (c_b + c_s \cdot a) \cdot r_c \cdot m \cdot \theta$$

Laut (A4) führt die Modifikation einer Komponente pro nicht dokumentierter Abhängigkeit mit Wahrscheinlichkeit e und pro dokumentierter Abhängigkeit mit Wahrscheinlichkeit $e - \varepsilon$ zu einem Fehler in der jeweils angrenzenden Komponente. Folglich werden bei der Modifikation einer Komponente initial im Erwartungswert $(1 - r_d) \cdot a \cdot e + + r_d \cdot a \cdot (e - \varepsilon) = a \cdot (e - r_d \cdot \varepsilon)$ Fehler eingebaut. Da die Behebung dieser Fehler eine entsprechende Modifikation der betroffenen Komponenten erforderlich macht, führt die Behebung eines Fehlers wiederum im Erwartungswert zu $a \cdot (e - r_d \cdot \varepsilon)$ Fehlern. Da alle Fehler unmittelbar behoben werden, tritt dieser Effekt theoretisch unendlich oft in Folge auf. Die erwartete Anzahl aller Fehler θ, welche unter Berücksichtigung dieser Fehlerausbreitung in Summe aus der Modifikation einer Komponente resultieren, kann also durch folgende geometrische Reihe beschrieben werden:

$$\theta = \sum_{k=0}^{\infty} \big(a \cdot (e - r_d \cdot \varepsilon)\big) \cdot \big(a \cdot (e - r_d \cdot \varepsilon)\big)^k$$

Für die geometrische Reihe gilt:

$$\lim_{j \to \infty} \sum_{k=0}^{j} \big(a \cdot (e - r_d \cdot \varepsilon)\big) \cdot \big(a \cdot (e - r_d \cdot \varepsilon)\big)^k = \frac{a \cdot (e - r_d \cdot \varepsilon)}{\big(1 - a \cdot (e - r_d \cdot \varepsilon)\big)}$$

sofern

$|a \cdot (e - r_d \cdot \varepsilon)| < 1$ (Konvergenzkriterium)

Andernfalls divergiert die Reihe. Da $a \cdot (e - r_d \cdot \varepsilon) \geq 0$ gilt, kann die Bedingung für die Konvergenz dieser Reihe umgeformt werden in

$$r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon}$$

Ist also die Anzahl der pro Komponente nicht dokumentierten Abhängigkeiten $(1 - r_d) \cdot a$ multipliziert mit der Wahrscheinlichkeit e für die Entstehung eines Fehler bei einer nicht dokumentierten Abhängigkeit zuzüglich der Anzahl der pro Komponente dokumentierten Abhängigkeiten $r_d \cdot a$ multipliziert mit der entsprechend reduzierten Fehlerwahrscheinlichkeit $(e - \varepsilon)$ größer oder gleich 1, führt die Modifikation einer Komponente unter den getroffenen Annahmen zu unendlich vielen Fehlern und damit theoretisch auch zu unendlich hohen Auszahlungen für die Fehlerbehebung. Denn in diesem Fall wird mit der Behebung eines Fehlers im Erwartungswert mehr als ein neuer Fehler eingefügt. Da durch die Fehlerbehebung mehr Fehler eingebaut als behoben werden, folgt einer wie in (A4) angenommenen, fortlaufenden Fehlerbehebung eine stetig steigende Anzahl von Fehlern in SW. Ein solches Phänomen ist beispielsweise laut Parnas (1994, S. 281) auch in der Realität und bereits seit den frühen Jahren der Softwarebranche mehrfach beobachtbar.

Es gilt:

$$a \cdot e > 1 \quad \Rightarrow \quad r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} > 0$$

Zur Vermeidung einer unendlich großen Fehleranzahl ist ein Dokumentationsgrad $r_d > 0$ notwendig, sofern die Anzahl der angrenzenden Komponenten a multipliziert mit der Wahrscheinlichkeit e für die Entstehung eines Fehler bei einer nicht dokumentierten Abhängigkeit größer als 1 ist. Gegeben sei zur Illustration das folgende Beispiel: Jede

Komponente in SW besitzt fünf angrenzende Komponenten. Zudem entsteht bei der Modifikation einer Komponente mit einer Wahrscheinlichkeit von 25% ein Fehler in einer angrenzenden Komponente mit nicht dokumentierter Abhängigkeit. Bei Verfügbarkeit einer technischen Programmdokumentation reduziert sich diese Wahrscheinlichkeit um 10% pro angrenzende Komponente. In diesem Fall führt nur ein Dokumentationsgrad $r_d > 50\%$ zu einer endlichen Fehleranzahl. Ein kleinerer Dokumentationsgrad würde zu unendlich vielen Fehlern führen und eine Stabilisierung von SW könnte im Rahmen der Wartung nicht mehr gelingen.

Darüber hinaus gilt:

$$a \cdot (e - \varepsilon) \geq 1 \quad \Rightarrow \quad \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \geq 1$$

Ist die Anzahl der pro Komponente angrenzenden Komponenten a multipliziert mit der Wahrscheinlichkeit $(e - \varepsilon)$ für die Entstehung eines Fehlers bei einer dokumentierten Abhängigkeit größer oder gleich 1, führt die Modifikation einer Komponente unter den getroffenen Annahmen selbst bei einem Dokumentationsgrad $r_d = 100\%$ zu unendlich vielen Fehlern.

Der Erwartungswert der Auszahlungen für die Behebung aller Fehler zu einem Zeitpunkt t lässt sich also mit einer entsprechenden Einschränkung der Definitionsmenge folgendermaßen formalisieren:

$$CF_{r,t}(r_d) = (c_b + c_s \cdot a) \cdot r_c \cdot m \cdot \frac{a \cdot (e - r_d \cdot \varepsilon)}{\left(1 - a \cdot (e - r_d \cdot \varepsilon)\right)}$$

mit

$$\mathbb{D} = \left\{ r_d \in \mathbb{R} | 0 \leq r_d \leq 1 \wedge a \cdot (e - \varepsilon) < 1 \wedge r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \right\}$$

Da gemäß (A3) zu Beginn jeder Periode während des gesamten Betriebs von SW der gleiche Anteil r_c der existierenden Komponenten modifiziert wird, fallen die Auszahlungen für die Fehlerbehebung zu den Zeitpunkten $t = 1$ bis $t = n$ in jeweils gleicher Höhe an. Diese lassen sich somit als gleich bleibende Rente von $t = 1$ bis $t = n$ darstellen. Der erwartete Barwert aller Auszahlungen für die Fehlerbehebung während der Wartung ergibt sich folglich als Produkt der erwarteten Auszahlungen für die Fehlerbehebung zu einem Zeitpunkt t und des Rentenbarwertfaktors β für eine gleich bleibende, nachschüssige Rente (Kruschwitz 2006, S. 48-53):

$$CF_r(r_d) = CF_{r,t}(r_d) \cdot \beta$$

wobei

$$\beta = \frac{(1+i)^n - 1}{i \cdot (1+i)^n}$$

Damit kann schließlich der erwartete Barwert der Auszahlungen für die Erstellung der technischen Programmdokumentation und für die Fehlerbehebung festgehalten werden:

$$CF(r_d) = CF_d(r_d) + CF_r(r_d)$$

$$CF(r_d) = m \cdot a \cdot r_d \cdot c_d + (c_b + c_s \cdot a) \cdot r_c \cdot m \cdot \frac{a \cdot (e - r_d \cdot \varepsilon)}{\left(1 - a \cdot (e - r_d \cdot \varepsilon)\right)} \cdot \beta$$

mit

$$\mathbb{D} = \left\{ r_d \in \mathbb{R} | 0 \leq r_d \leq 1 \wedge a \cdot (e - \varepsilon) < 1 \wedge r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \right\}$$

Der Zusammenhang zwischen dem Dokumentationsgrad und dem Erwartungswert der barwertigen Auszahlungen für die Dokumentation und Fehlerbehebung wird im Folgenden näher analysiert. Ziel dieser Analyse ist die Deduktion weiterer, empirisch prüfbarer Hypothesen, welche vermutlich nicht unmittelbar aus Beobachtungen in der Realität ableitbar sein dürften.

1.6 Analyse des Modells: Diskussion der Auszahlungsfunktion

Für die Bestimmung des auszahlungsminimalen Dokumentationsgrads r_d^* muss das Minimum der Auszahlungsfunktion $CF(r_d)$ berechnet werden. Durch das Gleichsetzen der ersten Ableitung mit 0 und dem anschließenden Auflösen der Gleichung nach r_d kann ein möglicher Extrempunkt von $CF(r_d)$ ermittelt werden:

$$r_{d,1} = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} + \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon}$$

Dabei liegt $r_{d,1}$ unter folgenden Bedingungen innerhalb des Definitionsbereichs $\mathbb{D}$:

$$c_d \geq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - \varepsilon)\right)^2} \quad \Leftrightarrow \quad r_{d,1} \leq 1$$

$$(a \cdot e \geq 1) \quad \vee \quad \left(a \cdot e < 1 \quad \wedge \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2} \right) \quad \Leftrightarrow \quad r_{d,1} \geq 0$$

Da die zweite Ableitung an diesem Punkt größer ist als 0, besitzt die Auszahlungsfunktion $CF(r_d)$ unter den obigen Bedingungen bei $r_{d,1}$ ein globales Minimum. In diesem

Fall entspricht $r_{d,1}$ also dem auszahlungsminimalen Dokumentationsgrad r_d^*:

$$\left(c_d \geq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - \varepsilon)\right)^2}\right) \quad \wedge$$

$$\left((a \cdot e \geq 1) \quad \vee \quad \left(a \cdot e < 1 \quad \wedge \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}\right)\right)$$

$$\Rightarrow r_d^* = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} + \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon}$$

Ist eine der beiden obigen Bedingungen nicht erfüllt, liegt $r_{d,1}$ außerhalb von $\mathbb{D}$:

$$c_d < \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - \varepsilon)\right)^2} \quad \Rightarrow \quad r_{d,1} > 1 \quad \Rightarrow \quad r_d^* = 1$$

$$a \cdot e < 1 \quad \wedge \quad c_d > \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2} \quad \Rightarrow \quad r_{d,1} < 0 \quad \Rightarrow \quad r_d^* = 0$$

Aufgrund der Einschränkung $0 \leq r_d \leq 1$ ist $CF(r_d)$ bei $r_d^* = 1$ minimal, sofern $r_{d,1} > 1$. Analog dazu gilt $r_d^* = 0$, sofern $r_{d,1} < 0$.

Nach der Bestimmung des auszahlungsminimalen Dokumentationsgrads r_d^* werden im Folgenden Abweichungen von r_d^* untersucht:

Ungeachtet der Einschränkung $0 \leq r_d \leq 1$ weist die Funktion der barwertigen Auszahlungen $CF(r_d)$ eine positive Krümmung auf:

$$r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \quad \Rightarrow \quad \frac{d^2 CF(r_d)}{dr_d{}^2} > 0$$

Die Krümmung nimmt dabei mit steigendem Wert für den Dokumentationsgrad r_d stetig ab:

$$r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \quad \Rightarrow \quad \frac{d^3 CF(r_d)}{dr_d{}^3} < 0$$

Die Steigung der Auszahlungsfunktion $CF(r_d)$ ändert sich also bei einem niedrigeren Wert für r_d stärker als bei einem höheren Wert für r_d. Daraus folgt, dass die betragsmäßige Steigung von $CF(r_d)$ bei einer negativen Abweichung vom optimalen Dokumentationsgrad r_d^* größer ist als bei einer positiven Abweichung in gleicher Höhe:

$$\left|\frac{dCF(r_d^* - \Delta r_d)}{d(r_d^* - \Delta r_d)}\right| > \left|\frac{dCF(r_d^* + \Delta r_d)}{d(r_d^* + \Delta r_d)}\right| \quad \text{mit} \quad r_d^* - \Delta r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \quad \wedge \quad \Delta r_d > 0$$

Dementsprechend nimmt auch die Auszahlungsfunktion bei einer negativen Abweichung von r_d^* einen größeren Wert an als bei einer positiven Abweichung in gleicher Höhe:

$$CF(r_d^* - \Delta r_d) > CF(r_d^* + \Delta r_d) \quad \text{mit} \quad r_d^* - \Delta r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \quad \wedge \quad \Delta r_d > 0$$

Für den Definitionsbereich $\mathbb{D}$ kann damit folgende Hypothese formuliert werden:

$$\forall\, \Delta r_d > 0 \quad \left(r_d^*, (r_d^* - \Delta r_d), (r_d^* + \Delta r_d) \in \mathbb{D} \longrightarrow CF(r_d^* - \Delta r_d) > CF(r_d^* + \Delta r_d)\right)$$

Unter den Einschränkungen der getroffenen Annahmen ist eine positive Abweichung vom optimalen Dokumentationsgrad stets vorteilhaft im Vergleich zu einer betragsmäßig gleichen, negativen Abweichung vom optimalen Dokumentationsgrad.

Abb. III-2 illustriert diese Hypothese anhand eines beispielhaften Verlaufs der Auszahlungsfunktion $CF(r_d)$.

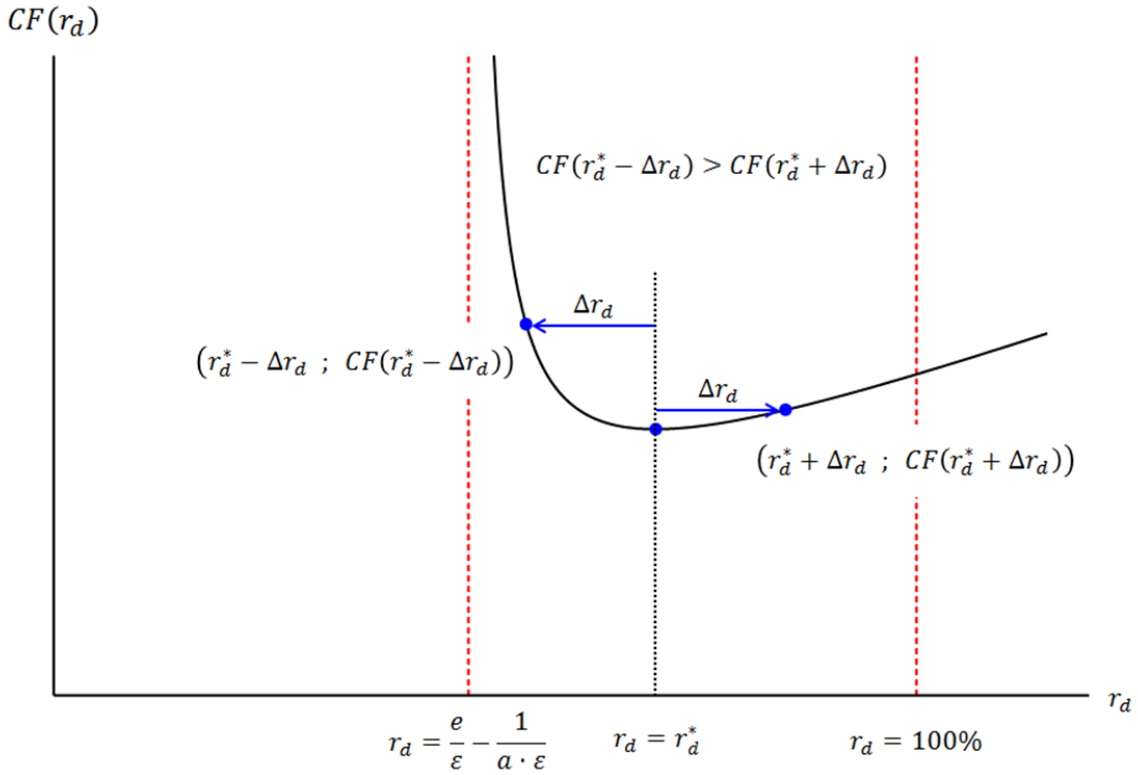

Abb. III-2: Abweichungen vom optimalen Dokumentationsgrad r_d^*

1.7 Kritische Würdigung

1.7.1 Robustheit der Ergebnisse

Bemerkenswert ist, dass die im vorherigen Abschnitt abgeleitete Hypothese unter den Einschränkungen der getroffenen Annahmen für alle zulässigen Werte aller gegebenen Parameter Gültigkeit besitzt. Diese gilt dabei insbesondere auch für jeden beliebig kleinen Wert des Parameters ε. Das heißt, solange die verfügbare technische Programmdokumentation irgendeinen reduzierenden Effekt auf die Fehlerwahrscheinlichkeit von nachträglichen Modifikationen ausübt $(\varepsilon > 0)$, welcher empirisch mehrfach bestätigt ist (Arisholm et al. 2006, S. 365, Dzidek et al. 2008, S. 424, Prechelt et al. 2002, S. 595), bedeutet eine positive Abweichung vom optimalen Dokumentationsgrad immer niedrigere Auszahlungen als eine betragsmäßig gleiche, negative Abweichung. Wie groß dieser Effekt ist, also um welchen Wert sich die Fehlerwahrscheinlichkeit bei Verfügbarkeit einer technischen Programmdokumentation reduziert, spielt dabei keine Rolle. Unter der Annahme, dass sich die Qualität der technischen Programmdokumentation auf den Parameter ε auswirkt, besitzt die Dokumentationsqualität keine Bedeutung für die obige Hypothese. Auch der konkrete Wert für den Parameter e schränkt die Gültigkeit dieser Hypothese nicht ein. Selbst bei einer beliebig kleinen Wahrscheinlichkeit, dass die Modifikation einer Komponente einen Fehler in einer angrenzenden Komponente mit nicht dokumentierter Abhängigkeit verursacht, ist für $0 < \varepsilon \leq e$ eine positive Abweichung vom optimalen Dokumentationsgrad stets vorteilhaft im Vergleich zu einer gleich hohen, negativen Abweichung.

Darüber hinaus ist die Hypothese ebenfalls unabhängig vom Wert des Rentenbarwertfaktors β und damit insbesondere auch für jeden beliebig hohen Kalkulationszinssatz i gültig. Gemäß den obigen Annahmen fallen die Auszahlungen für die Erstellung der technischen Programmdokumentation in $t = 0$ und die Auszahlungen für die Fehlerbehebung zu den Zeitpunkten $t = 1$ bis $t = n$ an. Im Gegensatz zu den gegenwärtigen Auszahlungen für die Dokumentation wertet also ein positiver Kalkulationszinssatz die zukünftigen Auszahlungen für die Fehlerbehebung auf ihren Barwert ab. Eine zunehmend negative Abweichung vom optimalen Dokumentationsgrad würde die gegenwärtigen Auszahlungen für die Dokumentation reduzieren und im Gegenzug die Anzahl der Fehler und damit die zukünftigen Auszahlungen für die Fehlerbehebung erhöhen. Der Zinseffekt würde sich dabei zunehmend positiv auf die barwertigen Auszahlungen $CF(r_d)$ auswirken. Eine zunehmend positive Abweichung vom optimalen Dokumentationsgrad hätte dagegen eine zunehmend negative Auswirkung des Zinseffekts

auf $CF(r_d)$ zur Folge, da die gegenwärtigen Auszahlungen für die Dokumentation erhöht und die zukünftigen Auszahlungen für die Fehlerbehebung reduziert würden. Dennoch übertrifft die Veränderung der Auszahlungen für die Fehlerbehebung im Falle einer Abweichung von r_d^* selbst bei einem beliebig hohen Kalkulationszinssatz stets die Auswirkungen des Zinseffekts. Auch beliebig hohe Auszahlungen c_d für die Erstellung der technischen Programmdokumentation oder beliebig niedrige Werte für den Anteil r_c der zu modifizierenden Komponenten schränken die Gültigkeit der obigen Hypothese nicht ein.

Des Weiteren weist die Hypothese ebenfalls eine gewisse Robustheit bezüglich des Zusammenhangs zwischen der Anzahl der dokumentierten Abhängigkeiten und den Auszahlungen für die Erstellung der technischen Programmdokumentation auf. Für die Erstellung des Modells wird in Abschnitt 5 ein linearer Zusammenhang angenommen. Es kann jedoch gezeigt werden, dass die obige Hypothese ebenfalls auf der Basis einer quadratischen Auszahlungsfunktion für die Dokumentation ableitbar ist. Erst bei einem kubischen Zusammenhang, welcher jedoch nicht realistisch erscheint, kann diese Hypothese nicht mehr für alle Werte der gegebenen Parameter deduziert werden.

1.7.2 Limitationen des Modells und der Ergebnisse

Da das Modell auf vereinfachenden Annahmen beruht, unterliegen die Ergebnisse und deren Aussagekraft entsprechenden Einschränkungen.

So werden in der Annahme (A3) jegliche Erweiterungen von SW um neue Komponenten im Rahmen der Wartung ausgeschlossen. Dementsprechend bildet das Modell nur die Auswirkungen der technischen Programmdokumentation auf die Fehleranfälligkeit von Änderungen existierender Komponenten bei konstantem Umfang und gleichbleibender struktureller Komplexität ab. In der Realität erfolgt eine Anpassung an neue Anforderungen sowohl über Änderungen existierender Komponenten als auch durch Erweiterungen der Software um neue Komponenten (Krishnan et al. 2004, S. 396). Dabei haben derartige Erweiterungen in der Regel einen Anstieg der strukturellen Komplexität zur Folge, da sie die Anzahl der Interaktionen und Abhängigkeiten zwischen den Komponenten einer Software erhöhen (Lehman 1996, S. 109). Da zugleich die Fehleranfälligkeit einer Softwarekomponente umso höher ist, desto mehr angrenzende Komponenten diese besitzt (Briand et al. 2002, S. 714), dürften nachträgliche Erweiterungen um neue Komponenten ebenfalls die Fehleranfälligkeit erhöhen. Folglich blieben die vorgestellten Ergebnisse nicht nur bestehen, sondern würden sogar noch verschärft.

Darüber hinaus betrachtet das Modell ausschließlich die Auswirkungen der Dokumentation von Abhängigkeiten auf Fehler, welche sich über diese Abhängigkeiten auf angrenzende Komponenten ausbreiten (Beszédes et al. 2007, S. 296). Die vorgestellten Ergebnisse besitzen daher ausschließlich Gültigkeit im Rahmen dieser Einschränkung. Vernachlässigt werden die Dokumentation von Code-Fragmenten ohne Abhängigkeiten zu angrenzenden Komponenten sowie alle Fehler innerhalb einer Komponente ohne Auswirkungen auf angrenzende Komponenten. Eine Auflösung dieser Einschränkung kann grundsätzlich eine fundamentale Änderung der vorgestellten Ergebnisse bedeuten. Jedoch lassen sich auch erste Argumente anführen, dass insbesondere die im vorherigen Abschnitt deduzierte Hypothese auch bei einer entsprechenden Modellerweiterung bestehen bleiben könnte. Einerseits ist die obige Hypothese unabhängig von den Parametern e und ε. Solange also eine beliebig kleine Anzahl an Fehlern dem Effekt der Ausbreitung auf angrenzende Komponenten unterliegt, besitzt die Hypothese Gültigkeit. Eine zusätzliche Berücksichtigung von Fehlern ohne Auswirkungen auf angrenzende Komponenten dürfte das Ergebnis daher sogar noch verschärfen. Andererseits würde jedoch eine zusätzliche Betrachtung der Dokumentation von unabhängigen Code-Fragmenten vermutlich auch höhere Auszahlungen für die Erstellung einer technischen Programmdokumentation bedeuten. Da aber selbst in diesem Fall eine Funktion dritten oder höheren Grades für die Dokumentationsauszahlungen unrealistisch erscheint und die obige Hypothese auch für eine quadratische Auszahlungsfunktion deduzierbar ist, dürfte auch eine entsprechende Modellerweiterung zu einem vergleichbaren Ergebnis führen. Dennoch sei darauf hingewiesen, dass zukünftige Forschungsarbeiten diese Vermutung erst noch belegen müssen.

