Risiko

Manuskripteingang: 8.7.2005
Angenommen am: 10.4.2006

Uwe Aßmann, Birgit Demuth und Falk Hartmann

Risiken in der Softwareentwicklung

Die Kosten der Softwareproduktion sind für viele Firmen von entscheidender Bedeutung, da Software in einer ständig wachsenden Anzahl von Firmenprozessen und Produkten eingesetzt wird und immer mehr Anwendungsbereiche durchdringt. Allerdings birgt die qualitäts-, zeit- und kostengerechte Fertigstellung von Softwareprojekten enorme Risiken. Dieses Papier nennt spezifische Risiken der Softwareentwicklung anhand von fehlgeschlagenen Projekten, skizziert die Möglichkeiten der Risikovermeidung in der Softwareentwicklung und die Behandlung des Risikomanagements im Prozess der Softwareentwicklung. Der zunehmende Einsatz von Software in eingebetteten Systemen begründet die Notwendigkeit des Wissens um Risiken der Softwareentwicklung auch in den klassischen Ingenieurdisziplinen.

The overall costs of software production are of crucial importance to many companies, as software is involved in a constantly increasing number of business processes and products. On the other hand, the completion of software projects in accordance with defined quality, time and cost requirements involves a high level risk. This paper enumerates specific risks within software development, outlines possibilities for risk prevention and illustrates the treatment of risks within the software development process. The increasing application of software within embedded systems also founds a necessity of knowledge concerning the risks in software development for the established engineering disciplines.

1 Softwareentwicklung als Ingenieurdisziplin

Die Softwareentwicklung hat sich in der zweiten Hälfte des letzten Jahrhunderts schrittweise zur Ingenieurdisziplin entwickelt, für die auf der legendären NATO-Konferenz 1968 [6] der Begriff „Software Engineering“ geprägt wurde. Im Vergleich zu klassischen Ingenieurdisziplinen wie zum Beispiel Bauwesen oder Maschinenbau ist die Softwareentwicklung damit eine sehr junge Ingenieurwissenschaft. Die Erfahrung zeigt, dass Ingenieurwissenschaften Zeit brauchen, um sich zu entwickeln und zu verbreiten. Auf der anderen Seite sind die Kosten der Softwareentwicklung für viele Firmen von entscheidender Bedeutung, da Software in einer ständig wachsenden Anzahl von Firmenprozessen und Produkten eingesetzt wird und immer mehr Anwendungsbereiche durchdringt. Es verwundert deshalb nicht, wenn die qualitäts-, zeit- und kostengerechte Fertigstellung von Softwareprojekten große Risiken birgt. Dieses Phänomen wird gut durch den Begriff der „Softwarekrise“ charakterisiert, der ebenfalls 1968 geprägt wurde, aber bis heute nichts an seiner Aktualität eingebüßt hat. Die Situation ist allerdings nicht ganz so schlimm, wie sie oberflächlich erscheint. Obwohl die Komplexität der Software einschließlich der Anforderungen an Qualität und Sicherheit stark wächst, wurden dennoch viele dieser technischen Probleme durch neue Technologien, Methoden, Werkzeuge, Standardsoftware und Betriebssysteme gelöst. Ein Brückenbauer kann es sich nicht leisten, dass seine Brücke einstürzt. Um dieses Risiko zu vermeiden, greift er auf lang erprobtes Wissen zurück. Dagegen nehmen viele „Softwareingenieure“ billigend in Kauf, dass neue Produkte in vielen Fällen eine Reifezeit beim Kunden brauchen, bevor sie verlässlich und den Anforderungen angemessen arbeiten. Diese Vorgehensweise hat ihre Ursachen nicht nur im Mangel an ausgereiften Werkzeugen und erprobten Technologien, sondern auch in den Schwierigkeiten der Beherrschung eines komplexen Entwicklungsprozesses und der nicht immer gegebenen Verfügbarkeit qualifizierter Softwareentwickler.