Weiterhin beschränkt sich das Modell neben der Anzahl der Abhängigkeiten einer Komponente nur auf die technische Programmdokumentation einer Software als Einflussgröße auf die Fehleranfälligkeit von nachträglichen Modifikationen. In der Realität dürften jedoch wesentlich mehr Faktoren wie zum Beispiel die Qualität des Quellcodes oder die Erfahrung des Wartungspersonals einen Effekt auf die Entstehung von Fehlern während der Softwarewartung ausüben. Die vorgestellten Ergebnisse besitzen demnach einzig im Rahmen dieser Einschränkung Gültigkeit. Für die Berücksichtigung weiterer Faktoren auf die Fehleranfälligkeit von Wartungsmaßnahmen ist eine entsprechende Erweiterung des vorgestellten Modells erforderlich.

Zudem wird vereinfachend angenommen, dass jede Komponente gleich viele angrenzende Komponenten besitzt, die Auszahlungen für die Dokumentation einer Abhängigkeit stets gleich hoch sind, sich die dokumentierten Abhängigkeiten gleichmäßig auf alle

Komponenten verteilen, in jeder Periode ein jeweils gleicher Anteil der existierenden Komponenten modifiziert wird, die Fehlerwahrscheinlichkeiten von nachträglichen Änderungen bei dokumentierten und bei nicht dokumentierten Abhängigkeiten jeweils gleich hoch sind sowie die Korrektur eines Fehlers eine Auszahlung in stets gleicher Höhe bedeutet. Wie bereits diskutiert werden diese Annahmen in der Realität nur sehr bedingt zutreffen. Daher müssen für diese Parameter entsprechende Durchschnittswerte herangezogen werden. Zeichnet sich ein Softwaresystem dennoch durch eine starke Heterogenität bezüglich dieser Parameter aus, ist eine separate Anwendung des Modells auf einzelne Teile der Software sinnvoll, für die ein ausreichendes Maß an Homogenität angenommen werden kann. An dieser Stelle sei jedoch nochmals betont, dass die konkreten Werte dieser Parameter auf die im vorherigen Abschnitt abgeleitete Hypothese, nämlich dass eine positive Abweichung vom optimalen Dokumentationsgrad im Vergleich zu einer gleich hohen negativen Abweichung stets vorteilhaft ist, keinen Einfluss besitzen. Aufgrund dieser Homogenisierung kann das vorgestellte Modell allerdings keinen Hinweis liefern, welche konkreten Abhängigkeiten von SW bevorzugt dokumentiert werden sollten. Für die Generierung derartiger Hypothesen wäre eine Erweiterung des Modells notwendig, welche die Homogenisierungsannahmen auflöst.

Mit den Auszahlungen für die Erstellung einer technischen Programmdokumentation und für die Fehlerbehebung berücksichtigt das Modell nur einen Teil der zu einem Software-Projekt gehörigen Zahlungsströme. Folglich eignet sich das Modell nicht für eine Beurteilung der absoluten Vorteilhaftigkeit der Erstellung einer technischen Programmdokumentation in Softwareprojekten. Diese muss unter Einbezug aller relevanten Ein- und Auszahlungen beispielsweise mittels der Kapitalwertmethode oder der kapitalwertigen Risikoanalyse (Bamberg et al. 2006) erfolgen.

Ferner bezieht sich die existierende Literatur, auf die sich ein Großteil der obigen Modellannahmen stützt, überwiegend auf objektorientierte Softwaresysteme. Folglich beschränkt sich die Aussagekraft der vorgestellten Ergebnisse nur auf derartige Software. Für eine Prüfung, inwieweit das Modell auch außerhalb dieser Einschränkung Gültigkeit besitzt, müssen die den Annahmen zugrunde liegenden Zusammenhänge ebenfalls im Kontext anderer Programmierparadigmen empirisch geprüft werden.

1.8 Theoretischer und praktischer Beitrag

Aus einer wissenschaftlichen Perspektive wurden die Auswirkungen der technischen Programmdokumentation eines Softwaresystems auf die Fehleranfälligkeit von nachträglichen Modifikationen im Zuge der Wartung und die damit verbundenen ökonomi-

schen Fragestellungen stark vernachlässigt. Zwar bestätigen einige empirische Studien einen reduzierenden Effekt der technischen Programmdokumentation auf die Entstehung von Fehlern im Rahmen der Softwarewartung (Arisholm et al. 2006, S. 365, Dzidek et al. 2008, S. 424, Prechelt et al. 2002, S. 595). Dieser Zusammenhang genügt jedoch nicht als Grundlage für die Lösung der damit verbundenen, praxisrelevanten Entscheidungsprobleme beispielsweise hinsichtlich des ökonomisch sinnvollen Investitionsvolumens in eine technische Programmdokumentation.

Darüber hinaus beziehen sich die existierenden Ansätze zur Prognose von Fehlerhäufigkeiten oder Versagensfällen weitestgehend auf die Entwicklungsphase oder auf den Zeitpunkt der Inbetriebnahme einer Software (Grottke 2005). Die Entstehung von Fehlern aufgrund nachträglicher Modifikationen während der Wartungsphase wird überwiegend vernachlässigt. Außerdem berücksichtigt ein Großteil dieser Ansätze nicht die Verfügbarkeit einer technischen Programmdokumentation als Einflussfaktor auf die Entstehung und Ausbreitung von Fehlern. Eine Ausnahme bildet beispielsweise der Beitrag von Takahashi und Kamayachi (1989), in dem die Autoren einen signifikanten Zusammenhang zwischen dem Umfang der Designdokumentation und der Anzahl der Fehler zeigen, welche zu Beginn der Testphase während der Entwicklung in der Software verbleiben. Jedoch liegt der Fokus dabei wiederum nicht auf der Entstehung und Ausbreitung von Fehlern während der Softwarewartung. Als Konsequenz eignen sich auch diese Ansätze nicht für eine befriedigende Beantwortung der oben erwähnten Fragestellung, inwieweit eine Investition in die Erstellung einer technischen Programmdokumentation während der Softwareentwicklung zur Reduktion der Fehleranfälligkeit von nachgelagerten Wartungsmaßnahmen ökonomisch sinnvoll ist.

Vor diesem Hintergrund präsentiert der vorliegende Beitrag einen neuartigen Ansatz, welcher die barwertigen Auszahlungen für die Erstellung einer technischen Programmdokumentation während der Softwareentwicklung und für die Behebung der während der Softwarewartung aufgrund von nachträglichen Modifikationen entstehenden Fehler gegenüberstellt. Dazu wird mittels Deduktion ein neuer, über die bisher bekannten Zusammenhänge hinausgehender Zusammenhang zwischen dem Dokumentationsgrad einer Software und der Anzahl der im Zuge der Softwarewartung entstehenden Fehler abgeleitet. Diese Hypothese kombiniert dabei insbesondere die in der bestehenden Literatur bereits empirisch bestätigten Zusammenhänge zwischen der Anzahl der Fehler aufgrund nachträglicher Änderungen und der Verfügbarkeit einer adäquaten technischen Programmdokumentation (Arisholm et al. 2006, S. 365, Dzidek et al. 2008, S. 424, Prechelt et al. 2002, S. 595) sowie zwischen der Fehleranfälligkeit einer Soft-

warekomponente und der Anzahl ihrer Abhängigkeiten zu angrenzenden Komponenten (Briand et al. 2002, S. 714, Gyimóthy et al. 2005, S. 909, Singh et al. 2010, S. 31). Dabei wird auch dem Umstand Rechnung getragen, dass die technische Programmdokumentation beispielsweise abhängig von ihrer Qualität einen unterschiedlichen Mehrwert für das Verständnis des zu modifizierenden Codes durch das Wartungspersonal bieten und damit auch die Fehlerwahrscheinlichkeit von nachträglichen Änderungen in unterschiedlichem Maße reduzieren kann. Darauf aufbauend werden die barwertigen Auszahlungen für die Dokumentation während der Softwareentwicklung und für die Fehlerbehebung während der nachgelagerten Wartung modelliert sowie der auszahlungsminimale Dokumentationsgrad berechnet. Insbesondere ein solcher Zusammenhang kann nun eine geeignete Grundlage für die Lösung der oben erläuterten Problemstellung bilden. Ferner ermöglicht die Analyse der Auswirkungen einer Abweichung vom optimalen Dokumentationsgrad die Generierung einer weiteren Hypothese: Eine positive Abweichung vom optimalen Dokumentationsgrad ist stets vorteilhaft gegenüber einer betragsmäßig gleichen, negativen Abweichung. Diese Aussage ist dabei unter den Einschränkungen der getroffenen Annahmen für jeden zulässigen Wert aller unabhängigen Parameter gültig. Gerade eine solche Hypothese lässt sich vermutlich nicht unmittelbar aus Beobachtungen in der Realität induzieren, sondern nur durch einen deduktiven Ansatz, wie er im vorliegenden Beitrag gewählt wurde, ableiten.

Eine Grundlage des Modells bilden aber auch einige vereinfachende und verallgemeinernde Annahmen. Daher möchten das vorgestellte Modell und deren Ergebnisse vorrangig als Grundlage für weitere Forschungsaktivitäten verstanden werden. Eine unmittelbare Ableitung von Handlungsempfehlungen für die Praxis beispielsweise hinsichtlich eines ökonomisch sinnvollen Investitionsvolumens in die Erstellung einer adäquaten technischen Programmdokumentation zur Reduzierung der Fehleranfälligkeit von Wartungsaktivitäten erscheint alleine auf der Basis der vorgestellten Modellergebnisse noch nicht sinnvoll. Zuvor dürften sich entsprechende Modellerweiterungen im Rahmen zukünftiger Forschungsarbeiten lohnen, welche insbesondere die gegenüber der Realität vereinfachenden Annahmen auflösen und weitere Zusammenhänge aufdecken. Ferner muss eine kritische Prüfung der abgeleiteten Hypothesen erfolgen. Erst bei einer empirisch hinreichend starken Bestätigung ist eine fundierte Grundlage für die Generierung normativer Aussagen gegeben.

Sofern jedoch insbesondere die im sechsten Abschnitt vorgestellte Hypothese auch nach der Auflösung einiger vereinfachenden Annahmen bestätigt werden kann, müsste das aktuelle Vorgehen in der Praxis kritisch hinterfragt werden. Aktuell scheint in Soft-

wareprojekten die Ansicht vertreten zu sein, angesichts eines hohen Zeit- und Kostendrucks auf eine adäquate Dokumentation verzichten zu können. Der Fokus scheint überwiegend auf der Erreichung des nächsten Meilensteins zu liegen und die langfristige Lebenszyklusbetrachtung wird vernachlässigt (Zarnekow et al. 2004, S. 181). Sowohl Entwickler als auch die zuständigen Führungskräfte nehmen offenbar die Dokumentation weitgehend als unattraktive Maßnahme wahr, da diese das Erreichen des nächsten Meilensteins nicht beschleunigt, sondern eher verzögert (Parnas 1994, S. 283). Die abgeleiteten Hypothesen stellen an dieser Stelle zumindest ein erstes Indiz dar, dass dieses Vorgehen insbesondere in Bezug auf die technische Programmdokumentation ökonomisch nicht sinnvoll sein könnte. Jedoch stellen Softwareentwickler oftmals eine „knappe Ressource“ dar und mit der Erstellung einer umfassenderen technischen Programmdokumentation sind diese entsprechend länger an ein Softwareprojekt gebunden. Daher stellt sich die Frage nach einem ökonomisch sinnvollen Investitionsvolumen in eine adäquate technische Programmdokumentation nur im ersten Schritt für ein isoliert betrachtetes Projekt und ist darüber hinaus auch in der Frage nach dem ökonomisch sinnvollen Zeitpunkt für den Beginn von Folgeprojekten eingebettet. Dieser Umstand könnte eine Ursache dafür sein, dass der bereits bekannte und empirisch bestätigte Zusammenhang zwischen der Verfügbarkeit einer technischen Programmdokumentation und der Fehleranfälligkeit von Wartungsmaßnahmen offenbar keine Auswirkungen auf das bisherige Dokumentationsverhalten in der Praxis hatte.

1.9 Zusammenfassung und Ausblick

Wie bereits erläutert ergibt sich für jedes Softwareprojekt die Frage, inwieweit die Erstellung einer technischen Programmdokumentation während der Entwicklung zur Reduzierung der Fehleranfälligkeit von nachgelagerten Wartungsmaßnahmen ökonomisch sinnvoll ist. Die existierende Literatur bietet jedoch mitnichten eine ausreichende Grundlage für eine befriedigende Beantwortung dieser Fragestellung. Vor diesem Hintergrund stellt der vorliegende Beitrag einen neuartigen Ansatz vor, welcher, ergänzend zur bisherigen Literatur, im Sinne einer wissenschaftlichen Erklärung neue Zusammenhänge beispielsweise zwischen dem Dokumentationsgrad einer Software und der Anzahl der im Zuge der Softwarewartung entstehenden Fehler deduziert. Da als Grundlage jedoch einige vereinfachende und verallgemeinernde Annahmen getroffen werden, möchten der vorgestellte Ansatz und deren Ergebnisse weniger als unmittelbar in der Praxis anwendbares Modell, sondern vielmehr komplementär zu den bisherigen Beiträgen beispielsweise zur Prognose von Fehlerhäufigkeiten und Softwarezuverlässigkeit

als Grundlage für weitere Forschungsaktivitäten verstanden werden.

Zukünftige empirische Forschungsarbeiten müssen nun die generierten Hypothesen einer kritischen Prüfung unterziehen, inwieweit sie die entsprechenden Phänomene in der Realität prognostizieren können. Aufgrund der Wahl eines formalen Ansatzes liegen diese in einer insbesondere für eine derartige Prüfung geeigneten Form vor (Meredith et al. 1989, S. 308). Darüber hinaus sollten zukünftige Forschungsbeiträge beispielsweise mithilfe entsprechender Modellerweiterungen die Auflösung der gegenüber der Realität vereinfachenden Annahmen sowie die Deduktion zusätzlicher Hypothesen verfolgen. Diese können dann wiederum als Grundlage für die weitere empirische Forschung in diesem Bereich sowie für die Lösung der relevanten Entscheidungsprobleme dienen. Im Konkreten könnten zukünftige Modellerweiterungen eine zusätzliche Betrachtung von Fehlern ohne Auswirkungen auf angrenzende Komponenten sowie die Berücksichtigung von Erweiterungen einer Software im Rahmen der Wartung zum Gegenstand haben. Auch der Einbezug weiterer Faktoren auf die Fehleranfälligkeit von nachträglichen Modifikationen wie zum Beispiel die Code-Qualität oder die Erfahrung des Wartungspersonals erscheint lohnenswert. Derartige Beiträge könnten analysieren, ob und inwieweit insbesondere die im Abschnitt 6 dargelegte Hypothese auch nach Auflösung der entsprechenden Annahmen deduzierbar ist. Für die Generierung weiterer Hypothesen in Bezug auf die Frage, welche Softwarekomponenten und Abhängigkeiten bevorzugt dokumentiert werden sollten, könnten sich zukünftige Forschungsarbeiten auf eine Auflösung der Homogenisierungsannahmen konzentrieren. Darüber hinaus kann das vorgestellte Modell die Grundlage für eine kapitalwertige Risikoanalyse bilden, welche alle relevanten Ein- und Auszahlungen berücksichtigt und die absolute Vorteilhaftigkeit der technischen Programmdokumentation einer Software untersucht. Auch eine Anpassung des Modells auf die Spezifika iterativer beziehungsweise inkrementeller Vorgehensmodelle bei der Softwareentwicklung und auf Situationen, in denen die Entwicklung und Wartung eines Softwaresystems von unterschiedlichen Parteien durchgeführt werden, könnte zu relevanten Erkenntnissen führen.

Zugleich dürften die Zusammenhänge zwischen der technischen Programmdokumentation und der Softwarewartung auch auf weitere praxisrelevante Entscheidungssituationen Einfluss besitzen: Inwieweit ist die Wartung eines Softwaresystems insgesamt ökonomisch sinnvoll? Unter welchen Bedingungen kann sich das Outsourcing der Softwarewartung lohnen? Wann sind welche Maßnahmen des Re-Engineerings sinnvoll und zu welchem Zeitpunkt sollte ein Altsystem durch eine neue Software ersetzt werden? Für die vollständige und befriedigende Beantwortung dieser und vieler weiterer

Fragestellungen sind die vorgestellten Ergebnisse jedoch mitnichten ausreichend. Vielmehr ist die Aufdeckung weiterer Zusammenhänge im Rahmen zukünftiger Forschungsarbeiten erforderlich, für welche der vorliegende Beitrag eine Grundlage bilden kann.

Anhang

In Abschnitt 6 wird der optimale Dokumentationsgrad r_d^* bestimmt, welcher die barwertigen Auszahlungen für die Dokumentation und für die Fehlerbehebung minimiert. Grundlage dafür bildet die im Folgenden dargestellte Berechnung des Minimums der Auszahlungsfunktion

$$CF(r_d) = m \cdot a \cdot r_d \cdot c_d + (c_b + c_s \cdot a) \cdot r_c \cdot m \cdot \frac{a \cdot (e - r_d \cdot \varepsilon)}{\left(1 - a \cdot (e - r_d \cdot \varepsilon)\right)} \cdot \beta$$

mit

$$\mathbb{D} = \left\{ r_d \in \mathbb{R} | 0 \leq r_d \leq 1 \wedge a \cdot (e - \varepsilon) < 1 \wedge r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} \right\}$$

Durch das Gleichsetzen der ersten Ableitung

$$\frac{dCF(r_d)}{dr_d} = m \cdot a \cdot \left(c_d - \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - r_d \cdot \varepsilon)\right)^2} \right)$$

mit 0 und dem Auflösen dieser Gleichung nach r_d ergeben sich zwei Kandidaten für Extremwerte:

$$\frac{dCF(r_d)}{dr_d} \stackrel{\text{def}}{=} 0$$

$$\Rightarrow r_{d,1} = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} + \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon}$$

$$r_{d,2} = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} - \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon}$$

Es ist leicht ersichtlich, dass $r_{d,2}$ für alle zulässigen Werte der gegebenen Parameter außerhalb des Definitionsbereichs $\mathbb{D}$ liegt:

$$r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} > r_{d,2}$$

Dagegen gilt für $r_{d,1}$:

$$r_d > \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} < r_{d,1}$$

Damit liegt $r_{d,1}$ unter folgenden Bedingungen innerhalb des Definitionsbereichs $\mathbb{D}$:

(B.1) $$c_d \geq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - \varepsilon)\right)^2} \quad \Leftrightarrow \quad r_{d,1} \leq 1$$

(B.2) $$(a \cdot e \geq 1) \quad \vee \quad \left(a \cdot e < 1 \quad \wedge \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}\right) \quad \Leftrightarrow \quad r_{d,1} \geq 0$$

Unter den Bedingungen (B.1) und (B.2) bildet $r_{d,1}$ folglich den einzigen kritischen Punkt von $CF(r_d)$. Da zudem die zweite Ableitung

$$\frac{d^2 CF(r_d)}{dr_d{}^2} = \frac{2 \cdot m \cdot a^2 \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon^2 \cdot \beta}{\left(1 - a \cdot (e - r_d \cdot \varepsilon)\right)^3}$$

an diesem Punkt größer ist als 0, besitzt die Auszahlungsfunktion $CF(r_d)$ bei $r_{d,1}$ ein globales Minimum:

$$\frac{d^2 CF\left(r_{d,1}\right)}{dr_{d,1}{}^2} = \frac{2 \cdot m \cdot a^2 \cdot c_d{}^2 \cdot \varepsilon}{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}} > 0$$

Abschließend müssen noch (B.1) und (B.2) bewiesen werden:

Beweis zu (B.1):

$$r_{d,1} = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} + \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon} \leq 1$$

$$\Leftrightarrow \quad \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon} \leq 1 - \frac{e}{\varepsilon} + \frac{1}{a \cdot \varepsilon}$$

$$\Leftrightarrow \quad \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon} \leq \frac{1 - a \cdot (e - \varepsilon)}{a \cdot \varepsilon}$$

$$\Leftrightarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} \leq 1 - a \cdot (e - \varepsilon) \quad \text{mit} \quad a \cdot (e - \varepsilon) < 1 \text{ und } \sqrt{c_d} > 0$$

$$\Leftrightarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{1 - a \cdot (e - \varepsilon)} \leq \sqrt{c_d}$$

mit $\sqrt{c_d} > 0$ und $a \cdot (e - \varepsilon) < 1$ und $\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta} > 0$

$$\Leftrightarrow \quad \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{\left(1 - a \cdot (e - \varepsilon)\right)^2} \leq c_d$$

q.e.d.

Beweis zu (B.2):

$$r_{d,1} = \frac{e}{\varepsilon} - \frac{1}{a \cdot \varepsilon} + \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon} \geq 0$$

$$\Leftrightarrow \quad \frac{\sqrt{c_d \cdot (c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{c_d \cdot a \cdot \varepsilon} \geq \frac{1 - a \cdot e}{a \cdot \varepsilon}$$

$$\Leftrightarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} \geq 1 - a \cdot e$$

Für $a \cdot e \geq 1$ gilt:

$$\frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} > 0 \geq 1 - a \cdot e$$

Für $a \cdot e < 1$ ist die Ungleichung erfüllt g.d.w.

$$\frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} \geq 1 - a \cdot e \quad \text{mit} \quad 1 - a \cdot e > 0 \quad \text{und} \quad \sqrt{c_d} > 0$$

$$\Leftrightarrow \quad \sqrt{c_d} \leq \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{1 - a \cdot e}$$

$$\text{mit} \quad \sqrt{c_d} > 0 \text{ und } \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{1 - a \cdot e} > 0$$

$$\Leftrightarrow \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}$$

Daraus folgt:

$$(a \cdot e \geq 1) \quad \vee \quad \left(a \cdot e < 1 \quad \wedge \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}\right)$$

$$\Rightarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} \geq 1 - a \cdot e \quad \Leftrightarrow \quad r_{d,1} \geq 0$$

Darüber hinaus gilt:

$$(a \cdot e < 1) \quad \wedge \quad \left(a \cdot e \geq 1 \quad \vee \quad c_d > \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}\right)$$

$$\Rightarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} < 1 - a \cdot e \quad \Leftrightarrow \quad r_{d,1} < 0$$

Daraus folgt wiederum:

$$(a \cdot e \geq 1) \quad \vee \quad \left(a \cdot e < 1 \quad \wedge \quad c_d \leq \frac{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}{(1 - a \cdot e)^2}\right)$$

$$\Leftarrow \quad \frac{\sqrt{(c_b + c_s \cdot a) \cdot r_c \cdot \varepsilon \cdot \beta}}{\sqrt{c_d}} \geq 1 - a \cdot e \quad \Leftrightarrow \quad r_{d,1} \geq 0$$

q.e.d.

Literatur (Kapitel III.1)

Abdelmoez W, Nassar DM, Shereshevsky M, Gradetsky N, Gunnalan R, Ammar HH, Yu B, Mili A (2004) Error Propagation in Software Architectures. In: Proceedings of the International Symposium on Software Metrics (METRICS '04). IEEE Computer Society, 384-393

Abdelmoez W, Shereshevsky M, Gunnalan R, Ammar HH, Yu B, Bogazzi S, Korkmaz M, Mili A (2005) Quantifying Software Architectures: An Analysis of Change Propagation Probabilities. In: Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications (AICCSA '05). IEEE, 124-vii

Aggarwal KK, Singh Y, Kaur A, Malhotra R (2009) Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study. Software Process: Improvement and Practice 14(1):39-62

Arisholm E, Briand LC, Hove SE, Labiche Y (2006) The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. IEEE Transactions on Software Engineering 32(6):365-381

Bamberg G, Dorfleitner G, Krapp M (2006) Unternehmensbewertung unter Unsicherheit: Zur entscheidungstheoretischen Fundierung der Risikoanalyse. ZfB 76(3):287-307

Beszédes Á, Gergely T, Jász J, Tóth G, Gyimóthy T, Rajlich V (2007) Computation of Static Execute After Relation with Applications to Software Maintenance. In: Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM '07). IEEE, 295-304

Binkley AB, Schach SR (1998) Validation of the Coupling Dependency Metric as a Predictor of Run-Time Failures and Maintenance Measures. In: Proceedings of the 20th International Conference on Software Engineering (ICSE '98). IEEE Computer Society, 452-455

Briand LC, Melo WL, Wüst J (2002) Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects. IEEE Transactions on Software Engineering 28(7):706-720

de Souza SCB, Anquetil N, de Oliveira KM (2005) A Study of the Documentation Essential to Software Maintenance. In: Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information. ACM, 68-75

Dzidek WJ, Arisholm E, Briand LC (2008) A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. IEEE Transactions on Software Engineering 34(3):407-432

Eaddy M, Zimmermann T, Sherwood KD, Garg V, Murphy GC, Nagappan N, Aho AV (2008) Do Crosscutting Concerns Cause Defects? IEEE Transactions on Software Engineering 34(4):497-515

Fenton NE, Ohlsson N (2000) Quantitative Analysis of Faults and Failures in a Complex Software System. IEEE Transactions on Software Engineering 26(8):797-814

Grottke M (2005) Prognose von Softwarezuverlässigkeit, Softwareversagensfällen und Softwarefehlern. In: Mertens P., Rässler S. (eds) Prognoserechnung. 6. Auflage, Physica, Heidelberg, S 459-487

Gyimóthy T, Ferenc R, Siket I (2005) Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Transactions on Software Engineering 31(10):897-910

Hanssen GK, Yamashita AF, Conradi R, Moonen L (2009) Maintenance and Agile Development: Challenges, Opportunities and Future Directions. In: Proceedings of the 25th IEEE International Conference on Software Maintenance (ICSM '09). IEEE, 487-490

Hruschka P (2003) Agility. Informatik-Spektrum 26(6):397-401

Ko AJ, Myers BA, Coblenz MJ, Aung HH (2006) An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Task. IEEE Transactions on Software Engineering 32(12):971-987

Krishnan MS, Mukhopadhyay T, Kriebel CH (2004) A Decision Model for Software Maintenance. Information Systems Research 15(4):396-412

Kruschwitz L (2006) Finanzmathematik: Lehrbuch der Zins-, Renten-, Tilgungs-, Kurs- und Renditerechnung. 4. Auflage, Vahlen, München

Lehman MM (1996) Laws of Software Evolution Revisited. In: Springer. (eds) Proceedings Lecture Notes in Computer Science 1149. Springer, 108-124

MacCormack A, Rusnak J, Baldwin CY (2006) Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. Management Science 52(7):1015-1030

Meredith JR, Raturi A, Amoako-Gyampah K, Kaplan B (1989) Alternative Research Paradigms in Operations. Journal of Operations Management 8(4):297-326

Olague HM, Etzkorn LH, Gholston S, Quattlebaum S (2007) Empirical Validation of Three Software Metrics Suites to Predict Fault-Proneness of Object-Oriented Classes Developed Using Highly Iterative or Agile Software Development Processes. IEEE Transactions on Software Engineering 33(6):402-419

Parnas DL (1994) Software Aging. In: Proceedings of the 16th International Conference on Software Engineering (ICSE '94). IEEE Computer Society Press, Los Alamitos, USA, 279-287

Prechelt L, Unger-Lamprecht B, Philippsen M, Tichy WF (2002) Two Controlled Experiments Assessing the Usefulness of Design Pattern Documentation in Program Maintenance. IEEE Transactions on Software Engineering 28(6):595-606

Radatz J (1990) IEEE Standard Glossary of Software Engineering Terminology. Std 610.12-1990, Standards Coordinating Committee of the Computer Society of the IEEE

Rajlich V (2009) Intensions are a Key to Program Comprehension. In: IEEE 17th International Conference on Program Comprehension (ICPC '09). IEEE, 1-9

Rajlich V (2006) Changing the Paradigm of Software Engineering. Communications of the ACM 49(8):67-70

Riedl R, Kepler J (2003) Begriffliche Grundlagen des Business Process Outsourcing. Information Management & Consulting 18(3):6-11

Robillard MP, Coelho W, Murphy GC (2004) How Effective Developers Investigate Source Code: An Exploratory Study. IEEE Transactions on Software Engineering 30(12):889-903

Rostkowycz AJ, Rajlich V, Marcus A (2004) A Case Study on the Long-Term Effects of Software Redocumentation. In: Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM '04). IEEE, 92-101

Sharif KY, Buckley J (2008) Developing Schema for Open Source Programmers' Information-Seeking. In: 3rd International Symposium on Information Technology (ITSIM '08). IEEE, 1-9

Singh Y, Kaur A, Malhotra R (2010) Empirical Validation of Object-Oriented Metrics for Predicting Fault Proneness Models. Software Quality Journal 18(1):3-35

Sneed HM (2008) Offering Software Maintenance as an Offshore Service. In: IEEE International Conference on Software Maintenance (ICSM '08). IEEE, 1-5

Statista (2010a) Weltweite Ausgaben von Wirtschaft und öffentlicher Verwaltung für IT-Outsourcing im Jahr 2010 in Milliarden US Dollar. http://de.statista.com/statistik/daten/studie/163365/umfrage/weltweite-ausgaben-fuer-it-outsourcing-in-2010/, Abruf am 18.02.2011

Statista (2010b) Weltweite IT-Ausgaben von Wirtschaft und öffentlicher Verwaltung für ITK Produkte und Services im Jahr 2010 in Milliarden US-Dollar. http://de.statista.com/statistik/daten/studie/163257/umfrage/weltweite-ausgaben-fuer-itk-produkte-und-services-in-2010/, Abruf am 18.02.2011

Subramanyam R, Krishnan MS (2003) Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects. IEEE Transactions on Software Engineering 29(4):297-310

Takahashi M, Kamayachi Y (1989) An Empirical Study of a Model for Program Error Prediction. IEEE Transactions on Software Engineering 15(1):82-86

Tan Y, Mookerjee VS (2005) Comparing Uniform and Flexible Policies for Software Maintenance and Replacement. IEEE Transactions on Software Engineering 31(3): 238-255

Van Vliet H (2008) Software Engineering: Principles and Practices. 3. Auflage, John Wiley & Sons, West Sussex, England

Yu P, Systa T, Müller H (2002) Predicting Fault-Proneness using OO Metrics: An Industrial Case Study. In: Proceedings of the 6th European Conference on Software Maintenance and Reengineering (CSMR '02). IEEE Computer Society, Los Alamitos, USA, 99-107

Zarnekow R, Scheeg J, Brenner W (2004) Untersuchung der Lebenszykluskosten von IT-Anwendungen. Wirtschaftsinformatik 46(3):181-187

Zimmermann T, Nagappan N (2008) Predicting Defects Using Network Analysis on Dependency Graphs. In: Proceedings of the 30th International Conference on Software Engineering (ICSE '08). ACM, 531-540

2 Beitrag: „*Monetary Valuation of Alternative Software Structures Considering both Technical and Factual Dependencies – A Formal Deductive Approach*“

Autoren:	Peter Bartmann, Prof. Dr. Marco C. Meier, Martina Lindermeir Kernkompetenzzentrum Finanz- & Informationsmanagement, Lehrstuhl für BWL, Wirtschaftsinformatik, Informations- & Finanzmanagement (Prof. Dr. Hans Ulrich Buhl) Universität Augsburg, D-86135 Augsburg peter.bartmann@wiwi.uni-ausburg.de, marco.meier@wiwi.uni-augsburg.de, martina.lindermeir@wiwi.uni-augsburg.de

Abstract:

Companies spend up to 80% of software life cycle costs on maintenance. The predominant approach to reduce change propagation and the consequential costs for maintenance is the development of software architectures with low coupling between modules. As a great deal of all software modifications is made up of factually driven adaptations, factual dependencies between modules obviously have to be taken into account as well. But the existing literature barely considers factual dependencies in the same way as technical dependencies. Thus, there seems to be a lack of appropriate methods for an economic valuation of alternative modular architectures considering maintenance effort. Hence, this paper proposes a formal deductive approach to infer new hypotheses in terms of cause-effect relationships between different modular architectures and the net present values of corresponding cash outflows for software maintenance in order to make them comparable. In doing so, both technical and factual dependencies are considered. These hypotheses may serve as a basis for further (empirical) research and contribute to supporting design decisions.

2.1 Introduction

For the year 2011 Gartner as one of the world's leading information technology market research companies forecasts worldwide software spending to rise 5.1 percent to 253 billion US dollars, with the majority of enterprise software markets growing (Gartner 2011). Costs for software maintenance attain up to 80% of software life cycle costs (Krishnan et al. 2004, p. 396; Zarnekow et al. 2004, p. 181). This, as well as findings from several other studies like e.g. a survey from December 2009 whereupon the DAX-30-firms spend at least 1 billion Euros annually on software changes (jCOM1 2009), underlines the economic relevance of this topic for software engineering.

Thus, a significant contribution to value-based management is to reduce software maintenance effort as long as the competitive position of a firm is not affected. A fundamental software engineering theory in this field is information hiding by Parnas (1972), according to which the adaptability of software improves if design decisions that are likely to change are decoupled (Sullivan et al. 2001, p. 99). As the results of the literature review will show, the latest research in this domain concentrates on technical dependencies of software elements as decision criteria for alternative modular software structures.

However, a great deal of all software modifications is made up of factually driven adaptations, meaning adaptations due to changes of e.g. business processes, market requirements, user requests, or legal regulations. For example, Krishnan et al. (2004, p. 396) state that "*in the course of time, user requests for various system enhancements may dominate software maintenance*". Although the importance of factual dependencies of software elements seems to be obvious for software design and development there are only few contributions that consider factual dependencies of software elements in the same way as technical dependencies. In the rare cases where factual dependencies are considered, e.g. at Friedrichsen (2009, p. 45), an economic valuation is missing.

This motivates the basic question of how different modular software structures should be economically evaluated in order to reduce the effort for software maintenance while considering both technical and factual dependencies.

In order to state more precisely the scope of this paper, first the subject of investigation will be outlined before formal requirements for a scientific contribution in this field are elaborated - which in turn will be the criteria for the evaluation of the results.

2.1.1 Outline of Research Scope

This paper's focus is on firms which individually develop application software. During the development phase amongst others, decisions about the software architecture have to be made, which influence the effort for maintenance later. Modularly structured software in particular can reduce maintenance effort by splitting the whole software into smaller units which encapsulate content and reduce diffusion of changes (Marinescu 2002, p. 9). Other decisions that may also have an impact on maintenance, e.g. concerning development environments, technical platforms or (soft) skills, are out of scope of this paper, but might be also a promising subject to further investigation from a techno-economic point of view.

The key issue of decisions regarding the modular structure is how functionalities of the software should be allocated to modules, where "functionality" characterizes the solution for a subtask that the software has to accomplish. The execution of tasks is usually based on the interaction of multiple functionalities. They interact by exchanging data, hereafter labeled "technical dependency". For example, parameters or return values can be exchanged among methods (Zöller-Greer 2010, p. 65). Due to technical dependencies a change affecting one functionality may diffuse to others (Helmke et al. 2007, p. 318; Baldwin and Clark 2000, p. 63). "Modularization" is an opportunity to prevent such diffusion (Parnas 1972, p. 1054; Baldwin and Clark 2000, p. 63). Therefore, functionalities are encapsulated within modules as enclosed units. Consequently, changes can only diffuse to other modules if there is a technical dependency called "interface" between these modules. Thus, a modular structure with fewer interfaces is preferred in software development in theory and practice (Marinescu 2002, p. 13; Yu et al. 2002, p. 3; Henry and Kafura 1981, p. 516, 517; Kemerer 1995, p. 6, 10, 18).

Moreover, there are factual dependencies between functionalities. For example, Baghdadi (2004, p. 586) defines two attributes describing business objects or coordination artifacts as factually dependent "*if they are concerned by the same business event*". Bearing this concept of factual dependencies in mind, as a basis for the following approach two functionalities are defined as factually dependent, if they are based on the same business or legal requirement. Thus, if a requirement changes, all related functionalities have to be adapted with certain probabilities. In business practice, cases exist where functionalities are factually dependent, but do not exchange data, so that they are technically independent. For example, Eaddy et al. (2008, p. 497) state that a crosscutting concern, which emerges if a software requirement's implementation is

scattered across the program, “*is harder to implement and change consistently because multiple – possibly unrelated – locations in the code have to be found and updated simultaneously*”.

The following simplified example indicates that the predominant approach to cluster functionalities according to technical dependencies might be suboptimal referring to the substantial goal to reduce maintenance effort.

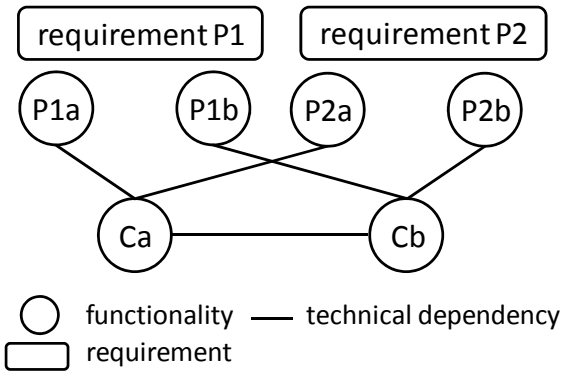

Fig. III-3: Examples of Dependencies

Fig. III-3 illustrates the basic scenario: let P1x be functionalities concerning a product group 1 and P2x functionalities concerning a product group 2 in a way that P1a and P1b are factually dependent as they are based on the same business and legal requirements. In the same way P2a and P2b are factually dependent. Ca and Cb might be functionalities that execute certain calculations in the sales process. There are now various options to allocate these functionalities and their dependencies to modules (cf. Parnas 1972). One specific option will be called “modular structure option (MSO)”. In general, such design decisions about modularization have to be made in advance of software implementation (Parnas 1972, p. 1054). But the dependencies’ allocation in turn affects the effort for changes during software maintenance. In detail, the MSO has an impact on (1) the number of modules in which functionalities that have to be changed together are implemented, and (2) on the number of interfaces between modules and therefore on the diffusion of changes to other modules. According to the chosen MSO technical dependencies within the modules (cohesion) and technical dependencies between the modules (coupling) differ (Pomberger and Pree 2004, pp. 137-138). Considering technical dependencies, the effort caused by the diffusion of changes can

be reduced by selecting a MSO with low coupling.

Therefore, following the predominant approach to cluster functionalities according to technical dependencies in a way that the number of interfaces is minimized may result in the following MSO_1 (see Fig. III-4 left). In this case there is only one interface between two modules. An alternative MSO_2 (see Fig. III-4 right) may cluster functionalities according to factual dependencies. At first glance MSO_2 seems to be much more complex than MSO_1 as there are more modules and interfaces. Thus, MSO_2 can be assumed to cause higher maintenance effort.

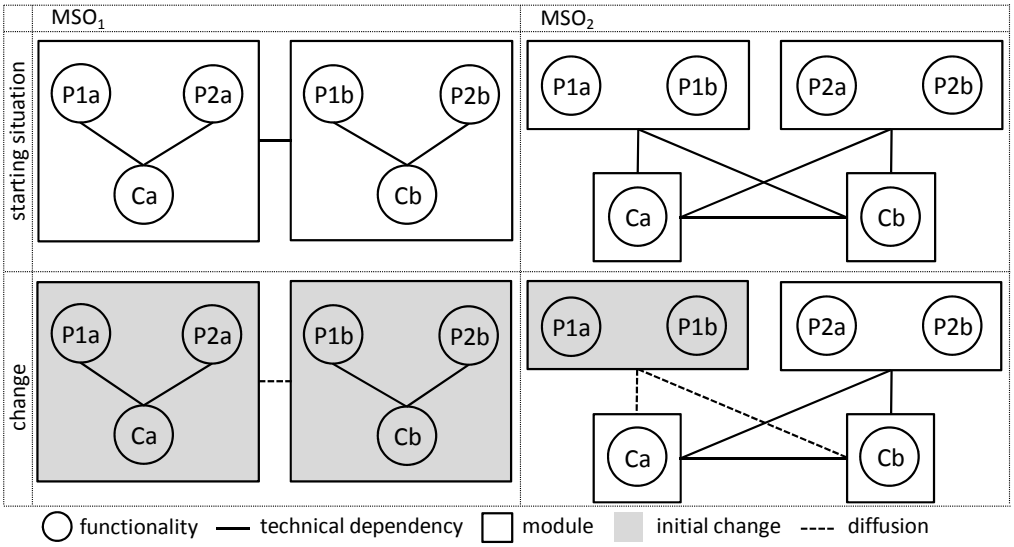

Fig. III-4: Factual Changes of Different MSOs

Nevertheless, such a structure might have advantages compared to MSO_1, if e.g. a legal requirement for product group 1 changes and as a consequence functionalities P1a and P1b must be adapted. In this case, both modules of MSO_1 need to be initially changed and the change diffuses through the interface with a certain probability. In contrast, only one module of MSO_2 needs to be changed initially, but the change diffuses through two interfaces to connected modules with a certain and larger probability as Fig. III-4 illustrates.

This example indicates that there might be cases where the predominant paradigm to cluster functionalities according to technical dependencies may miss the basic goal to reduce maintenance effort. In business practice, factual and technical dependencies

overlap to different extents (cf. Eaddy et al. 2008, p. 497). Moreover, technical dependencies differ in terms of volume and therefore in terms of impact on maintenance effort. A nontrivial question arises about the most beneficial modular structure in order to reduce maintenance effort.

In summary this leads to the main objective of this paper to infer new hypotheses in terms of cause-effect relationships between a MSO and the corresponding net present value of the cash outflows for factually driven changes during maintenance. In doing so, both technical and factual dependencies have to be considered. These hypotheses serve as a basis to make different MSOs comparable in an environment that is affected by uncertainty.

2.1.2 Research Methodology

The constantly recurring research cycle (see Fig. III-5), consisting of description, explanation and testing (through prediction) serves as foundation of research activities in this paper (Meredith et al. 1989, pp. 301-303). Thus, descriptive research precedes explanation and testing and seeks to investigate observable phenomena or to illuminate particular aspects in more detail. On this basis explanatory contributions infer cause-effect relationships, which may explain the described phenomena. Besides the mere explanation, they may also give a recommendation for action. Subsequent testing is done by studies exploring to what extend the explanatory contributions can reliably predict the corresponding phenomena of reality. The results of these studies form the basis for the progressive improvement of the explanatory contributions and of the generated recommendations for action.

The literature overview in the next chapter shows that descriptive research has already been conducted for the stated problem. However, literature is lacking explanatory contributions referring to the main objective formulated at the end of the last section. The existing papers therefore do not provide appropriate approaches to compare different MSOs with respect to the associated maintenance effort considering both technical and factual dependencies. Thus, new hypotheses in terms of cause-effect relationships between a certain MSO and the corresponding net present value of cash outflows for software maintenance have to be inferred. On this basis, further hypotheses whether a certain MSO causes more or less maintenance effort than another MSO can be derived. To reach this objective, a deductive approach is appropriate.

According to Meredith et al. (1989, p. 301 – 304) such hypotheses have to undergo a

critical examination of how reliably they can predict the explained phenomena of reality. Especially formal models are “*often characterized by a significant separation of the phenomenon from the researcher*” and are therefore appropriate for testing (Meredith et al. 1989, p. 308). Thus, the hypotheses will be developed as formal artifact embedded in a quantitatively described model environment. This should in particular serve as an intersubjective comprehensible starting point for testing and improvement. Furthermore, the new hypotheses may serve as basis for further empirical research activities, particularly because these hypotheses can more easily be inferred systematically through a deductive approach as applied in this paper, than solely and directly from observations of reality.

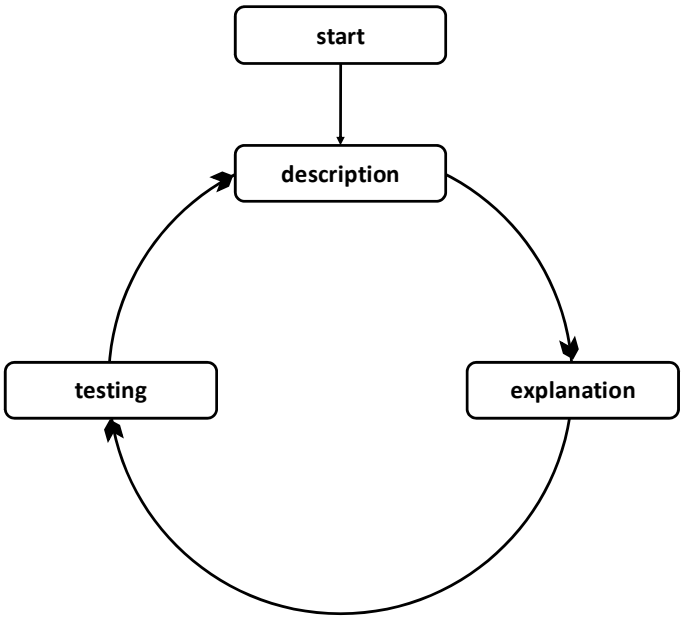

Fig. III-5: Research Cycle (Meredith et al. 1989, p. 302)

2.1.3 General and Specific Requirements of this Scientific Contribution

The model presented in this paper aims to meet some general and specific requirements. First, this scientific contribution has to be intersubjectively comprehensible and issue to validation (R-1). Furthermore, it must provide an innovative add on to the published state of the art (R-2), it must be applicable for a class of problems (R-3) and it must result in benefits for stakeholders today or in the future (R-4). As discussed before, especially a new contribution which provides an economic valuation (R-A) of modular

software architectures (R-B) considering technical (R-C) and factual dependencies (R-D) meets requirement R-2.

2.2 Related Literature Analysis

For literature enquiry, well-established scientific databases (ACM Digital Library, AIS Electronic Library, GoogleScholar, IEEEXplore, ScienceDirect, SpringerLink, Wiley InterScience) were investigated for papers, conference papers, specialist books, thesis and practice reports without timeframe restriction for the specific requirements mentioned above. Tab. III-2 presents a summary of - due to limited space - selected literature that according to the authors' opinion fit best to the research scope. Afterwards, the papers are described more precisely.