Der Artikel beschäftigt sich mit den Risiken der Softwareentwicklung, klassifiziert diese, illustriert Risiken anhand von fehlgeschlagenen oder in die Kritik geratenen Projekten und skizziert die Möglichkeiten der Risikovermeidung sowie des Risikomanagements im Prozess der Softwareentwicklung.

2 Klassifikation von Risiken

Risiken liegen in jeder Art von Ingenieurtätigkeit. Neben allgemeinen Risiken wie einem schlechten Projekt- und Firmenmanagement gibt es in der Softwareentwicklung spezifische Risiken, wie zum Beispiel die Verwendung unangepasster CASE[1]-Werkzeuge oder die ungenügende Spezifikation von Schnittstellen. Im Folgenden werden sowohl allgemeine als auch spezifische Risiken in der Softwareentwicklung klassifiziert und kurz beschrieben.

[1] CASE: Computer Aided Software Engineering

Risiken können unter verschiedenen Gesichtspunkten betrachtet werden. Wir unterscheiden

- die Ursache des Risikos, d. h. warum läuft etwas nicht wie erwartet,
- die Wahrscheinlichkeit des Eintretens des Risikos,
- das Ausmaß eines Risikos, d. h. wer ist vom Risiko betroffen und
- die Stärke der Auswirkung des Risikos.

Bild 1 zeigt typische Ausprägungen dieser Risikokategorien.

Aufgabe des Projektmanagements ist es zunächst, Risiken eines Softwareprojektes zu analysieren. Die Erkennung und Beurteilung von Risiken hängt im Wesentlichen vom Urteilsvermögen und der Erfahrung der verantwortlichen Personen ab und ist damit subjektiv geprägt. Im Normalfall gibt es in einem Softwareprojekt nicht ein einzelnes, sondern eine Vielzahl von Risiken. Ein einzelnes Risiko wird, basierend auf obiger Klassifikation, durch eine Kombination von Ausprägungen der vier Risikokategorien beschrieben. In der Praxis des Risikomanagements werden Risiken in einer Tabelle, wie zum Beispiel Tabelle 1, aufgelistet.

Natürlich ist es auch möglich, in dieser Tabelle die Auswirkungen und die Wahrscheinlichkeiten quantitativ zu erfassen. Das Maß, welches dann zur laufenden Überwachung/Behandlung des Risikos herangezogen werden sollte, ist die so genannte Risikogefährlichkeit, d. h. das Produkt aus Risikoauswirkung und Eintrittswahrscheinlichkeit. Dieses Maß zeigt den interessanten Effekt, dass nicht unbedingt die Risiken mit der höchsten Eintrittswahrscheinlichkeit bzw. der größten Auswirkung die gefährlichsten sein müssen (vgl. [1]).

Bild 1. Ausprägungen der Risikokategorien

3 Softwareentwicklung als iterativer Prozess

Im Zuge der zunehmenden Anforderungen an die Softwareentwicklung, insbesondere im Hinblick auf die massenhafte und schnelle Produktion und Weiterentwicklung von Software, wurde seit den späten sechziger Jahren eine Vielfalt so genannter Vorgehensmodelle entwickelt. Diese beschreiben die bei der Entwicklung von Software durchzuführenden Schritte sowie ihre Abhängigkeiten und treffen Aussagen über die benötigten Ressourcen. Dabei fand ein Paradigmenwechsel von nicht-iterativen („Wasserfallmodellen“) zu iterativen Prozessen („Spiralmodelle“) statt. Dieser wurde in erster Linie getrieben durch die Erkenntnis, dass ein einmaliges Durchlaufen eines Prozesses für die Softwareproduktion in der Praxis durch häufig auftretende Änderungen oder Präzisierungen der Kundenwünsche bzw. Markterfordernisse gestört wird. Die heute üblichen iterativen Prozesse behandeln derartige Anforderungsänderungen in der jeweils nächsten Iteration.