Tab. III-2: Literature Overview

category	source	factually driven changes during software maintenance			
		R-A: economic valuation	R-B: modularization	R-C: technical dependencies	R-D: factual dependencies
practical literature	Friedrichsen (2009)	not considered	possible conception	considered	dynamics of change
empirical literature	Kemerer (1995)	not considered	basic concept	empirical coherences to changes	not considered
conceptual literature	Parnas et al. (1985)	not considered	basic concept	segmentation criterion	partly possible resemblances
	Kazman et al. (2000)	not considered	possible architecture	as stakeholder interests	not considered
	Baghdadi (2004)	not considered	Partly applicable by the concept of web services	interfaces	considered

analytical literature	Parnas (1972)	not considered	basic concept	structure options	partly possible resemblances
	Sullivan et al. (2001)	real options	basic concept	design structure matrices	partly possible resemblances
	Al-Otaiby et al (2005)	not considered	basic concept	coupling as a metric	Requirement scenarios
	Gamba and Fusari (2009)	real options	basic concept	basis of valuation	not considered

By means of an empirical study Kemerer (1995) shows that high coupling generates more modifications of the software. The more global data was exchanged by interfaces, the more changes could be observed. From a practical point of view this relationship is supported by Friedrichsen (2009), who states that low coupling is advantageous for maintainability. Thus, the influence of technical dependencies on changes and their diffusion is made clear. However, according to Friedrichsen (2009) and Krishnan (2004) a great deal of changes during maintenance is factually driven. Hence it is beneficial as well if changes only affect a small part of the software, which is given by segmentation of factual dependencies. Thus, the trade-off between technical and factual dependencies reveals an important aspect for decisions on an appropriate architecture. Friedrichsen (2009) and Kemerer (1995) emphasize the importance of software design for maintenance and Friedrichsen (2009) in particular points out the enormous potential for savings. However, neither provides an economic valuation.

Using the Architecture Tradeoff Analysis Method (ATAM), Kazman et al. (2000) provide a method for an evaluation of software architectures by stakeholders (e. g. developer, user, client). They are interviewed about different aspects, like various change scenarios and technical dependencies. The subjective opinions of the stakeholders are then the foundation for decisions about software architectures. So the method does not include an economic valuation and moreover, factual dependencies are not considered. But Kazman et al. (2000) promote the importance of software evaluation during development regarding the whole life cycle.

Parnas (1972) provides a different approach for creating a modularized architecture. By

applying the Key Words in Context (KWIC) method he suggests to encapsulate parts of software which are likely to be changed in separate modules. Similar to that, Parnas et al. (1985) suggest a principle called information hiding for module development, whereby encapsulation leads to less data exchange and therefore to less diffusion of changes. Both papers constitute modularization as basic concept for software development and consider dependencies and their effects. Yet, there is a lack of clear differentiation between technical and factual dependencies as well as a lack of consideration of factual triggers for changes. Again, the authors do not provide an approach for economic valuation.

Baghdadi (2004) introduces the concept of factual dependency between attributes and develops a process to generate web services based on that concept. Noteworthy Baghdadi (2004) points out that most of the current approaches focus mainly on technical aspects. But the business perspective is of particular relevance for generating web services as well, because web services should reflect “*the actual and the potential activities of the organization*” (Baghdadi 2004, p. 585). However, the proposed process does not include an economic valuation.

Both Sullivan et al. (2001) and Gamba and Fusari (2009) evaluate modular software structures using real options. Since Sullivan et al. (2001) refer to the work of Parnas (1972) they do not differentiate between technical and factual dependencies either. Gamba and Fusari (2009) compare option values based on modular operators like splitting, substitution, augmentation, exclusion and inversion. The value of a currently implemented module is compared with the value of this module after it is changed by an operator. Although both papers provide an economic valuation, they do not consider dependencies which have their origin in the same factual requirements.

Al-Otaiby et al. (2005) provide a technique using a data mining method to cluster so called requirement scenarios in a way, that the ones within a cluster have a strong functional relationship among each other and a weak relationship to those in other clusters. The technique aims to minimize coupling between different clusters and to maximize cohesion within clusters. These clusters should serve as a basis for design decisions during modular software development. Though, an economic valuation is missing.

The literature overview reveals two aspects: The stated research gap is a relevant topic (see e. g. Friedrichsen (2009) and Kemerer (1995)). However, the investigated literature does not cover an approach which accomplishes all requirements specified above.

Thus, this paper intends to contribute to close the research gap by inferring new hypotheses in terms of cause-effect relationships between a MSO and the corresponding net present value of the cash outflows for factually driven changes of modular software during maintenance considering technical and factual dependencies. As a result, the cash outflows of different MSOs can be compared and may serve as a basis for decisions during software development.

2.3 Valuation of Modular Structure Options

2.3.1 Assumptions

A company develops an application software SW at the point in time $t = 0$, which consists of n functionalities: $SW = \{f_i | i = 1,2,3, \dots n\}$. A functionality f_i describes the solution of a software's subtask, like authentication in online-banking. The functionalities needed to implement SW are known and can be determined by a requirement analysis. All functionalities are partitioned into l modules, where a module MO_r (with $r = 1,2,3, \dots, l \wedge 1 \leq l \leq n$) consists of at least one functionality ($MO_r \neq \{\}$). Hence, each module is a subset of SW ($MO_r \subseteq SW$) and constitutes a closed, functional unit (cf. Balzert 2009, p. 41). Furthermore, each functionality is implemented in SW exactly once.

There are b different alternatives to subdivide all the needed functionalities into modules. One specific alternative is called MSO_v (with $v = 1,2,3, \dots, b$) and constitutes a partition of SW. Depending on the chosen MSO_v the number of required modules can vary. The number of modules of MSO_v are denoted by l_v with $1 \leq l_v \leq n$. MSO_v is characterized by its modules:

$$MSO_v = \left\{ MO_{v,r} \middle| r = 1,2,3, \dots, l_v \ \wedge \ 1 \leq l_v \leq n \right\} \text{ with}$$

$$\bigcup_{r=1}^{l_v} MO_{v,r} = SW \ \wedge \ \forall \, MO_{v,r} \neq \{\} \ \wedge \forall \, MO_{v,r}, MO_{v,s} \ \left(r \neq s \rightarrow MO_{v,r} \cap MO_{v,s} = \{\} \right)$$

As the number of functionalities which need to be developed is always the same, it can be assumed that the cash outflows for development of SW are identical for every MSO_v. However, the number and size of modules can differ per MSO_v. As a consequence, the cash outflows for development could indeed vary. But it is assumed that the majority of these cash outflows are caused by the development of functionalities. Therefore, the monetary difference of the development of different MSOs caused by different numbers of modules should not be significant and are thus assumed away.

(A1) The cash outflows for development of SW are the same for every MSO_v .

Functionalities can exchange zero, one or more data objects between each other (technical dependency). The number of exchanged data objects between two ties f_i and f_j is denoted by $g_{i,j}$ (with $g_{i,j} \in \mathbb{N}$). A functionality does not exchange data objects with itself (i.e. $\forall\, g_{i,j}\,(i = j \rightarrow g_{i,j} = 0)$). A technical dependency between functionalities located in different modules $MO_{v,r}$ and $MO_{v,s}$ ($r \neq s$) results in an interface for data exchange between these two modules. For the sake of simplicity, it is assumed that there exists at most one interface between two certain modules and all data exchanged between functionalities located in two different modules are transferred through the same interface between these two modules.

The more data objects are exchanged between two modules, the higher is the probability of diffusion of changes from one module to the other. This assumption is supported by Briand et al. (1999) who analyze to what extend several coupling metrics can predict the propagation of changes. By means of an empirical study the authors show that the probability of diffusion of changes is significantly related amongst others to an information-flow based coupling metric, which counts how often a class invokes methods of another class and which weights the number of these method invocations by the number of parameters of the invoked methods. A possible explanation of this phenomenon could be the following: If a module $MO_{v,r}$ needs to be changed, a data object processed by $MO_{v,r}$ needs to be changed with certain probability, too. Therefore, the more data objects are processed by $MO_{v,r}$, the higher is the probability of a required adaption of at least one of these data objects in the case that $MO_{v,r}$ is affected by a change. Furthermore, if a change of $MO_{v,r}$ affects a data object, which is exchanged between $MO_{v,r}$ and another module $MO_{v,s}$, $MO_{v,s}$ also needs to be changed with certain probability as this data object is processed by $MO_{v,s}$ as well. Accordingly, the more data objects are exchanged between two modules, the higher is the probability that a change of one module diffuses to the other. In this context the same data object may be transferred more than once between two modules. Such a data object has to be processed repeatedly e.g. at different times or by different functionalities within these two modules. Thus, if a particular data object is transferred more than once between two modules, an adaption of one module is more likely to diffuse to the other module. Therefore, every data object is counted as often as it is transferred. Both the number of interfaces and the number of exchanged data objects per interface are defined by the particular sen MSO_v.

(A2) The number of data objects $h_{v,r,s}$ exchanged through the interface tween $MO_{v,r}$ and $MO_{v,s}$ are equal to the sum of all exchanges of data objects between any functionality in $MO_{v,r}$ and any functionality in $MO_{v,s}$:

$$h_{v,r,s} = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{i,j} \cdot x_i \cdot y_j \quad \text{with} \quad x_i = \begin{cases} 1 & \text{if } f_i \in MO_{v,r} \\ 0 & \text{if } f_i \notin MO_{v,r} \end{cases}, y_j = \begin{cases} 1 & \text{if } f_j \in MO_{v,s} \\ 0 & \text{if } f_j \notin MO_{v,s} \end{cases}$$

Factual requirements define the software's tasks. There are k factual requirements to be fulfilled by SW. One specific factual requirement is called A_a (with $a = 1,2,3,\dots,k$). Every task is mapped at least to one functionality in SW. Factual dependencies emerge by involvement of more than one functionality on the same factual requirement. It is assumed that factual dependencies do not have to coincide with technical ones, that means, factual dependencies between two functionalities may exist without any technical dependencies and vice versa (cf. Eaddy et al. 2008, p. 497).

SW is put into operation at $t = 1$ and is removed from operation at $t = m$. During this operation phase changes of the factual requirements trigger maintenance activities to adapt SW to the changed requirements. To be more concrete, a change of the factual requirement A_a occurs at a certain point in time t_a (with $t_a = 1,2,3,\dots,m$) and with a certain probability p_a (with $0 \leq p_a \leq 1$). Thus, all requirements MA can be characterized by the points in time and the probabilities of later changes: $MA = \{\{p_a, t_a\} | a = 1,2,3,\dots,k\}$. If a requirement A_a changes, the corresponding functionalities need to be changed with conditional probability $q_{a,i}$. All adaptions of functionalities to a changed requirement are carried out at the same point in time t_a at which the corresponding requirement changes. Thus, each requirement A_a is determined by the conditional probabilities of necessary changes of the corresponding functionalities: $A_a = \{q_{a,i} | i = 1,2,3,\dots,n\}$.

It is assumed that all probabilities of future changes are known. Even though this strong assumption does not picture reality, it is not specifically related to this model but to numerous ex ante decision models. Thus this seems to be a simplifying but justified assumption.

The cash outflows CoF for the change of one functionality are assumed to be identical for every functionality and at all points in time. They are determined by average values grounded on experience or cost estimation models like constructive cost model (COCOMO) or function-points (Sommerville 2001, p. 519, 523). In reality some functionalities could be heterogeneous in size and complexity causing different cash outflows for

changes. In that case, the proposed model should be applied separately to parts of SW, for which a certain degree of homogeneity in size and complexity of all functionalities can be assumed.

(A3) The points in time t_a and the probabilities p_a of changes of factual requirements as well as the conditional probabilities $q_{a,i}$ of changes of corresponding functionalities are known. Cash outflows CoF for the change of one functionality are identical for every functionality and occur at the same point in time at which the corresponding requirement changes.

If a functionality needs to be changed, it is assumed that for every functionality within the affected module cash outflows of a certain and identical amount incur. At a first glance this seems to be a quite unrealistic assumption. But for various reasons it can be assumed that a change of a factual requirement, which initially affects only a part of the functionalities of a module, has an effect on the other functionalities of this module as well. Basically, the preceding analysis and comprehension of the existing code is part of all subsequent changes during software maintenance (Robillard et al. 2004, p. 889, Rostkowycz et al. 2004, p. 94). In this context Ko et al. (2006, p. 972) point out that developers spend, on average, 35% of their time solely on navigating between relevant code fragments in order to seek, collect and comprehend information that is relevant to perform a change. As a module is a unit (Pomberger and Pree 2004, p. 137) it can be assumed that every functionality of a module has to be investigated if at least one functionality within this module is affected by a change. Furthermore, in general, every change within a software system has to be tested (Frühauf 2007, p. 34). To be more precise, a module test has to be done if any change has been performed inside a module to ensure that all contained functionalities cooperate accurately (Myers 2004, p. 91). For example, a path coverage test, which is used amongst others for module tests, aims to test each independent execution path of a software component and covers all lines of code (Sommerville 2001, p. 457). Therefore, it is assumed that every functionality within a module affected by a change needs to be tested, regardless of whether or not a certain functionality has actually been modified. Summing up, if at least one functionality needs to be changed, it can be assumed that all functionalities of the affected module at least have to be investigated and tested. Even some more functionalities than those initially affected by a change may need to be changed to ensure accurate viability of the affected module, too. Of course higher cash outflows may incur if a functionality has to be changed and not only be investigated and tested. But as comprehension and testing take a great deal of the time necessary to adapt a functionality and the modification of

the code itself only takes a small part of the required time, incurrence of identical (average) cash outflows for every functionality within an affected module seems to be a simplifying but justified assumption.

(A4) If a module is affected by a change, that means, at least one functionality of this module is affected by the change of a factual requirement, all functionalities of this module need to be investigated. Then a part of these functionalities are modified as needed for adaption of the module. After that, all functionalities of the affected module need to be tested. Since investigation and testing take a great deal of the time necessary to adapt a functionality, incurrence of identical (average) cash outflows for every functionality of a module affected by a change is assumed.

If a module is changed, the change can diffuse to connected modules through the existing interfaces (Helmke et al. 2007, p. 318; Baldwin and Clark 2000, p. 63). The paradigm of information hiding encapsulates module content most widely (Parnas 1972, p. 1056), so that changes diffuse just with a certain probability. This probability $u_{v,r,s}$ is presented by a function $d(h_{v,r,s})$. Thus, coherence between the number of exchanged data objects per interface $h_{v,r,s}$ and the probability of diffusion can be established. There is no interface if no data objects are exchanged and thus no diffusion is possible. The probability is assumed to increase monotonically with the number of exchanged objects and approaches one. The direction of exchange is irrelevant, since both the submitter and receiver module need to process all data objects being exchanged. Because the number of exchanged data objects $h_{v,r,s}$ is a natural number in reality, $d(h_{v,r,s})$ is a discrete function. Simplifying, it is assumed here that this function is continuously defined, but only evaluated at discrete points.

(A5) Changes diffuse between modules independently of direction of data exchange. There is a function $d(h_{v,r,s})$ which describes the probability for diffusion of changes through an interface depending on the number of exchanges of data objects $h_{v,r,s}$ for every interface. It is a continuous, strictly monotonically increasing function where $d(0) = 0$ and $lim_{h_{v,r,s}\to\infty} d(h_{v,r,s}) = 1$.

First, all functionalities of an affected module are investigated, modified and tested as needed to adapt this module. Then, conditioned by the diffusion all directly connected modules are processed. This is repeated for every module containing functionalities affected by the change of requirement A_a successively. In software systems simultane-

ous changes are often avoided since that can cause faults. Changes carried out successively ensure that they have no mutual influence (Fowler 2003, p. 66, 67). This leads to the above-mentioned procedure. It is assumed that in each case only directly connected modules are affected by the encapsulation of content. In reality, diffusion of changes to indirectly connected modules is possible (Beszédes et al. 2007, S. 296), but encapsulation makes it nominal. For simplification it is assumed that there is no such diffusion on indirectly connected modules.

(A6) First, all changes affecting one module are executed. Afterwards all changes caused by diffusion to adjacent modules are executed, whereby only directly connected modules can be affected by diffusion. This procedure is repeated for every affected module successively.

Risk is considered but not valued in the presented model since a risk neutral decision maker is assumed.

(A7) A risk neutral decision maker is assumed. The interest rate is denoted by z.

Tab. III-3: Variables

Variable	Intent
t	point in time
z	interest rate
v	count variable of MSOs
i, j	count variable of functionalities f_i and f_j with $i \neq j$
r, s	count variable of modules $MO_{v,r}$, $MO_{v,s}$ of MSO_v, with $r \neq s$
$n_{v,r}, n_{v,s}$	number of functionalities within modules $MO_{v,r}$ and $MO_{v,s}$ of MSO_v
l_v	number of modules of MSO_v
$g_{i,j}$	number of exchanged data objects between f_i and f_j
$h_{v,r,s}$	number of exchanged data objects between module $MO_{v,r}$ and $MO_{v,s}$ of MSO_v
a	count variable of factual requirements A
p_a	probability of change of factual requirement A_a
$q_{a,i}$	conditional probability, that functionality f_i needs to be changed if factual requirement A_a changes
CoF	(average) cash outflows per functionality that needs to be investigated, modified (if necessary) and tested

2.3.2 Model

The following model calculates the net present value of the expected cash outflows for factually driven changes during maintenance for one MSO. The inferred hypotheses may serve as a basis to compare different MSOs and to identify the MSO with the lowest expected cash outflows. According to the assumptions, only cash outflows for factually driven changes during maintenance are relevant.

The model calculates the net present value of the total expected cash outflows CoT_v of factually driven changes per MSO_v. They are the sum of the net present values of the expected cash outflows $CoT_{v,a}$ for adaption of SW caused by alteration of requirement A_a:

$$CoT_v = \sum_{a=1}^{k} CoT_{v,a}$$

The net present value of the expected cash outflows $CoT_{v,a}$ for changing all modules of MSO_v, which are affected by a change of requirement A_a, consists of the sum of the net present values of the expected cash outflows for each affected module. This sum is added up over all modules. It is shown later on that cash outflows for a certain module are larger than zero if and only if a module is affected by an initial change. The net present value of the expected cash outflows for an affected module consist of the net present value of the expected cash outflows for each initial change $CoMI_{v,a,r}$ plus the sum of the net present values of the expected cash outflows $CoMS_{v,a,s}$ for the diffusion of changes to connected modules. Again, the sum of $CoMS_{v,a,s}$ is added up over all modules. Likewise, it is shown later on that only cash outflows for directly connected modules are considered, since the probability for diffusion to not directly connected modules is always zero.

$$CoT_{a,v} = \sum_{r=1}^{l_v} \left(CoMI_{v,a,r} + \sum_{s=1}^{l_v} CoMS_{v,a,s} \right)$$

The net present value of the expected cash outflows $CoMI_{v,a,r}$ for the initial change of a module $MO_{v,r}$ consists of the product of probability $o_{v,a,r}$ of an initial change of $MO_{v,r}$, the number of functionalities $n_{v,r}$ of $MO_{v,r}$ and according to *(A3)* the discounted value of the cash outflows CoF for investigation, modification (if necessary) and test of one functionality.

$$CoMI_{v,a,r} = o_{v,a,r} \cdot n_{v,r} \cdot \frac{CoF}{(1+z)^{t_a}}$$

The probability $o_{v,a,r}$, for a change of $MO_{v,r}$ is the product of the probability p_a for the change of requirement A_a and the conditional probability of the change of at least one functionality of $MO_{v,r}$ on condition that A_a changes. This conditional probability constitutes the complement to the conditional probability that no functionality changes. The conditional probability that no functionality of $MO_{v,r}$ changes is equal to the product of each conditional probability $(1 - q_{a,i})$ that the particular functionality within $MO_{v,r}$ does not change:

$$o_{v,a,r} = p_a \cdot \left(1 - \prod_{i=1}^{n} (1 - q_{a,i})^{x_i}\right) \quad \text{with} \quad x_i = \begin{cases} 1 & \text{if } f_i \in MO_{v,r} \\ 0 & \text{if } f_i \notin MO_{v,r} \end{cases}$$

By determining the probability through $o_{v,a,r}$ it is ensured that expected cash outflows for a change of any module $MO_{v,r}$ only exceed zero $(o_{v,a,r} > 0)$ if at least one functionality of this module is affected by a change with conditional probability exceeding zero $(q_{a,i} > 0)$.

The net present value of the expected cash outflows $CoMS_{v,a,s}$ for diffusion of changes from module $MO_{v,r}$ to a directly connected module $MO_{v,s}$ consists of the product of the probability $o_{v,a,r}$ for the module $MO_{v,r}$ initially to be changed, the probability $u_{v,r,s}$ for diffusion through the corresponding interface between $MO_{v,r}$ and $MO_{v,s}$, the number of functionalities $n_{v,s}$ within the corresponding adjacent module $MO_{v,s}$ and the discounted value of the cash outflows CoF for investigation, modification (if necessary) and test of one functionality:

$$CoMS_{v,a,s} = o_{v,a,r} \cdot u_{v,r,s} \cdot n_{v,s} \cdot \frac{CoF}{(1+z)^{t_a}}$$

According to *(A5)* the probability $u_{v,r,s}$ of diffusion through an interface between the modules $MO_{v,r}$ and $MO_{v,s}$ is given by a function $d(h_{v,r,s})$ with the properties stated in *(A5)*. An exemplary function $d(h_{v,r,s})$ is:

$$u_{v,r,s} = d(h_{v,r,s}) = 1 - \left(\frac{1}{h_{v,r,s}+1}\right)^{\tau} \quad \text{with} \quad \tau \in \mathbb{R}_0^+$$

Changes can only diffuse through existing interfaces. If there is no interface as in the case that module $MO_{v,s}$ is not directly connected to $MO_{v,r}$ the probability $u_{v,r,s}$ of diffusion

of changes equals zero ($u_{v,r,s} = 0$) and no cash outflows accrue. As stated above, this is the reason that expected cash outflows $CoMS_{v,a,s}$ can mathematically be added up for all modules and not only for adjacent modules.

Summing up the net present value of the total expected cash outflows CoT_v for all factually driven changes per MSO_v of SW arises as:

$$CoT_v = \sum_{a=1}^{k} \frac{1}{(1+z)^{t_a}} \cdot \left(\sum_{r=1}^{l_v} \left(o_{v,a,r} \cdot n_{v,r} \cdot CoF + \sum_{s=1}^{l_v} \left(o_{v,a,r} \cdot u_{v,r,s} \cdot n_{v,s} \cdot CoF \right) \right) \right)$$

Using the presented model different modular structure options MSO_v can be compared regarding their net present value of expected cash outflows for factually driven changes. On this basis, further hypotheses on which MSOs cause more or fewer cash outflows for factually driven changes during software maintenance than others can be inferred.

2.3.3 Application Example

As stated in Riege et al. (2009, p. 81) an application example is an appropriate method to check the artifact for the stated research gap. Hence we refer again to the scenario described in the chapter on the outline of research scope (see Fig. III-3). In addition to the mapping of the functionalities to the requirements, the number of exchanged data objects is shown in Fig. III-6.

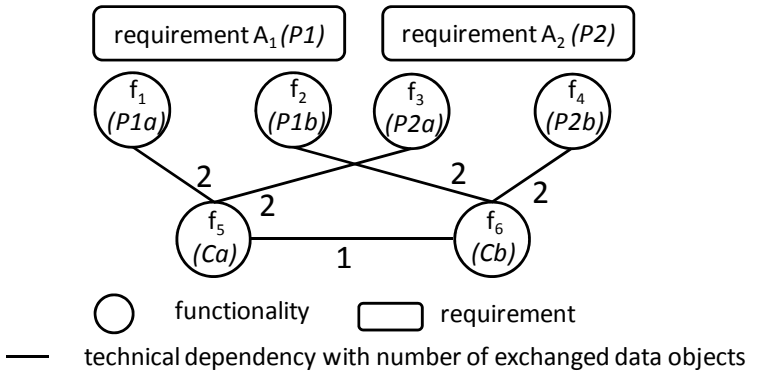

Fig. III-6: Application Example – Dependencies

The probabilities and the points in time of changes of requirements and the conditional probabilities of changes of functionalities, estimated by experts, are illustrated in Tab. III-4.

Tab. III-4: Application Example – Probabilities

A_a	t_a	p_a	$q_{a,i}$					
			f_1 (P1a)	f_2 (P1b)	f_3 (P2a)	f_4 (P2b)	f_5 (Ca)	f_6 (Cb)
A_1 (P1)	1	0.5	0.6	0.7	0.0	0.0	0.0	0.0
A_2 (P2)	1	0.7	0.0	0.0	0.4	0.8	0.0	0.0

Based on practical experience, in this example we assume that the average value of payoffs for investigation, modification (if necessary) and test of one functionality is determined by 0.5 person days on average with an imputed daily rate of 500€. Thus, 250€ correspond to cash outflows for investigation, modification (if necessary) and test of one functionality. The probability for diffusion is given by the stated function $u_{v,r,s} = d(h_{v,r,s}) = 1 - \left(\frac{1}{h_{v,r,s}+1}\right)^{\tau}$ whereby τ is exemplarily set to 0.8. The interest rate is equal to 0.1. The results of the calculated net present value of the cash outflows for factually driven changes of the two compared MSOs (see Fig. III-4) are summarized in Tab. III-5.

Tab. III-5: Application Example – Results

	MSO_1	MSO_2
$CoT_{v,1}$	631.82 €	316.95 €
$CoT_{v,2}$	816.51 €	443.73 €
CoT_v	1448.33 €	760.68 €

The currently popular approach in theory and practice would have preferred MSO_1, since it yields low coupling. But by applying the presented model it became clear that MSO_2 reduces the cash outflows for factually driven changes during maintenance considering technical and factual dependencies, because the number of initially affected modules decreases. In this example, only six functionalities were evaluated. In practice,

software systems have many more functionalities and therefore a higher potential for savings.

To afford a more widespread evaluation in practice a scenario analysis can be used. The procedure is to estimate the probabilities for changes and diffusion of changes for three different scenarios (optimistic, neutral and pessimistic) and assign an occurrence probability for each scenario. Hence, reality can be approximated in a more realistic way. In this application example a scenario analysis could for instance look like Tab. III-6.

Tab. III-6: Application Example – Scenario Analysis

optimistic scenario – occurrence probability = 0.2								
A_a	t_a	p_a	$q_{a,i}$					
			f_1 (P1a)	f_2 (P1b)	f_3 (P2a)	f_4 (P2b)	f_5 (Ca)	f_6 (Cb)
A_1 (P1)	1	0.5	0.3	0.3	0.0	0.0	0.0	0.0
A_2 (P2)	1	0.5	0.0	0.0	0.4	0.4	0.0	0.0
neutral scenario – occurrence probability = 0.5								
A_a	t_a	p_a	$q_{a,i}$					
			f_1 (P1a)	f_2 (P1b)	f_3 (P2a)	f_4 (P2b)	f_5 (Ca)	f_6 (Cb)
A_1 (P1)	1	0.5	0.6	0.7	0.0	0.0	0.0	0.0
A_2 (P2)	1	0.7	0.0	0.0	0.4	0.8	0.0	0.0
pessimistic scenario – occurrence probability = 0.3								
A_a	t_a	p_a	$q_{a,i}$					
			f_1 (P1a)	f_2 (P1b)	f_3 (P2a)	f_4 (P2b)	f_5 (Ca)	f_6 (Cb)
A_1 (P1)	1	0.8	0.65	0.7	0.0	0.0	0.0	0.0
A_2 (P2)	1	0.8	0.0	0.0	0.75	0.8	0.0	0.0

The results of the stated scenario analysis in terms of the cash outflows of both MSOs for factually driven changes are shown in Tab. III-7.

Tab. III-7: Application Example – Results of Scenario Analysis

	MSO_1	MSO_2
$CoT_{v,1}$	689.17 €	349.94 €
$CoT_{v,2}$	847.61 €	432.21 €
CoT_v	1536.78 €	782.15 €

Although MSO_1 seems to be easier to maintain than MSO_2 at a first glance, in this case the cash outflows for changes of MSO_1 are higher than the cash outflows for changes of MSO_2.

This example demonstrates that the proposed model can be used to infer new hypotheses whether a certain MSO causes more or less maintenance effort than other MSOs. Furthermore, it becomes clear that there are possible cases, where the predominant approach to cluster functionalities solely according to technical dependencies misses the basic goal of reducing software maintenance effort. Hence, it is beneficial to consider both technical and factual dependencies when valuating different modular software architectures with respect to the associated maintenance effort.

2.4 Evaluation

After the development of an approach as a basis to value different MSOs the contribution now has to be evaluated according to the requirements formulated above.

A major issue of this contribution was to increase intersubjective comprehensibility and to build an initial artifact, which can be validated by transforming the problem of MSO valuation to a quantitative formal model. According to Meredith et al. (1998, p. 314) experts of the respective field of study can understand and verify the research due to an explanation of actually observable cause-effect relationships which were described in the literature review. However, the development and operation of IT systems is not only influenced by technical factors which can be strictly modeled quantitatively, but also by social and organizational factors. These impacts are often not apparent or cannot be modeled economically, which makes a holistic validation impossible. This is not a method related effect, but applies to most formal deductive artifacts of all areas of research in all scientific disciplines. Nevertheless R-1 (intersubjective comprehensibility)

should be fulfilled.

Moreover, the results of the literature analysis revealed that there is no appropriate approach which considers all the particular requirements (economic valuation of modular software architectures considering technical and factual dependencies). Thus, the presented model satisfies requirement R-2 (innovative add-on) as well.

The scope of validity of the valuation model is not restricted to a single case. It is valid for different kinds of individually developed business software in several industries. Considering the use of abstract constructs like functionalities, data exchange, factual dependencies etc. the model is also applicable to different kinds of software and programming languages. Therefore, the model meets R-3 (generalization), too.

Last but not least, the intended benefit - according to the substantial objective of this contribution - is to reduce software maintenance costs. This serves users as well as developers of individual business software, so that there are also arguments in favor of R-4 (benefit for stakeholders).

Of course the formal model is affected by limitations and assumptions, which are discussed below. First of all, the model only considers expected cash outflows for factually driven changes. On this basis, different MSOs can be compared. But to investigate if the implementation of a certain MSO is economically beneficial, all cash flows have to be considered and the model has to be extended.

(A1) implies identical cash outflows for the development of each MSO that is irrelevant for the decision. Indeed, these cash outflows can vary depending on the number of implemented modules and interfaces. Furthermore, according to *(A3)* it is assumed that cash outflows for changing one functionality are always the same. In practice, these cash outflows can be different depending on size and complexity of the functionalities. They may also vary according to the kind of change, for example whether a single parameter within a method or the whole control logic needs to be changed. Therefore, average values should be used. If the software is too heterogeneous, the model should be applied separately to different parts of the system with sufficient homogeneity instead.

Furthermore, *(A4)* assumes that for all functionalities within a module, the same cash outflows incur due to investigation, modification (if necessary) and test, if at least one functionality is affected by the initial change within this module. In reality, most likely not all functionalities of a module are affected by a change in the same way. Some function-

alities may only need to be investigated and tested superficially while others have to be analyzed and tested in depth and even have to be modified. Thus, cash outflows of varying amounts may incur for different functionalities. Nevertheless, as a module is a unit (Pomberger and Pree 2004, p. 137) it seems to be an arguable assumption that all functionalities within a module are affected by changes in some way or another. To deal with this simplification, average values for the incurring cash outflows should be used.

Beyond this, the provided approach only considers changes of existing functionalities. In reality, changing requirements lead to addition of new functionalities, too (Krishnan et al. 2004, p. 396). Since, by adding further functionalities and modules, new dependencies are introduced as well, the comparison of different MSOs can lead to different results when considering changes of existing functionalities as well as addition of new functionalities. Therefore, future work must aim to extend the model accordingly, for which the provided approach can serve as a basis.

In contrast to *(A6)*, changes can possibly diffuse on indirectly connected modules as well (Beszédes et al. 2007, S. 296). By using the principle of encapsulation, the probabilities and accruing costs should be very low.

2.5 Synopsis and Prospect

2.5.1 Scientific Contribution

As elaborated in the literature overview, there is a research gap referring to the allocation of functionalities to modules considering both technical and factual dependencies. Thus, new hypotheses in terms of cause-effect relationships between a certain MSO and the corresponding net present values of cash outflows for factually driven changes during software maintenance are inferred. On this basis, different MSOs can be compared to support the choice of a MSO during software development.

By using a formal deductive research approach, this contribution provides an artifact which formulates the research problem and a possible solution in a formal and quantitative way. As so far, literature lacks such an artifact, this contribution provides a scientific benefit. This artifact may serve as a starting point for a profound evaluation. Beyond this, further hypotheses can be inferred, for example whether a certain MSO causes more or less maintenance effort due to factually driven changes than another MSO. As such hypotheses could be hard to be inferred directly from observations of reality, this contribution may provide additional benefit for future empirical research in this field. Using a simplified application example for the evaluation of the presented model, it was already

possible to show that there are possible cases where a MSO clustering functionalities according to factual dependencies causes less maintenance effort than the predominant paradigm to cluster functionalities primarily according to technical dependencies.

Thus, this confirms that further discussions and research in this field seem to be beneficial. Of course the model presented in this paper has to be regarded as a modest contribution towards a holistic approach to value based software development and maintenance. As elaborated in the evaluation, as in all formal deductive research approaches, simplifying assumptions had to be made. One objective of further research has to be to relax the assumptions and to further incrementally evaluate the approach by means of multiple case studies and empirical research.

2.5.2 Practical Impact

As mentioned above, the presented model provides a method to value modular software architectures with respect to the associated maintenance effort considering both technical and factual dependencies. Thus, companies can apply this method to compare different modular software architectures during the design phase of software projects in order to identify and choose the architecture with the lowest net present value of expected cash outflows for factually driven adaptions during maintenance. But, at this point, it has to be made clear, that the new hypotheses presented in this paper cannot be used to derive recommendations for actions without any restrictions. The presented model is based on simplifying assumptions and, therefore, all results are. Hence, future research must aim to extend the proposed model in order to relax simplifying assumptions. Furthermore, all hypotheses inferred by a formal deductive approach have to undergo a critical examination. Thus, future empirical research activities should aim to test the hypotheses since empirically confirmed hypotheses provide a more solid foundation for recommendation of actions.

Nevertheless, as already mentioned in the introduction of this paper, costs for software maintenance attain up to 80% of software life-cycle costs (Krishnan et al. 2004, p. 396; Zarnekow et al. 2004, p. 181). This indicates that there is leverage for saving costs and increasing company values. A vision would be to use tools for software development like computer aided design (CAD) applications in industry, where the CAD software already estimates follow-up costs for different design alternatives of products to support design decisions. Though a similar solution for design alternatives of business software seems to be a distant prospect, the authors hope to provide a next step towards such a solution and to trigger a fruitful scientific discussion with the intention to shorten the distance.

References (Chapter III.2)

Al-Otaiby TN, AlSherif M, Bond WP (2005) Toward Software Requirements Modularization Using Hierarchical Clustering. In: Proceedings of the 43rd annual Southeast Regional Conference – Volume 2, 2-223 – 2-228

Baghdadi Y (2004) A Web Services-Oriented Approach to Unlock Information. In: Proceedings of the Information Science and Information Technology Education Joint Conference (InSITE ’04), pp. 585-596

Baldwin CY, Clark KB (2000) Design Rules: The power of modularity. Volume 1, 1st Edition, Boston: MIT

Balzert H (2009) Lehrbuch der Softwaretechnik – Basiskonzepte und Requirements Engineering. 3rd Edition, Heidelberg: Spektrum

Beszédes Á, Gergely T, Jász J, Tóth G, Gyimóthy T, Rajlich V (2007) Computation of Static Execute After Relation with Applications to Software Maintenance. In: Proceedings of the 23rd IEEE International Conference on Software Maintenance (ICSM '07), IEEE, 295-304

Briand LC, Wüst J, Lounis H (1999) Using Coupling Measurement for Impact Analysis in Object-Oriented Systems. In: Proceedings of the 15th IEEE International Conference on Software Maintenance (ICSM '99), IEEE, 475-482

Eaddy M, Zimmermann T, Sherwood KD, Garg V, Murphy GC, Nagappan N, Aho AV (2008) Do Crosscutting Concerns Cause Defects?. IEEE Transactions on Software Engineering 34(4):497-515

Fowler M (2003) Patterns for Enterprise Application Architecture. 1st Edition, Boston: Pearson Education

Friedrichsen U (2009) Der Mythos Wiederverwendung: ”Design für Wartung” als eigentliches Ziel. Objektspektrum (4/2009), 42–47

Frühauf K, Ludewig J, Sandmayr H (2007) Software-Prüfung: Eine Anleitung zum Test und zur Inspektion. vdf Hochschulverlag, Zürich

Gamba A, Fusari N (2009) Valuing Modularity as a real Option. Management Science 55(11):1877–1896

Gartner (2011) Gartner Says Worldwide IT Spending to Grow 5.1 Percent in 2011. http://www.gartner.com/it/page.jsp?id=1513614 recalled 2011-05-02

Helmke H, Höppner F, Isernhagen R (2007) Einführung in die Software-Entwicklung: Vom Programmieren zur erfolgreichen Software-Projektarbeit. 1st Edition. München: Hanser

Henry S, Kafura D (1981) Software Structure Metrics Based on Information Flow. IEEE Transactions on Software Engineering 7(5):510–518

jCOM1 (2009) Software-Änderungen kosten die DAX 30-Konzerne über 1 Mrd. Euro jährlich. http://www.metasonic.de/softwareaenderungen-kosten-dax-30-konzerne-ueber-1-mrd-euro-jaehrlich, recalled 2010-12-02

Kazman R, Klein M, Clements P (2000) ATAM: Method for Architecture Evaluation. Software Engineering Institute, http://www.sei.cmu.edu/library/abstracts/reports/00tr004.cfm, recalled 2010-08-24

Kemerer CF (1995) Software complexity and software maintenance: A survey of empirircal research. Annals of Software Engineering 1(1):1–22

Ko AJ, Myers BA, Coblenz MJ, Aung HH (2006) An Exploratory Study of How Developers Seek, Relate, and Collect Relevant Information during Software Maintenance Task. IEEE Transactions on Software Engineering 32(12):971-987

Krishnan MS, Mukhopadhyay T, Kriebel CH (2004) A Decision Model for Software Maintenance. Information Systems Research 15(4):396–412

Marinescu R (2002) Measurement and Quality in Object-Oriented Design. Ph.D. thesis. Department of Computer Science, Politehnica University of Timisoara

Meredith J, Raturi A, Amoako-Gympah K, Kaplan B (1989) Alternative Research Paradigms in Operations. Journal of Operations Management 8(4):279–326

Myers GJ (2004) The Art of Software Testing. 2nd Edition, New Jersey: John Wiley & Sons

Parnas DL (1972) On the criteria to be used in decomposing systems into modules. Communications of the ACM 15(12):1053–1058

Parnas DL, Clements PC, Weiss DM (1985) The modular structure of complex system. IEEE Transactions on Software Engineering 11(3):259-266

Pomberger G, Pree W (2004) Software Engineering - Architektur-Design und Prozessorientierung. 3rd Edition, Wien: Hanser

Riege C, Saat J, Bucher T (2009) Systematisierung von Evaluationsmethoden in der Wirtschaftsinformatik. In: Becker J, Krcmar H, Niehaves B eds., Wissenschaftstheorie und gestaltungsorientierte Wirtschaftsinformatik, Heidelberg: Physica-Verlag, 69–86

Robillard MP, Coelho W, Murphy GC (2004) How Effective Developers Investigate Source Code: An Exploratory Study. IEEE Transactions on Software Engineering 30(12):889-903

Rostkowycz AJ, Rajlich V, Marcus A (2004) A Case Study on the Long-Term Effects of Software Redocumentation. In: Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM '04). IEEE, 92-101

Sommerville I (2001) Software Engineering. 6th Edition, Boston: Addison Wesley

Sullivan KJ, Griswold WG, Cai Y, Hallen B (2001) The structure and value of modularity in software design. In: Proceedings of the 8th European Software Engineering Conference held jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE-9), 99–108

Yu P, Systä T, Müller H (2002) Predicting Fault-Proneness Using OO Metrics: An Industrial Case Study. In: Proceedings of the 6th European Conference on Software Maintenance and Reengineering (CSMR '02), 99-107

Zarnekow R, Scheeg J, Brenner W (2004) Untersuchung der Lebenszykluskosten von IT-Anwendungen. Wirtschaftsinformatik 46(3):181–187

Zöller-Greer P (2010) Software-Architekturen – Grundlagen und Anwendungen. 3rd Edition, Wärtersbach: Composia

IV Systemische Risiken in Rechnernetzen im Rahmen der IT-Sicherheit (Beitrag: „*Model-based Security Analysis for Mobile Communications*“)

Autoren:	Jan Jürjens The Open University, UK, http://www.jurjens.de/jan Jörg Schreck O_2 (Germany), Joerg.Schreck@acm.org Peter Bartmann Kernkompetenzzentrum Finanz- & Informationsmanagement, Lehrstuhl für BWL, Wirtschaftsinformatik, Informations- & Finanzmanagement (Prof. Dr. Hans Ulrich Buhl) Universität Augsburg, D-86135 Augsburg peter.bartmann@wiwi.uni-augsburg.de

Erschienen in:	Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), Leipzig (Germany). ACM, 2008; p. 683-692.

Abstract:

Mobile communication systems are increasingly used in companies. In order to make these applications secure, the security analysis has to be an integral part of the system design and IT management process for such mobile communication systems. This work presents the experiences and results from the security analysis of a mobile system architecture at a large German telecommunications company, by making use of an approach to Model-based Security Engineering that is based on the UML extension UMLsec. The focus lies on the security mechanisms and security policies of the mobile applications which were analyzed using the UMLsec method and tools. Main results of the paper include a field report on the employment of the UMLsec method in an industrial telecommunications context as well as indications of its benefits and limitations.

1 Introduction

The use of mobile communication technologies has experienced an explosive growth. However, this usage carries critical risks concerning information security that are particularly significant for mobile systems, due both to the inherent vulnerability of the devices and the significant complexity of the architectures (as explained in Sect. 2). In order to address these risks and enable secure mobile communication systems, the security analysis has to be embedded in the development and management of the systems.

This work presents the results of the model-based security analysis of parts of the corporate security architecture and security policies for mobile communication systems at the German telecommunications company O_2 (Germany). The security critical parts of the system were analyzed using UMLsec (Jürjens 2004), a UML extension which allows the user to embed security related information into the system design, as well as to conduct security analyses on the model layer. The goal of this work was to gain experiences in the use of the UMLsec method in an industrial telecommunications context and to show its benefits and limitations.

Empirical studies on the use of model-based development in general have so far been limited. The challenges faced by empirical software engineering in general are discussed in Perry et al. (2000), and specifically regarding model-based developments in Schalken (2005). A report on using UMLsec in industrial development of security-critical software can be found in Apvrille and Pourzandi (2005). Another case study employing a method close to the UMLsec approach in an industrial context is Grünbauer et al. (2003), reporting on a project with a major German bank. Vetterling et al. (2002) report on a student project using a security extension of the AutoFocus tool (that was developed jointly and in parallel with the UMLsec extension) to develop a payment application. Best et al. (2007) report on a case-study using UMLsec regarding an intranet search machine at BMW. An overview on other applications of model-based security engineering in practice is given in Jürjens (2006a). A case-study on using model-based approach to secure software engineering and management in telecommunication systems (and in particular mobile communication systems) has to the extent of our knowledge not been published so far. Although this paper does not aim to be a complete or controlled empirical study, we hope to contribute to filling this gap using our report on the industrial case-study presented here.

The paper is structured as follows. In Sect. 2 we give some background on the security

challenges to software for mobile systems in an industrial telecommunications environment. In Sect. 3, we explain what the goals of the security analysis were, we introduce the UMLsec based analysis methodology. In Sect. 4, we present and discuss the results of the security analysis at O_2 (Germany) at the hand of some representative fragments of the models that were constructed. In Sect. 5, we discuss some lessons learned from the case study. We end with a summary indicating further work.

2 Software Security Challenges in Mobile Communications

We discuss the particular characteristics of mobile communication systems regarding security requirements, and how software engineering and software management processes have to be adapted to the application domain of mobile telecommunication systems in order to deal with the resulting challenges.

2.1 Characteristics of Mobile Security

Mobile networks differ from fixed networks in many respects. We give a rough overview over the most obvious differences with respect to fixed networks which can be found in almost every company. With the term *mobile security*, we refer to all security aspects which are relevant for mobile networks, many of which are also relevant for fixed networks.

2.1.1 Architecture

The most distinguishing aspect of mobile networks is their architecture. Whereas static structures and homogeneous components predominate fixed networks, the situation is usually quite the opposite for mobile networks. In the following we discuss some representative aspects of this which lead to significant complexity:

Different access media

The type of access medium to the network depends not only on the need of the network’s users but also on environmental conditions such as costs or availability. Often, one access medium alone is not sufficient because differences in throughput, latency, cost, and availability justify having alternatives. The access media evolved over time and comply to different standards, such as:

- *WLAN:* IEEE 802.11{a,b,e,g,h,i}, WEP, WPA{1,2}
- *Bluetooth:* IEEE 802.15.1, Version 1.1, 1.2, 2.0
- *Infrared:* IrDA (IrLAP, IrCOMM, IrOBEX, IrLAN)

- *Telephony:* GSM, GPRS, UMTS.

Unfortunately, the different standards for one type of access medium (e.g. WLAN) offer different security features and must thus be considered separately. The standards may be “downwards compatible” and security features may be treated as cumulative. Even within one standard, the security considerations have to consider the different subsets of the standard. This is the case, for instance, with the different *profiles* that can be activated while using Bluetooth.

Variety of end user devices

Another difference is the variety of end user devices which take part in mobile networks. With respect to security considerations, the most important criteria include:

- *Form factor*: laptops, PDAs, cellphones, smartphones etc.
- *Operating system*: Windows (XP, 2000), Windows Mobile, Symbian OS, Palm OS, Linux, Blackberry etc.

Due to different operating systems and hardware capabilities several security requirements (e.g., separation of duties of different system users) are already fulfilled, can be added to the system, or cannot be fulfilled due to restrictions of the underlying system. This leads to various combinations depending on the set of requirements essential for the given usage scenario. Some of them are described in the following.

Different protection mechanisms

As described above, different security requirements may be applicable for different usage scenarios. Within those we can identify a subset of requirements which are applicable for almost each usage scenario. The following list shows the most important and gives some explanation why they cannot be treated in general for all possible architectures.

Authentication: The required grade of authentication depends on the data and services to be protected. While public information might be accessed without authentication, this is not acceptable for private data or services which are offered to employees only. In addition to the different levels which might require authentication (e.g., the end user device, network connections, internal services) there are also several grades of authentication (e.g., password, one-time passwords, two-factor authentication, biometrical procedures) to choose from which increases the number of possible combinations that have to be checked.

Network security: Mobile devices usually offer very limited means to secure the network layer. A reason for this are their limited resources which often do not allow for transparent encryption (e.g., for a VPN tunnel) and the limited availability of the necessary software for the chosen platform (e.g., firewall, virus scanner).

Application security: Mobile devices which are not always connected cannot be secured like their counterparts within a fixed network. E.g., the provision of patches, update of signatures for virus scanners, and central processing of log file entries can only performed the next time the device is connected. Also well-established security measures are often not feasible on certain handhelds, e.g., due to their low performance (such as an IPS running on a smartphone).

Secrecy: As stated above, the type of encryption of data (e.g., transparent or after usage) depends on the performance of the device. Moreover, the secrecy of the data is also influenced by the physical capabilities of the device. Some are able to use enhanced encryption techniques like smartcards or biometric components while others are not.

Availability: As mentioned, developers of mobile systems often need to design secure architectures for different access media and often also to include different access media into one architecture. More complexity is added when only one security concept must be developed with disregard to the access media - which might change during the usage of the system due to environmental conditions.

2.1.2 Usage

The above-mentioned differences are mainly due to technical restrictions. Beside these restrictions there are also restrictions with origin in the users’ behavior when using mobile networks. Some of them are sketched below:

Spontaneous usage: Beside users who take advantage of mobile networks as a replacement of their office desk, there is a high amount of *nomadic users* who have to adapt to different environmental conditions or time constraints with only very limited options several times a day, e.g., travelers at an airport. In contrast to users in an office who are willing to accept (once a day) a fixed procedure to gain (and suspend) access to resources and services, these nomadic users usually do not accept login or logoff procedures which are not negligible with respect to the usage period. Since the usage periods are often quite short, the procedures for preparing the service have to be much shorter. Unfortunately, this often leads to reduced acceptance of full-blown security

checks and therefore to a lower security standard.

Limitation of services: On the other hand, the reduced usage period entails also a limited set of services which can be used in it. As a consequence, users accept a reduced amount of services compared to the ones they are offered in an office environment. Often, it is sufficient to provide access to emails, appointments, and addresses.

Always on / ABC: Users of mobile networks often want to be connected always in order to receive the latest information and to have necessary information at their disposal. Also, they want the best combination between reliability/throughput and price of the access media (*ABC, always best connected*). This shows once again the necessity to change the access medium within a session.

Average useful life: The average useful life of devices used within mobile networks usually is much shorter than that of components within fixed networks. As a rule of thumb, the average useful life should not expected to be longer than 18 months (in contrast to 36 months), although one should note that a hard end-point on this can usually not be defined. This entails that the respective architecture has to be evaluated with each new generation of end devices which leads to the double amount of assessment.

2.2 Implications for Software Engineering and Management

Based on the discussion of the particular characteristics of mobile telecommunication systems in particular with respect to security requirements in the previous section, one can derive the following characteristics that have to be taken into account specifically when designing mobile communication architectures:

- higher variety of access media
- higher variety of devices and operating systems
- less protection mechanisms included by default
- spontaneous use of usually less services
- cost of additional software might exceed device cost
- biased compromise between security and usability
- reduced time for amortization and frequent redesign
- only few audits or certifications available.

Taking into account all different possible combinations of the above mentioned criteria we are faced with an amount of several thousands of different (possible) architectures

which must be analyzed and optimized for their proper implementation of security requirements in order to choose the most effective, the cheapest, and the most comfortable. Each of these architectures may in itself be quite complex and non-trivial to analyze for the given security requirements.

The discussions above and in the last subsection make clear that the necessity of analyzing all of these potentially quite complex architectural alternatives against the security requirements makes this task only feasible at a satisfactory level of trustworthiness if it is supported by an systematic and efficient process which makes use of automated security analysis tools for the architectural options as far as possible. To perform this task at the given application at O_2 (Germany), we therefore developed such a process, which is explained in the following section and then applied in Sect. 4.

3 Security Analysis of Mobile Communication Architectures

3.1 Requirements on the Security Analysis

The main goal of a security analysis is a satisfactory level of confidence that a given security policy or particular security requirements are fulfilled. We give some further requirements on the security assessment process for mobile communication architectures, motivated by the discussions in the previous section, in particular by the high number of architectural alternatives that may need to be analyzed:

Reproducability: The results need to be reproducible for a given architecture without risk of misinterpretation.

Delegability: It is required that at least parts of the analysis can be delegated to be feasible in practice.

Efficiency: The analysis must be performed in a given time-frame with a defined expectation regarding thoroughness and scope. The necessary amount of work done by a human security expert should be reducable by limiting the scope of the analysis.

Parallelization: It must be possible that parts of the analysis can be performed in parallel and independently.

Traceability: Results of the analysis must be traceable and give guidance how negative results can be improved on.

Expressiveness: Results must carry enough information to enable an overall risk analysis of a given architecture.

To achieve these requirements, we decided to evaluate the use of a security assessment process which includes the use of models related to the given architectures and security requirements, and of automated tools to analyze these models against the given security requirements. To keep the amount of additional training bounded, we chose an approach based on the Unified Modeling Language (UML), and one of the options available here is the security extension UMLsec of the UML (to be introduced in the next subsection).

3.2 Security Analysis using UMLsec

Model-based Security Engineering (MBSE) (Jürjens 2004, Jürjens 2005, Jürjens and Shabalin 2007) provides a soundly based approach for developing security-critical software where recurring security requirements (such as secrecy, integrity, authenticity and others) and security assumptions on the system environment, can be specified either within a UML specification, or within the source code as annotations (cf. Fig. IV-1). Various analysis plugins in the associated UMLsec tool framework (UMLsec tool 2008) (Fig. IV-2) generate logical formulas formalizing the execution semantics and the annotated security requirements. Automated theorem provers and model checkers automatically establish whether the security requirements hold. If not, a Prolog-based tool automatically generates an attack sequence violating the security requirement which can be examined to determine and remove the weakness. Thus we encapsulate knowledge on prudent security engineering and make it available to developers who may not be security experts. Since the analysis that is performed is too sophisticated to be done manually, it is also valuable to security experts. Part of the MBSE approach is the UML extension UMLsec for secure systems development which allows the evaluation of UML specifications for vulnerabilities using a formal semantics of a simplified fragment of the UML. The UMLsec extension is given in form of a UML profile using the standard UML extension mechanisms. *Stereotypes* are used together with tags to formulate the security requirements and assumptions. *Constraints* give criteria that determine whether the requirements are met by the system design, by referring to a precise semantics of the used fragment of UML. The security-relevant information added using stereotypes includes security-relevant information covering the following aspects:

- Security assumptions on the physical system level, for example the stereotype `<<encrypted>>`, when applied to a link in a UML deployment diagram, states that this connection has to be encrypted.

- Security requirements on the logical level, for example related to the secure handling and communication of data, such as `<<secrecy>>` or `<<integrity>>`.
- Security policies that system parts are required to obey, such as `<<fair exchange>>` or `<<data security>>` (see Sect. 4 for explanations).

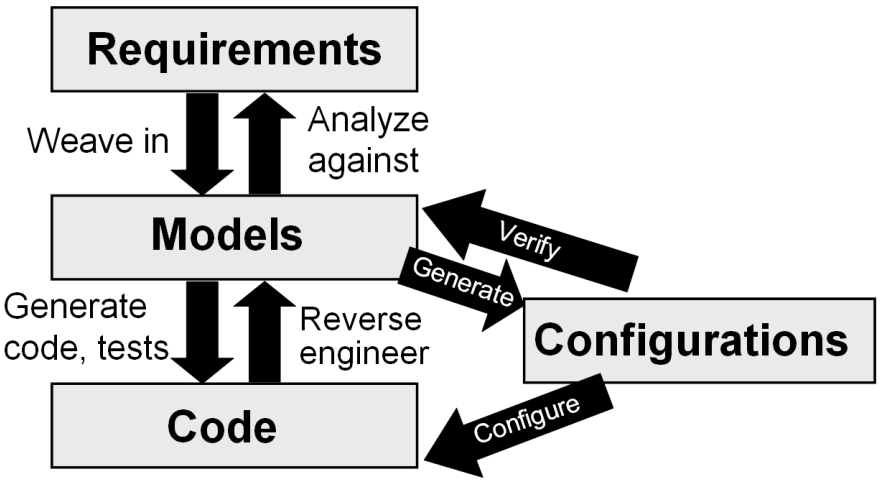

Fig. IV-1: Model-based Security Engineering

The UMLsec tool-support in Fig. IV-2 can then be used to check the constraints associated with UMLsec stereotypes mechanically, based on XMI output of the diagrams from the UML drawing tool in use (UMLsec tool 2008, Jürjens 2005). There is also a framework for implementing verification routines for the constraints associated with the UMLsec stereotypes. Thus advanced users of the UMLsec approach can use this framework to implement verification routines for the constraints of self-defined stereotypes. The semantics for the fragment of UML used for UMLsec is defined in Jürjens (2004) using so-called *UML Machines*, which is a kind of state machine with input/output interfaces and UML-type communication mechanisms. On this basis, important security requirements such as secrecy, integrity, authenticity, and secure information flow are defined.

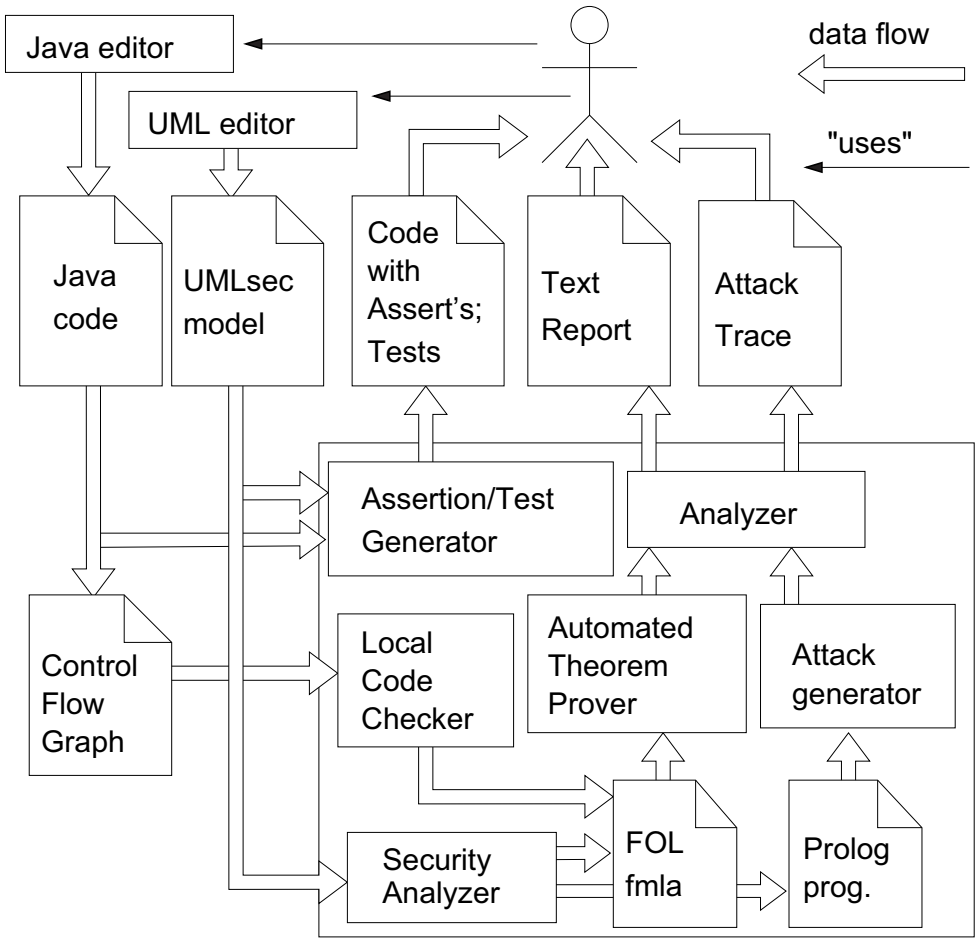

Fig. IV-2: Model-based Security Tool Suite

3.3 Modeling of Security Requirements

Starting from a given set of security requirements (i.e., the company`s security policy) which focuses on mobile security, we conducted an analysis to identify the parts of the UMLsec framework which are most suitable to model these requirements and allow for a subsequent security analysis.

The main objectives for the selection of requirements, parts of the frameworks, and modeling objects have been:

- focus on requirements which affect the architecture
- reduction of complexity by abstraction and generalization
- reusability of the model

The models resulting from the modeling process are thus:

- data
- data flows
- network objects and their connections
- services
- protection goals
- protection measures and devices
- processes

As a result of this modeling process, we develop a model focused on particular parts of a given security policy, which is considered to be complete, consistent, rather stable, and independent of the concrete architecture to be analyzed.

3.4 Analysis Process

In order to support the goals of the security analysis in Sect. 3.1 we developed an analysis process which takes advantage of three different kinds of models. These models are sketched in Fig. IV-3 and described as follows (see Sect. 4 for examples):

Security Requirements Model: The creation of this model is described in Sect. 3.3. To be able to specify the security requirements, this model may include architectural or behavioral views of the system, but in a prescriptive rather than descriptive way. This model satisfies all relevant security requirements by construction.

Usage Model: The Usage Model is a less formalized model (usually not using UMLsec) which describes the requirements of a target architecture from a user’s point of view.

Concretized Model: The Concretized Model describes a concrete architecture by means of UMLsec. It is considered to satisfy the requirements of the Usage Model.

The Concretized Model can be obtained by adapting the architecture-independent Security Requirements Model until it fits the Usage Model with respect to, for instance, required functionality or available protection measures.

Thus, we obtain an architecture-dependent Concretized Model which usually differs from the Security Requirements Model. By making use of different parts of the UMLsec notation and tool-support (see Sect. 4 for examples) we can:

- identify gaps between these two models by applying the framework to the Concretized Model

- document security requirements which are not met by the Concretized Model
- improve the Concretized Model through iterations and reassessment

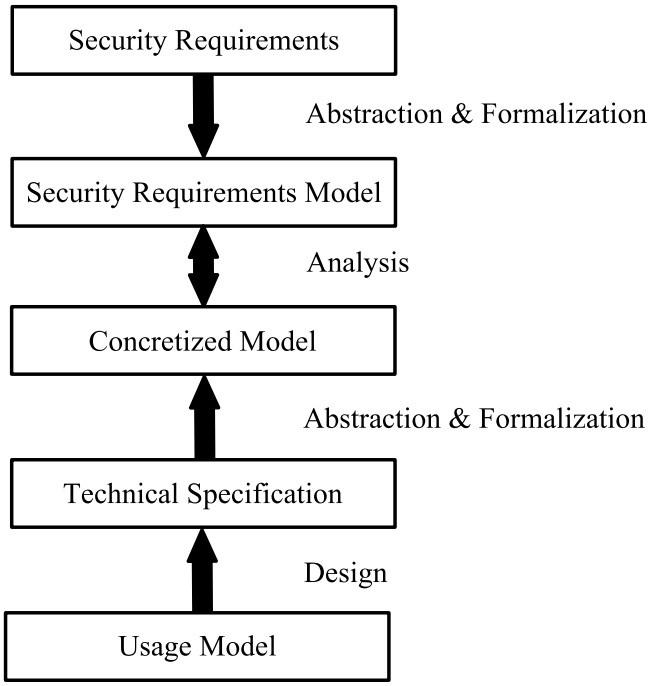

Fig. IV-3: Models used in the Analysis Process

By following this iterative process we are able to improve a target architecture with respect to security requirements. Due to the restrictions of the Usage Model and the factors described in Sect. 2, it is advisable to follow such an iterative approach in order to keep the protection measures that are needed to satisfy a given security policy subject to a given usage scenario minimal.

4 Application at O_2 (Germany)

We now explain how the process for model-based security analysis explained above was used at an application at the German telecommunications company O_2 (Germany). We illustrate this with a few representative examples.

Overall, we extracted 62 security requirements from the security policy, which we formalized for developing this model by employing the following stereotypes respectively extensions (see Jürjens (2004) for definitions and explanations):

- `<<fair exchange>>` and `<<provable>>` for 21 process-related requirements, which are formalized within eight activity diagrams.
- `<<secure links>>` for 10 security requirements regarding secrecy and integrity of data, which have to meet on the physical layer. All requirements are integrated in one deployment diagram.
- Three requirements concerning role-based access control.
- An extension of UMLsec based on logical formulas for formalizing 15 security requirements concerning net work services and dataflows which have to be regulated by firewalls resp. tested for malware by anti-virus software (see Sect. 4.3). The logical formulas which formalize all of these requirements are based on one network architecture model.
- For 13 requirements we were not able to find an appropriate UMLsec representation.

In each of the examples presented in this section, we will consider two kinds of UMLsec model: A model which represents the security requirements that should be realized according to the company security policy (Security Requirements Model), and a model which represents an implementation of the Security Requirements Model, which may already be in place within the company, or which is intended to be implemented (Concretized Model). One can then use the UMLsec tools to compare the Concretized Model with the Security Requirements Model to determine whether the Concretized Model enforces all the security requirements that should be in place according to the Security Requirements Model. In that sense, the Security Requirements Model plays the role of a “blueprint” which demonstrably enforces the security requirements given in the company security policy. Against that, the Concretized Model, which represents the existing or planned implementation, has to be compared.

4.1 Scenario 1: Virus Protection

For the first example we consider four security requirements concerning virus scans on a mobile device:

Requirement 1: If malware is found, the infected data has to be cleaned up or to be deleted completely.

Requirement 2: Furthermore, all involved users have to be informed about the malware that was found.

Requirement 3: The results of every virus scan have to be logged.

Requirement 4: A log-file has to be stored on a central storage device.

These requirements are described on a highly abstract level. So, for example, they only formulate that infected data has to be cleared up or to be deleted. They do not specify a detailed way in which these measures have to be accomplished. Thus the requirements listed above can be formalized on a process-related level. Fig. IV-4 shows the Security Requirements Model representing these security requirements in the notation of a UML activity diagram.

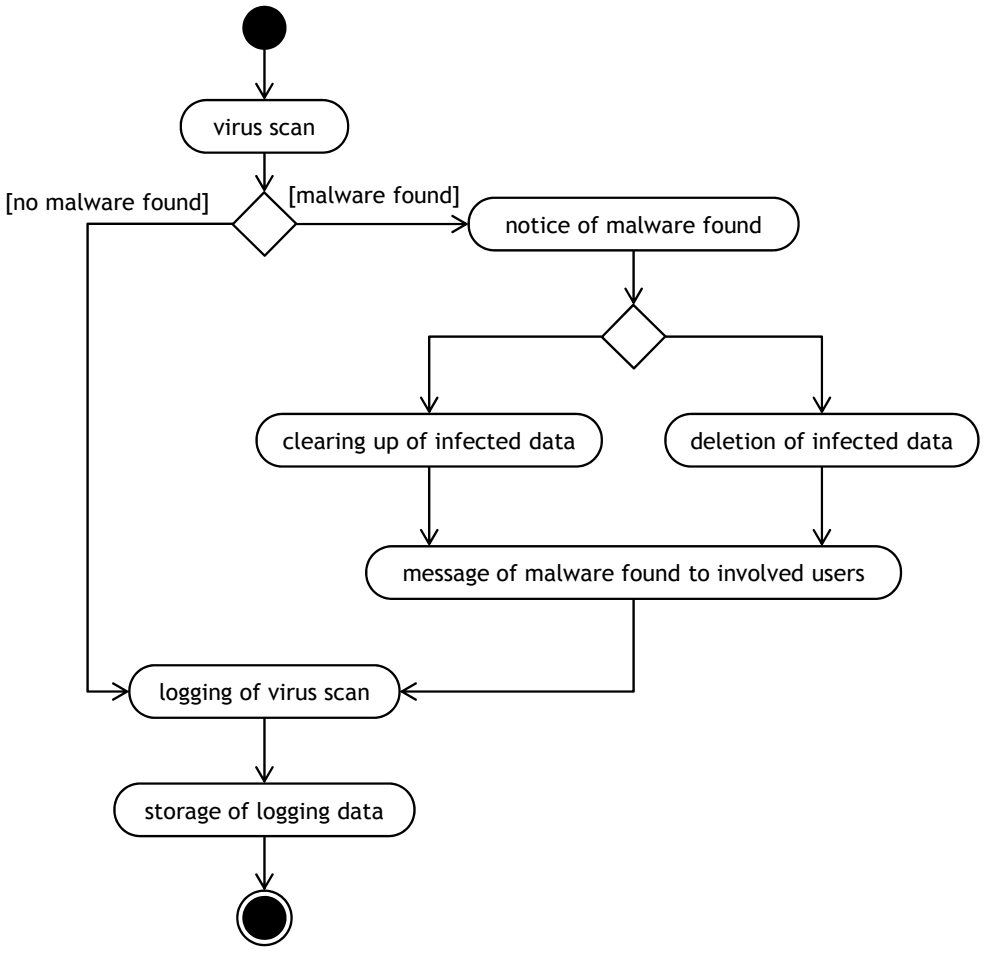

Fig. IV-4: Security Requirements Model for virus scan scenario as a UMLsec activity diagram

To complete the formalization of the security requirements, the stereotype `<<fair`

exchange>> is applied to the activity diagram and the associated tags {start} and {stop} are defined for each requirement separately:

- Tags for requirement 1:
  {start} =”notice of malware found”,
  {stop} =”clearing up of infected data”,
  {stop} =”deletion of infected data”
- Tags for requirement 2:
  {start} =”notice of malware found”,
  {stop} =”message of malware found to involved users”
- Tags for requirement 3:
  {start} =”virus scan”,
  {stop} =”logging of virus scan”
- Tags for requirement 4:
  {start} =”virus scan”,
  {stop} =”storage of logging data”

The constraint for the stereotype <<fair exchange>> requires that whenever a {start} state in the contained activity diagram is reached, a {stop} state will also eventually be reached (for a detailed description of UMLsec see Jürjens (2004)). For example, if the {start} state ”notice of malware found” is reached in the process shown in Fig. IV-4, the {stop} state ”clearing up of infected data” or the {stop} state ”deletion of infected data” will subsequently be reached. There is no sequence in which the {start} state is reached but afterwards none of the {stop} states (and this can also be checked automatically using the UMLsec tools). Thus the activity diagram combined with the stereotype <<fair exchange>> and the defined tags fulfill requirement 1, so they provide a Security Requirements Model for the first security requirement. Analogously, it can be shown that the other security requirements hold in the Security Requirements Model as well.

It is important to note that this Security Requirements Model is only reasonable under some assumptions: The Security Requirements Model only requires the existence of the process actions described above and the right embedding of these actions in the whole process, but does not describe how the particular actions itself have to work. So this security model can only be applied to a real architecture in a useful way when it is additionally ensured that these actions (e.g., ”clearing up of infected data”) work accurately

in a real world context. That is, we do not analyze the quality of these actions (e.g., the quality of an encryption algorithm), but their correct usage.

As the next step, a Concretized Model has to be built which can then be compared with the Security Requirements Model developed above. A Concretized Model is always based on a Usage Model as introduced in Sect. 3.4 which describes the software to be applied in the examined infrastructure. In this example, we assume the usage of an anti-virus software which provides functionality for deleting found malware and for logging positive results of a virus scan. Fig. IV-5 shows a simplified part of such a Concretized Model which visualizes the process of scanning data for malware on a mobile device.

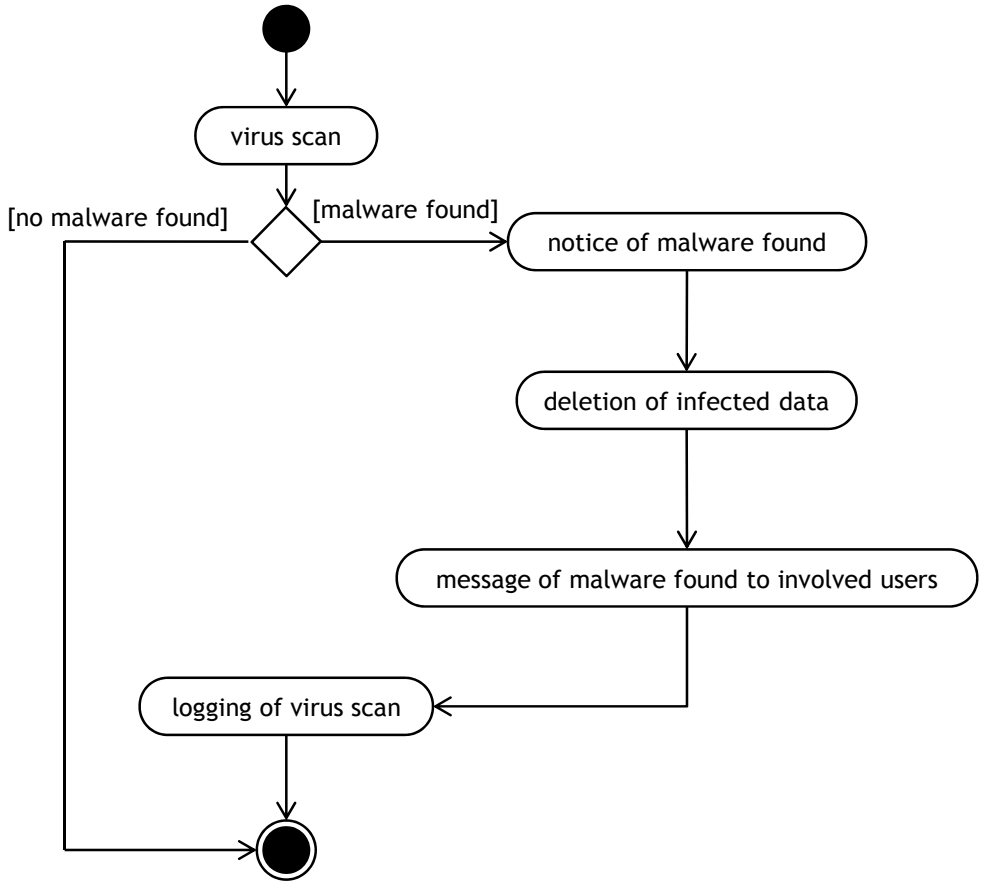

Fig. IV-5: Activity diagram for Concretized Model

Before analyzing this Concretized Model, the stereotype `<<fair exchange>>` and the tags defined for the security model are applied to the associated activity diagram. By

using the appropriate analysis plugin ”UMLsec Static Check I” in the UMLsec tool framework (UMLsec tool 2008), this Concretized Model can then be tested separately for each security requirement formalized in the Security Requirements Model above. The results of the analysis using the plugin ”UMLsec Static Check I” show that the first requirements hold in the Concretized Model: In every possible sequence of the process the state ”deletion of infected data” will be reached after the state ”notice of malware found” is reached. Analogously, the Concretized Model fulfills the second requirement. In contrast, the third and fourth requirements do not hold in the Concretized Model. If no malware is found, the process does not reach the state ”logging of virus scan”, and the state ”storage of logging data” is completely missing in the process. Thus the third and fourth security requirements are not met, which can be detected by the process described above. The gaps are displayed explicitely (e.g., by enumerating missing tags) and can be used for a subsequent improvement cycle or a risk analysis.

4.2 Scenario 2: Mobile Communication

We want to develop the second example on the basis of the following exemplary security requirements which refer to the communication between a mobile device and external storage devices respectively the company’s intranet:

Requirement 1: All data sent between a mobile device and the intranet is allowed to be read only by employees.

Requirement 2: The data stored on a mobile device is allowed to be stored only encrypted on an external storage device such that only the owner of the related mobile device is able to read the data.

Both items specify the requirement of secrecy of data sent between the mobile device and the intranet or stored on an external storage device like a memory card, which can be achieved by encryption. Since the stereotype `<<secure links>>` is used to ensure that security requirements on the communication are met by the physical layer, this stereotype seems to be appropriate to formalize these requirements. Fig. IV-6 shows a deployment diagram labeled by the stereotype `<<secure links>>` and is used as basis of the Security Requirements Model.

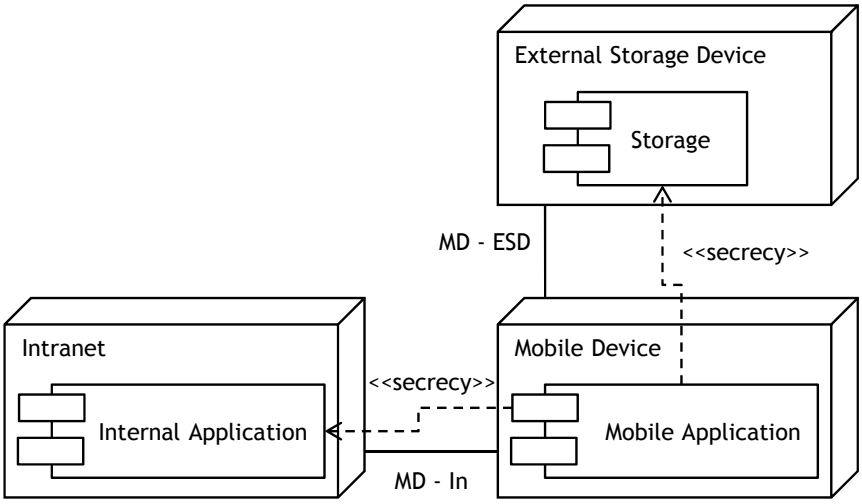

Fig. IV-6: Deployment diagram for Security Requirements Model

Part of the Security Requirements Model is the definition of the attacker types against which the security requirements have to hold. For our example we define two different attacker types, and for each the actions they can apply to various communication links (cf. Tab. IV-1 and Tab. IV-2).

Tab. IV-1: Threats from an external attacker

communication links	**threats**
plain	delete, read, insert
encrypted with company key	delete
encrypted with user key	delete

Tab. IV-2: Threats from an internal attacker

communication links	**threats**
plain	delete, read, insert
encrypted with company key	delete, read, insert
encrypted with user key	delete

For building an exemplary Concretized Model (which we can test for fulfilling the security

requirements listed above) we assume the usage of an encryption software which encrypts data sent between the mobile device and the intranet and data stored on an external storage device with a company-wide key (which every employee is assumed to know). The deployment diagram in Fig. IV-7 labeled with the stereotype `<<secure links>>` contains the relevant subsystem of the Concretized Model.

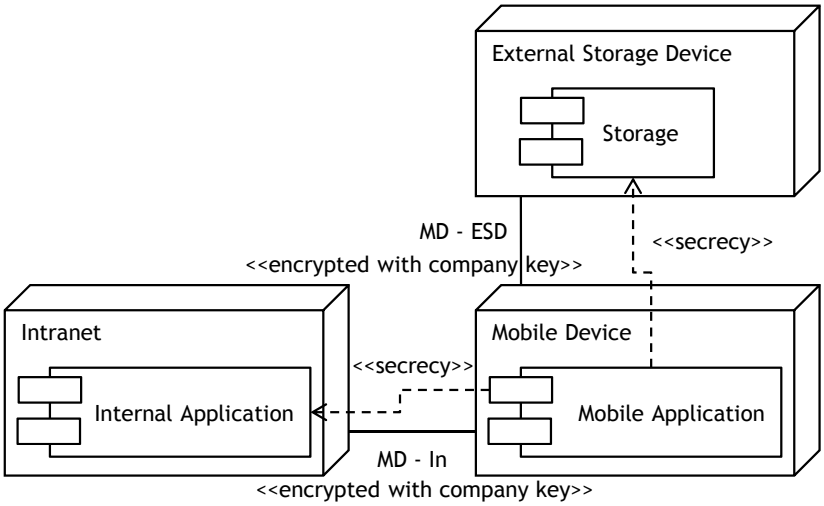

Fig. IV-7: Deployment diagram for Concretized Model

An analysis of this Concretized Model with the plugin ”UMLsec Static Check I” in the UMLsec tool framework shows that the model only satisfies the first security requirement. The constraint associated with the `<<secure links>>` enforces that for a dependency with stereotype `<<secrecy>>` between two objects on different nodes we have a communication link between these nodes such that a defined attacker type is not able to perform the threat ”read” on that communication link. According to the first requirement, an external attacker should not be able to read the data sent on the communication link between the nodes ”mobile device” and ”intranet”. Because this communication link is labeled with the stereotype `<<encrypted with company key>>`, an external attacker is not able to read any data sent on this link according to the definition of the related attacker type. By contrast, an internal attacker is able to read data sent on this communication link. However, the first requirement does not specify an employee as unauthorized to read this data. Hence the first requirement holds in the architecture model above. Analogously, an external attacker is not able to perform the threat ”read”

on the communication link between the nodes ”mobile device” and ”external storage device”, but an internal attacker is. Because the second requirement specifies that only the owner of the mobile device is allowed to read data stored on external memory, this requirement is violated by the architecture model. A solution could be to use encryption software for encrypting data with an individual key of the user.

4.3 Scenario 3: Network Security Architecture

Within the project, we identified a group of similar security requirements for whose formalization we did not find an appropriate UMLsec notation element and tool plugin. Most of these security requirements specify that there should not exist any network services or dataflows between an insecure component like the Internet and a critical component of the company’s intranet like internal server applications. So we developed a method to test a network architecture model for potentially dangerous network services and dataflows making use of automated theorem provers which can be easily integrated into UMLsec. Here, a potentially dangerous network service is understood as a service which is not regulated by a firewall and a potentially dangerous dataflow is understood as a dataflow not scanned for malware. The following exemplary requirements concerning the architecture in which a mobile device can be integrated belong to this group of security requirements:

Requirement 1: Every network service incoming from or outgoing to the Internet must be regulated by a firewall if an internal server application is using this network service.

Requirement 2: All data coming from the Internet must be tested for malware before being received and processed by an internal server application.

Again, one can create a Security Requirements Model formalizing these requirements and an associated Concretized Model that is supposed to describe the implemented security requirements for a given architecture. To be able to verify the Concretized Model for this kind of security requirements, we have developed an approach based on a formalization of the Security Requirements Model and the Concretized Model using first-order logic (FOL). The logical formulas arising from the Concretized Model can then be automatically verified against those arising from the Security Requirements Model using automated theorem provers (ATPs) for first-order logic, such as SPASS[1].

The set of formulas is composed of three main categories, as explained in the following

[1] http://spass.mpi-sb.mpg.de

paragraphs:

Network architecture: The first set of formulas formalizes the network architecture model to be tested. This part formalizes the Concretized Model which has a deployment diagram specifying the network architecture model as its basis. Within this deployment diagram, devices are modeled as nodes which may contain several components (for example software applications). For every node the available access medium is specified, and the nodes can be interconnected by communication links of different types of access media (such as LAN, WLAN, etc.). The components can be labeled with stereotypes describing the type of the component (such as firewall, anti-virus software, encryption software, critical component to be protected, etc). Network services and dataflows are modeled as directed dependencies between components. For each dependency, the protocol (such as HTTP, FTP, IMAP etc.) on which the related network service or dataflow is based is also specified using a stereotype. Additionally, for each component of the type `<<firewall>>` and `<<anti-virus>>` one needs to define which ports or protocol services they can regulate respectively scan. Fig. IV-8 shows such a deployment diagram of a simplified network architecture to be analyzed for the security requirements. Fig. IV-9 shows an exemplary logical formula which defines the component ”I-Server-Application” as a critical component of the node ”Intranet” and ”IW-Application” as component of ”Internal Workstation” with the stereotype `<<others>>`. In adition, this formula specifies that LAN is an active access medium on both nodes (”Intranet” and ”Internal Workstation”), that on this access medium an HTTP network service can be established, and that there exists an HTTP network service between the components ”I-Server-Application” and ”IW-Application”.

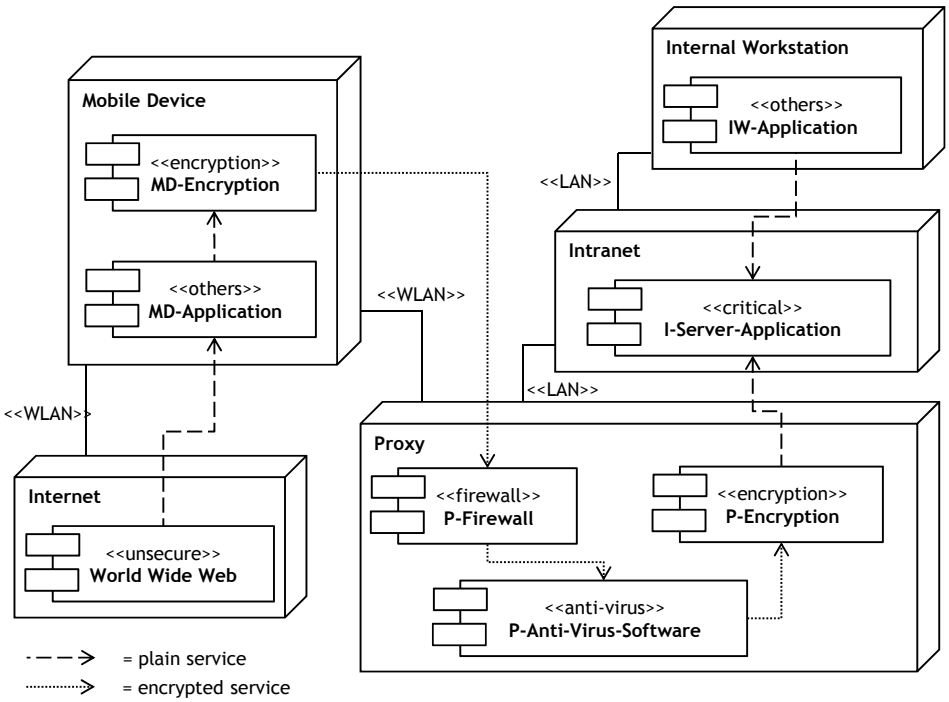

Fig. IV-8: Deployment diagram for Concretized Model

```
input_formula(network_architecture_model,axiom,(
     is_component_of(internal_workstation,iw-application) &
     is_component_of(intranet,i-server-application) &
     type_of_component(iw-application,others) &
     type_of_component(i-server-application,critical) &
     access_media_availability(internal_workstation,lan) &
     access_media_availability(intranet,lan) &
     service_on_access_media(http,lan) &
     connection(service,iw-application,i-server-application,
               http,plaindata))).
```

Fig. IV-9: Formulas for Network Architecture Model

Threat model: The second set of formulas formalizes the threat model against which the Concretized Model should be analyzed. On the one hand these logical formulas provide the rules for generating possible attacks. For example, if the same access medium standard is active on two different nodes, these two nodes can be connected by a communication link of this type of access medium. Furthermore, all kinds of network services and dataflows which can be established on this communication link are added to the network architecture model between the components of these two nodes, in a way that a

network service between two components in one direction always implies a dataflow between these two components in both directions. Fig. IV-10 shows a logical formula which formulates the rule for generating dataflows between two components in both directions from a directed network service between the same two components. More precisely, the formula formalizes that if there exists a network service of any type of protocol and encryption standard between the component `ComponentX` and the component `ComponentY`, there also exist a bidirectional dataflow of the same type of protocol and encryption standard between `ComponentX` and `ComponentY`. On the other hand, the formulas describe the rules for marking all potentially dangerous network services and dataflows as well as for connecting these dangerous services and dataflows transitively.

```
input_formula(connection_of_service_to_connection_of_dataflow,axiom,(
! [ComponentX,ComponentY,Service,DataEnc] : ( (
connection(service,ComponentX,ComponentY,Service,DataEnc) ) => (
connection(dataflow,ComponentX,ComponentY,Service,DataEnc) &
connection(dataflow,ComponentY,ComponentX,Service,DataEnc))))).
```

Fig. IV-10: Formulas for Dataflow Model

Security requirements: The third set of formulas formalizes the security requirements against which the Concretized Model has to be analyzed (i.e., the Security Requirements Model). For every security requirement a particular predicate has to be formulated. Intuitively, a formula formalizing the first security requirement specifies that there exists no transitive network service from an insecure component (in the example in Fig. IV-8, the component ”World Wide Web”) to a critical component (”I-Server-Application”) or vice versa not passing a component of the type `<<firewall>>`. More precisely, because of the way this requirement will be verified by the automated theorem prover (see below), we need to formalize the negation of this requirement (since the requirement is then fulfilled if the prover reports that this formalization is not derivable from the formulas explained above). For example, in the input notation for the FOL theorem provers, the formalization of the first requirement is shown in Fig. IV-11. Intuitively, this formula formalizes the logical conjecture that there exist an insecure component `ComponentX` and a critical component `ComponentY` which are connected by a transitive network service without firewall. If the theorem prover can derive this conjecture from the formulas formalizing the network architecture and the threat model, this

means that there is a potential vulnerability. If the theorem prover reports that such a derivation does not exist, there is no such vulnerability. The predicates such as `type_of_component()` used in this formula have to be defined in the first set of formulas. The second requirement (malware scan) can be formalized by a similar formula.

```
input_formula(requirement_1,conjecture,(
? [ComponentX,ComponentY,Service] : (
connection_without_firewall_regulation
(dataflow,ComponentX,ComponentY,Service,plaindata)&
type_of_component(ComponentX,insecure) &
type_of_component(ComponentY,critical)) )).
```

Fig. IV-11: Formalized Security Properties

To analyze the Concretized Model, the formulas explained above are given as input to the automated theorem prover which then tries to deduce the predicates formalizing the security requirements from the set of formulas describing the architecture model and the rules. This is done completely automatic (i.e. without any human interaction). Also, the formulas can be generated automatically from the Security Requirements Model and the Concretized Model so that the user can use this new verification technique in an automated way as part of the general UMLsec tool-flow (this generation plugin is currently in the development phase). If the theorem prover finds a deduction of one of these predicates, the related security requirement does not hold in the Concretized Model. If the theorem prover reports that no such deduction exists, the security requirement does hold. (In principle, it can happen that the theorem prover does not return a result at all, in which case one cannot draw any conclusions, but this did not happen with the models in the application reported here.)

Assuming that the access medium ”LAN” is active on the nodes ”Mobile Device” and ”Internal Workstation” and an HTTP network service can be established on a LAN communication link, the first security requirement does not hold in the Concretized Model in Fig. IV-8. For example, these two nodes can be connected by a communication link of the type LAN and hence an HTTP network service can be build on the following path not passing a firewall component: ”World Wide Web” → ”MD-Application” → ”IW-Application” → ”I-Server-Application”. Thus there exists a network service between the Internet and an internal server application which is not regulated by a firewall. A solution

could be to integrate a firewall on the device ”Internal Workstation” or to deactivate the access medium ”LAN” on this device.

The second requirement is violated in the Concretized Model when assuming, that ”P-Anti-Virus-Software” is not able to scan encrypted dataflows for malware. Because the dataflow coming from ”MD-Application” over ”MD-Encryption” and ”P-Firewall” is decrypted by ”P-Encryption” after passing ”P-Anti-Virus-Software”, there exists a dataflow from the Internet (”World Wide Web”) to an internal server application without being tested for malware.

5 Lessons Learned

In this section, we provide a discussion of the lessons learned from this application experience at the hand of some guiding questions.

Are there ways in which the application of UMLsec did not go as expected?

Generally, the application of UMLsec worked as expected, though analyzing sophisticated security mechanisms turned out to be less easy than applying simple consistency checks. One main reason is, that depending on the chosen level of abstraction the formalization of security requirements embedded and described in security policies always entails the reduction of real world complexity and thus loosing some of its context. Hence such a formalization of requirements for developing formal security models and analyzing architecture models only produce reasonable results in consideration of well-formed a sophisticated assumptions. Note that most users will not design their own security mechanisms but instead just want to make sure that they use existing mechanisms correctly.

Did the method have to be changed or adapted to work properly, and if so, in what way? Did UMLsec rules have to be changed to fit the application to mobile communication systems with their specific characteristics as discussed in Sec. 2? Were there security mechanisms in the system that UMLsec does not cover?

Of the 62 security requirements that were considered, 34 could be treated using UMLsec rules without having to be changed or adapted specifically to mobile systems. A further 15 were treated with a new verification technique that allows one to analyse a network architecture as to whether a sufficient number of firewall and virus scan nodes are in place. This analysis can be done automatically using automated theorem provers for first-order logic. This extension to UMLsec has been developed and added to UMsec

for this purpose as part of this project (see Sect. 4 for details). Also, all security mechanisms in the system could be analyzed using the UMLsec notation and the extension mentioned above.

Furthermore, there were some recommendations for desirable reasonable improvements of the analysis plugins in the associated UMLsec tool framework. For example, a lot of plugins are only able to test a UML diagram for only one requirement at one time. But often it turned out to be necessary to formalize more than one requirement in one diagram. Hence it would be desirable to add some functionality to these plugins for testing a diagram for many requirements at one time. Due to the amount of repeated checking tasks we thus enhanced the tool to automatically verify a number of arbitrary tags within one activity diagram.

Did the method yield interesting results?

The method showed that the system under consideration is indeed secure with respect to the security requirements and adversary model that were considered, which is certainly interesting giving the high number of insecure or untested systems. In particular, this seems to be the first time UMLsec has been used in a telecommunications environment (in particular for a mobile communication architecture), which in itself is a new and interesting result.

Did it not pick up issues that you would have hoped it would or should?

We are not aware of any issues that the approach did not notify but which were nevertheless present. In particular, the architecture under analysis was found to provide the desired level of security.

How did its use differ from previous uses?

To our knowledge, this was the first application of UMLsec to mobile telecommunications systems. It differs from previous applications in so far as the mobility-specific properties of the application had to be taken into account, which was done rather easily using the adversary definition mechanism built into UMLsec.

Can you say anything specific about the security of the application/system now that you have done the modeling?

Using the UMLsec approach, we were able to precisely demonstrate that the main security requirements central to the mobile communication architecture at hand are actually correctly enforced.

How can you be sure you have applied the method correctly or even optimally? Are there other ways in which you could have applied it? Would you use the UMLsec approach again to this kind of system?

We tried to apply the UMLsec method optimally in the sense that we focused on the security-critical core of the mobile communications system. Since the identification of this part was done informally, there always exists the possibility that security weaknesses may have gone undetected for this reason. However, we believe that this approach is cost-effective in a practical application in industry. In particular, we developed a security assessment process based on UMLsec which seems to be particularly suitable for mobile systems (see Sect. 3.4). Generally, the conclusion was the the use of UMLsec in the given application was successful and it would be worthwhile to apply it again next time to a similar kind of system.

Requirements on Security Analysis

Coming back to the requirements on the security analysis process that were formulated in Sect. 3.1, we can say that each of them has been reached to a satisfactory degree:

Reproducability: The results turned out to be reproducable for a given architecture with apparently little risk of misinterpretation.

Delegability: Analysis tasks could be delegated.

Efficiency: The analysis process was reasonably efficient and could be scaled down by limiting its scope.

Parallelization: It was possible to perform parts of the analysis in parallel and independently.

Traceability: The results of the analysis could be traced to the architecture and guidance derived how negative results can be improved on.

Expressiveness: The results carried enough information to enable an overall risk analysis of a given architecture.

General Discussion

The use of UMLsec was generally appropriate for this case study. We were able to start the security analysis based on the given company security policy and proceed down to the technical details of the security analysis of the mobile communication architecture. Some areas for future improvement of the UMLsec method remain. While the definition

of some simple (e.g., static security requirements) is quite intuitive for the user, the modeling of more sophisticated (e.g., behavioral or cryptography-related) security aspects turned out to be non-trivial for a user who has only a basic background in security. Future improvements on the usability of the method would therefore be useful, although it may never be achievable to give such a method to someone without any kind of prior knowledge in security. Conclusions from the model-based security analysis need to be drawn carefully, since it is based on the assumption that the actual software implementations that are used are themselves secure. Research on linking model-level security analysis to the implementation level is currently under way (Jürjens 2006b).

Although this case-study was not aimed at assessing the usefulness of model-based software development techniques in general, it turned out that such techniques do come with an added overhead with respect to training of the user and with respect to effort and time. One may speculate therefore that uptake of model-based software development in industry be fastest for application domains that involve highly sophisticated and critical requirements, such as security critical systems, since here the effort is most justified (although again, a controlled comparative study on this observation was not part of the scope of the current case-study but would be very interesting future work).

6 Summary

This paper presented a field report on the deployment of the UMLsec method in an industrial context. A model-based security analysis was conducted on a mobile communications system at a major German telecommunications company. The focus was on the application’s security mechanisms and policies. Using the UMLsec notation, the user was able to annotate his models with information regarding the security critical aspects of the system in a concise and clear way. Employing the UML profile of UMLsec, developers familiar with the extension mechanisms of the UML should have no problem to learn UMLsec quickly. Furthermore, by embedding the security analysis directly into the IT development and management process, a better understanding and clearer communication of these issues is made possible.

References (Chapter IV)

Apvrille A, Pourzandi M (2005) Secure software development by example. IEEE Security & Privacy 3(4):10–17

Best B, Jürjens J, Nuseibeh B (2007) Model-based security engineering of distributed information systems using UMLsec. In: Proceedings of the 29th International Conference on Software Engineering (ICSE 2007), ACM, 581–590

Grünbauer J, Hollmann H, Jürjens J, Wimmel G (2003) Modelling and verification of layered security-protocols: A bank application. In: Computer Safety, Reliability, and Security (SAFECOMP 2003), volume 2788 of LNCS, Springer, 116–129.

Jürjens J (2004) Secure Systems Development with UML. Springer, Heidelberg

Jürjens J (2005) Sound methods and effective tools for model-based security engineering with UML. In: Proceedings of the 27th International Conference on Software Engineering (ICSE 2005), IEEE, 322-331

Jürjens J (2006a) Model-based security engineering for real. In: 14th International Symposium on Formal Methods (FM ‘06), volume 4085 of LNCS, Springer, 600–606.

Jürjens J (2006b) Security analysis of crypto-based Java programs using automated theorem provers. In: Easterbrook S, Uchitel S, editors, Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering (ASE ‘06), ACM, 167-176

Jürjens J, Shabalin P (2007) Tools for secure systems development with UML. International Journal on Software Tools for Technology Transfer 9(5):527-544

Perry D, Porter A, Votta L (2000) Empirical studies of software engineering: a roadmap. In: Proceedings of the Conference on The Future of Software Engineering, 345–355

Schalken J (2005) Research methods for the empirical assessment of software processes. In: Proceedings of the 12th Doctoral Consortium at Conference on Advanced Information Systems Engineering (DC-CAiSE ‘05)

UMLsec tool (2008) http://computing-research.open.ac.uk/jj/umlsectool

Vetterling M, Wimmel G, Wisspeintner (2002) A Secure systems development based on the Common Criteria. In: Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering (SIGSOFT ‘02/FSE-10), ACM, 129–138

V Fazit und Ausblick

In diesem Kapitel werden die zentralen Ergebnisse der vorgestellten Beiträge zusammengefasst und Ansatzpunkte für künftigen Forschungsbedarf aufgezeigt.

1 Fazit

Ziel dieser Dissertationsschrift war es, einen Beitrag zum Management systemischer Risiken aus der Perspektive der Wirtschaftsinformatik zu leisten. Dazu wurde am Beispiel der Finanzkrise die Verantwortung der Wirtschaftsinformatik für systemische Risiken in der Finanzwirtschaft diskutiert sowie zukünftige Herausforderungen insbesondere für die IT-basierte Entscheidungsunterstützung abgeleitet (Kapitel II). Darüber hinaus wurde die Softwaredokumentation als präventive Maßnahme gegen die Entstehung von Fehlern und deren Ausbreitung im Rahmen der Softwarewartung einer ökonomischen Analyse unterzogen, eine neue Methodik zur Bewertung von alternativen modularen Softwarearchitekturen unter Berücksichtigung von technischen und fachlichen Abhängigkeiten vorgestellt (Kapitel III), sowie eine neue Methodik zur Prüfung von Rechnernetzwerkarchitekturen auf unsichere Verbindungen und Datenflüsse entwickelt (Kapitel IV).

- Das Ziel von Kapitel II war die auf der Darstellung der Ursachen und Zusammenhänge der Finanzkrise und der ursächlichen Strukturen basierende Diskussion, inwieweit die Wirtschaftsinformatik als „Werkzeugmacher“ für die Entstehung und den Verlauf der Krise verantwortlich ist. Dazu wurden zuerst die Verbriefungstransaktionen als die der Krise ursächlichen Strukturen in einer für Leser ohne Expertenwissen in diesem Fachgebiet geeigneten Form dargestellt. Anhand der Analyse von Ertrags- und Risikopositionen konnten dabei asymmetrische Verteilungen sowie Unzulänglichkeiten in der Risikobewertung und bei der Erstellung der Ratings identifiziert werden. Im zweiten Schritt wurden die Aufgabenfelder der Wirtschaftsinformatik herausgearbeitet, zu welchen unter anderem die Erforschung, Konzeption, Entwicklung, Nutzung und Bewertung von Informations- und Kommunikationssystemen (IKS) im Allgemeinen gehören. Dabei bilden die Entscheidungsunterstützungssysteme (EUS), welche die finanzmathematischen Modelle, auf denen die der Krise ursächlichen Verbriefungsstrukturen basieren, implementieren und welche beispielsweise im Risikomanagement und bei der Ratingerstellung eingesetzt werden, eine spezielle Form solcher IKS. Obgleich die im Rahmen der Krise festgestellten Unzulänglichkeiten des Risikomanagements

und der Ratings primär den finanzmathematischen Modellen zuzuordnen sind, kommt auch der Wirtschaftsinformatik insbesondere als angewandte Wissenschaft und als Interdisziplin zwischen der Betriebswirtschaft und der Informatik zumindest eine entsprechende Mitverantwortung zu. Denn es sind die von Wirtschaftsinformatikern konzipierten EUS, die auf Basis der finanzmathematischen Modelle Informationen und Handlungsempfehlungen unter anderem für das Risikomanagement von Finanzdienstleistern generieren und bereitstellen. Aus dieser Mitverantwortung wurden schließlich Lehren und zukünftige Herausforderungen insbesondere für die IT-basierte Entscheidungsunterstützung abgeleitet, welche es für die Vermeidung oder zumindest für die Verminderung der Auswirkungen von systemischen Risiken in der Finanzwirtschaft zu meistern gilt.

- Gegenstand des Kapitel III waren systemische Risiken im Rahmen des Managements von Anwendungssystemen und insbesondere die Ausbreitung von Fehlern beziehungsweise Änderungen zwischen den Komponenten einer Software über deren gegenseitige Abhängigkeiten als spezielle Form eines systemischen Risikos. Dabei beschäftigte sich der erste Beitrag mit der Dokumentation von Softwaresystemen als eine mögliche Maßnahme zur Reduzierung der Fehlerentstehung und Fehlerausbreitung während der Wartung. Sowohl in der Wissenschaft als auch in der Praxis wurden die technische Programmdokumentation und ihre Auswirkungen auf die Fehleranfälligkeit von Wartungsmaßnahmen bisher stark vernachlässigt. Vor diesem Hintergrund wurde ein formal-deduktiver Ansatz vorgestellt, welcher die Generierung eines neuen Zusammenhangs zwischen dem Dokumentationsgrad einer Software und der im Rahmen der Softwarewartung entstehenden Fehler ermöglichte. Darauf aufbauend wurden die barwertigen Auszahlungen für die Dokumentation während der Softwareentwicklung den barwertigen Auszahlungen für die Fehlerbehebung während der Wartung gegenübergestellt und der auszahlungsminimale Dokumentationsgrad ermittelt. Dies erlaubte schließlich eine Analyse der Auswirkungen einer Abweichung vom optimalen Dokumentationsgrad und die Deduktion einer weiteren Hypothese: Eine positive Abweichung vom optimalen Dokumentationsgrad ist stets vorteilhaft gegenüber einer betragsmäßig gleichen, negativen Abweichung. Diese Hypothese ist dabei für alle zulässigen Werte der gegebenen Parameter gültig und erweist sich auch gegenüber der Änderung einiger Annahmen als robust. Für die Verminderung von systemischen Risiken im Sinne einer Fehlerausbreitung aufgrund von nachträglichen Änderungen kann also die Maxime abgeleitet werden, dass

während der Softwareentwicklung selbst bei einer beliebig niedrigen Dokumentationsqualität, einem beliebig hohen Kalkulationszinssatz oder bei beliebig hohen Auszahlungen für die Dokumentation eher zu viel als vergleichsweise zu wenig dokumentiert werden sollte. Im Fokus des zweiten Beitrags stand dagegen die Entwicklung einer modularen Softwarearchitektur mit einer möglichst losen Kopplung der Module, welche vorwiegend von der existierenden Literatur als weitere präventive Maßnahme zur Reduzierung des Ausbreitungseffekts von Änderungen (*change propagation*) empfohlen wird. Jedoch besitzen nicht nur technische Abhängigkeiten im Sinne der Modulkopplung einen Einfluss auf die Auszahlungen für die Wartung, sondern auch fachliche Abhängigkeiten und damit die Anzahl der Module, deren Modifikation für die Anpassung eines Softwaresystems an eine geänderte Anforderung erforderlich ist. Da die Literatur fachliche Abhängigkeiten zwischen Softwarekomponenten bisher stark vernachlässigte, stellte der Beitrag eine Methodik vor, die eine ökonomische Bewertung und einen Vergleich von alternativen Softwarearchitekturen in Bezug auf die barwertigen Auszahlungen für fachlich getriebene Änderungen im Zuge der Wartung unter Berücksichtigung von sowohl technischen als auch fachlichen Abhängigkeiten erlaubt. Dabei wurde gezeigt, dass eine Softwarearchitektur mit möglichst loser Modulkopplung trotz des Ausbreitungseffekts von Änderungen theoretisch zu höheren Auszahlungen im Rahmen der Wartung führen kann als eine alternative Architektur, welche mit der Konsequenz einer stärkeren Kopplung die fachlichen Abhängigkeiten zwischen Modulen reduziert.

- Kapitel IV befasste sich schließlich mit systemischen Risiken in Rechnernetzen und den damit verbundenen Herausforderungen im Rahmen der IT-Sicherheit. Insbesondere bei der Konzeption von IT-Architekturen mit UML bietet sich für die Überprüfung der entwickelten Architekturmodelle, inwieweit diese die gestellten Sicherheitsanforderungen erfüllen, die Verwendung der UML-Erweiterung *UMLsec* an. Im Zuge einer modell-basierten Sicherheitsanalyse eines Teils der mobilen Kommunikationssysteme von O_2 (Germany) GmbH & Co. OHG stellte sich unter anderem die in der UMLsec-Notation und den zugehörigen Werkzeugen fehlende Möglichkeit für die Überprüfung von Netzwerkarchitekturen auf unsichere Verbindungen und Datenflüsse heraus. Wie im einleitenden Kapitel beschrieben, können gerade derartige Schwachstellen den Ursprung für systemische Risiken in Rechnernetzen bilden. Daher wurde eine neue Methodik entwickelt, die eine automatische Analyse von Netzwerkarchitekturen auf (potenzielle) Verbindungen ohne

Regulierung durch eine Firewall und auf (potenzielle) Datenflüsse ohne Überprüfung durch einen Virenscanner ermöglicht. Diese Methodik basiert auf prädikatenlogischen Ausdrücken erster Ordnung und bildet die Grundlage für eine entsprechende Erweiterung von UMLsec.

Abschließend lässt sich festhalten, dass die vorliegende Arbeit die Herausforderungen im Rahmen des Managements von systemischen Risiken aus drei unterschiedlichen Perspektiven der Wirtschaftsinformatik adressiert und einen Beitrag zur Lösung ausgewählter Problemstellungen wie zum Beispiel die Bestimmung eines ökonomisch sinnvollen Dokumentationsgrads von Anwendungssystemen im Rahmen der Softwareentwicklung oder die Entwicklung einer Methodik zur Analyse von Rechnernetzwerken auf unsichere Verbindungen und Datenflüsse anbietet. In diesem Kontext existieren jedoch noch viele weitere Fragestellungen, die im Rahmen zukünftiger Forschungsarbeiten entsprechender Aufmerksamkeit bedürfen.

2 Ausblick

Aus den untersuchten Themenbereichen dieser Arbeit ergibt sich eine Reihe weiterführender Fragestellungen, die Ansatzpunkte für zukünftigen Forschungsbedarf darstellen:

- Im Bereich der in Kapitel II vorgestellten Beiträge zur Mitverantwortung der Wirtschaftsinformatik für die Finanzkrise und zu den zukünftigen Herausforderungen insbesondere für die IT-basierte Entscheidungsunterstützung existiert unter anderem hinsichtlich der folgenden Punkte Forschungsbedarf:

  1. Als ein wesentliches Problem bei der Verwendung von entsprechenden EUS wurde im Rahmen der Finanzkrise eine teils blinde Modellgläubigkeit der Entscheidungsträger und Nutzer der EUS herausgearbeitet. Folglich wurde als zukünftige Herausforderung für die IT-basierte Entscheidungsunterstützung die Entwicklung von Theorien, Methoden und Konzepte für eine Gestaltung von EUS zur Reduktion einer blinden Modellgläubigkeit gefordert. An eben dieser Stelle müssen nun zukünftige Forschungsarbeiten ansetzen.
  2. Darüber hinaus erscheinen die Untersuchung und gegebenenfalls die Entwicklung von Methoden zur Evaluation solcher EUS und den darin umgesetzten Modellen unter Realweltbedingungen lohnenswert. Grundsätzlich besteht bereits heute die Möglichkeit, Erkenntnisse für die Gestaltung und Verbesserung von EUS aus den Erfahrungen des realen Einsatzes solcher Systeme zu gewinnen. In

diesem Fall muss jedoch zumindest das einmalige Eintreten eines bis dato unbekannten Risikos sozusagen als Lernerfahrung in Kauf genommen werden. Mit Bezug auf die Finanzwirtschaft formuliert Rudolph (2008) aber den Anspruch, dass sich das Risikomanagement weniger auf die rechenbaren Anteile verlassen darf, sondern sich vermehrt mit Szenarien beschäftigen muss, welche bislang unbekannte Risiken betreffen. Analog dazu scheint auch die Herausforderung für die Wirtschaftsinformatik, bisher unbekannte Formen des fehlerhaften Gebrauchs oder sogar des Missbrauchs für die Gestaltung von EUS zu antizipieren, als lohnenswerter Ansatzpunkt für die zukünftige Forschung.

- Auf der Basis der in Kapitel III vorgestellten Modelle ergibt sich unter anderem hinsichtlich folgender Punkte zukünftiger Forschungsbedarf:

  1. Das im ersten Beitrag vorgestellte Modell und die daraus abgeleiteten Hypothesen basieren auf zum Teil stark vereinfachenden Annahmen. So werden beispielsweise nur die Auswirkungen der technischen Programmdokumentation auf die Fehleranfälligkeit von Änderungen existierender Komponenten betrachtet und Erweiterungen eines Softwaresystems um neue Komponenten im Rahmen der Softwarewartung vernachlässigt. Darüber hinaus berücksichtigt das Modell ausschließlich die Auswirkungen der Dokumentation von Abhängigkeiten auf Fehler, die sich über diese Abhängigkeiten auf angrenzende Komponenten ausbreiten. Nicht betrachtet werden dagegen die Dokumentation von Code-Fragmenten ohne Abhängigkeiten zu anderen Fragmenten sowie alle Fehler, die nicht dem beschriebenen Ausbreitungseffekt unterliegen. Zukünftige Forschungsarbeiten müssen sich daher im Rahmen entsprechender Modellerweiterungen um eine Auflösung der gegenüber der Realität vereinfachenden Annahmen bemühen. Diese können dabei einerseits überprüfen, inwieweit sich die im Rahmen dieser Arbeit deduzierten Hypothesen ebenfalls nach einer entsprechenden Auflösung der Annahmen ableiten lassen, und andererseits die Deduktion weiterer Hypothesen verfolgen.

  2. Die Bestimmung eines ökonomisch sinnvollen Investitionsvolumens in eine zweckmäßige technische Programmdokumentation von Softwaresystemen sollte mitnichten nur für ein isoliert betrachtetes Projekt erfolgen. Da Softwareentwickler für Unternehmen oftmals eine „knappe Ressource“ darstellen und mit der Erstellung zusätzlicher Programmdokumentation entsprechend länger an ein Softwareprojekt gebunden sein dürften, sollten entsprechende Modelle ebenfalls die Frage nach dem ökonomisch sinnvollen Zeitpunkt für den Beginn von Folgeprojekten

adressieren.

3. Die im ersten Beitrag vorgestellten Zusammenhänge dürften auch einen Einfluss auf zahlreiche, weiterführende Fragestellungen mit Bezug auf die Softwarewartung und deren Fehleranfälligkeit besitzen. So könnten sich zukünftige Forschungsarbeiten auf Basis des präsentierten Modells beispielsweise mit der Frage nach der ökonomischen Vorteilhaftigkeit des Outsourcings der Softwarewartung oder nach dem ökonomisch sinnvollen Zeitpunkt für den Ersatz eines Altsystems durch eine neue Software beschäftigen.
4. Auch das im zweiten Beitrag vorgestellte Modell basiert auf teilweise stark vereinfachenden Annahmen. So berücksichtigt die Bewertungsmethodik ausschließlich die nachträgliche Änderung existierender Module und vernachlässigt die Erweiterung einer Software um neue Module. Da eine Erweiterung eines Softwaresystems ebenfalls die Einführung neuer Abhängigkeiten zwischen den Modulen bedeutet, kann ein Vergleich von alternativen Softwarearchitekturen in diesem Fall zu grundlegend anderen Ergebnissen führen. Daher müssen sich zukünftige Forschungsarbeiten um eine entsprechende Erweiterung des vorgestellten Modells bemühen.
5. Insbesondere empirische Forschungsarbeiten können sich einer kritischen Prüfung der in beiden Beiträgen generierten Hypothesen widmen und analysieren, inwieweit diese Hypothesen die entsprechenden Phänomene in der Realität prognostizieren können. Durch eine empirische Bestätigung der im Rahmen dieser Arbeit vorgestellten Hypothesen kann die Grundlage für die Generierung normativer Aussagen und für die Ableitung von Handlungsempfehlungen für die Praxis weiter fundiert werden.

- Insbesondere die in Kapitel IV vorgestellte Methodik zur automatischen Analyse von Netzwerkarchitekturen auf unsichere Verbindungen und Datenflüsse bietet Ansatzpunkte für weitere Forschungsaktivitäten:

  1. Im Konkreten überprüft die Methodik, ob eine (potenzielle) Verbindung zwischen einem unsicheren und einem zu schützenden Knoten des Netzwerks existiert, welche nicht eine Komponente des Typs „Firewall“ passiert. Analog dazu ermöglicht die Methodik eine Analyse, ob ein Datenfluss zwischen einem unsicheren und einem zu schützenden Knoten existieren kann, welcher nicht eine Komponente des Typs „Virenscanner“ passiert. Dabei muss für jede Komponente des Typs „Firewall“ oder „Virenscanner“ angegeben werden, welche Verbindungen

beziehungsweise Datenflüsse auf welchen Ports beziehungsweise auf Basis welcher Protokolle grundsätzlich reguliert respektive überprüft werden können. Die Methodik erlaubt jedoch keine Analyse, inwieweit eine Firewall beziehungsweise ein Virenscanner eine bestimmte Verbindung beziehungsweise einen bestimmten Datenfluss dann auch tatsächlich ordnungsgemäß und fehlerfrei reguliert respektive überprüft. Daher können sich zukünftige Forschungsarbeiten eine entsprechende Erweiterung der vorgestellten Methodik zum Ziel setzen.

2. Da die Methodik zur Überprüfung von Netzwerkarchitekturen im ersten Schritt unabhängig von den bis dato existierenden UMLsec-Werkzeugen entwickelt wurde, sollten zukünftige Forschungsarbeiten die Einbettung der Methodik in die existierenden UMLsec-Werkzeuge verfolgen. Dazu muss unter anderem ein Verfahren entwickelt werden, welches ein Modell einer Netzwerkarchitektur und der zugehörigen Sicherheitsanforderungen beispielsweise in Form eines UML-Komponentendiagramms automatisch in die entsprechenden prädikatenlogischen Ausdrücke überführt. Diese können dann wiederum unter Verwendung eines geeigneten Theorembeweisers automatisch analysiert werden.

In dieser Dissertationsschrift wurden nur einzelne Aspekte des Managements systemischer Risiken aus der Perspektive der Wirtschaftsinformatik vertieft behandelt. Folglich gilt es künftig, die Bewertung und Steuerung systemischer Risiken in der Wirtschaftsinformatik nicht nur für die im Rahmen dieser Arbeit untersuchten, sondern auch für weitere Aspekte voranzutreiben. Hierfür können die vorgestellten Beiträge zwar einen Ausgangspunkt darstellen. Allerdings birgt eine umfassende Untersuchung systemischer Risiken im Rahmen der Wirtschaftsinformatik sowie der Herausforderungen für deren Management zahlreiche zusätzliche Ansatzpunkte für künftige Forschungsaktivitäten.

Literatur (Kapitel V)

Rudolph B (2008) Lehren aus den Ursachen und dem Verlauf der internationalen Finanzkrise. Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung 60: 713–741