Ein typischer Softwareentwicklungsprozess besteht aus den Phasen Anforderungsanalyse, Softwareentwurf, Implementierung und Test. Häufig werden auch spätere Phasen wie Installation und Wartung als Teile des Entwicklungsprozesses betrachtet. Iterative Prozesse bieten gute Voraus-

Tabelle 1

Ausmaß	Ursache	Auswirkung	Wahrscheinlichkeit
Projekt	Komplexität: wird unterschätzt	ernst	hoch
Projekt	Anforderungsmanagement: ungenügende Anforderungsanalyse	ernst	hoch
Unternehmen	Personal: nur teilweise erfahrene Softwareentwickler	tolerierbar	mittel

setzungen sowohl für die Risikoerkennung als auch für die Risikobehandlung. Zum einen besteht die Möglichkeit der Überprüfung aller Zwischenergebnisse des Prozesses auf Kongruenz mit den Anforderungen, wodurch das häufig auftretende Risiko der Änderung in diesem Bereich gut beherrschbar ist. Dies gilt insbesondere, wenn der Softwareentwicklungsprozess zeitig die Entwicklung so genannter Prototypen vorsieht, welche Zwischenprodukte darstellen, die zugleich der Überprüfung der technischen Machbarkeit, der besseren Abschätzbarkeit der Komplexität und der Demonstration der Ergebnisse zum Beispiel beim Kunden dienen können.

Die iterativen Vorgehensmodelle führen andererseits allerdings zu einer schlechteren Abschätzbarkeit des aktuellen Projektfortschritts. Im Unterschied zu nicht-iterativen Prozessen, bei denen eine Phase abgeschlossen wird, bevor die nächste begonnen wird, ist die Verschiebung von Aufgaben in spätere Iterationen beim iterativen Ansatz eine gängige und unvermeidliche Technik, durch die neue Risiken eingeführt werden. Unter anderem ist es nicht mehr möglich, den Projektfortschritt einfach aufgrund des Fortschritts innerhalb der aktuellen Phase abzuschätzen. Stattdessen müssen Schätzungen des verbliebenen Aufwandes für alle Phasen durchgeführt werden – verbunden mit dem Risiko, Fehlabschätzungen zu begehen. Hinzu kommt die Gefahr, dass eine Verschleppung nicht-adressierter Probleme in späteren Iterationen den Projekterfolg generell gefährdet. Iterative Prozesse bergen diese Gefahr auf mehreren Ebenen: So kann sowohl die unzulässige Verzögerung technischer Entscheidungen als auch der mangelnde Arbeitsfortschritt Einzelner wissentlich oder unwissentlich verschleiert werden.

4 Risikomanagement in Softwareprozessen

Eine systematische Integration des Risikomanagements in ein iteratives Vorgehensmodell kann auf zwei verschiedene Arten erfolgen. Einerseits kann in jede Iteration eine spezielle Phase integriert werden, in der die Risikoanalyse und -behandlung durchgeführt wird. Andererseits kann ein parallel laufendes Risikomanagement etabliert werden. Beide Ansätze sowie diverse Mischformen sind verbreitet.

Das bekannteste Vorgehensmodell mit einer speziellen Phase zum Risikomanagement ist das Spiralmodell nach BOEHM [2] (vgl. Bild 2). Es wird aufgrund der Bedeutung, die die Risiko-Analyse in diesem Modell einnimmt, auch als risiko-getriebener Prozess bezeichnet.

Im Gegensatz zum Spiralmodell stellt das Continuous Risk Management Paradigma [4] des Instituts für Softwareengineering der Carnegie Mellon Universität in Pittsburgh einen eigenständigen Prozess dar, der unabhängig von der konkreten Phase bzw. Iteration im Softwareentwicklungsprozess permanent ausgeführt wird (vgl. Bild 3).

Beide Ansätze zur Integration des Risikomanagements beinhalten die wesentlichen Bestandteile der Risikobehandlung: die Identifikation, Analyse und Bekämpfung von Risiken, letztere bestehend aus der Planung von Gegenmaßnahmen und der Überprüfung ihrer Wirksamkeit. Orthogonal zu diesen Einzelmaßnahmen ist die Wichtigkeit der Kommunikation zwischen allen Prozessbeteiligten zu betonen. Risikomanagement setzt eine klare und gleichzeitig offene Kommunikationsstruktur voraus, um einerseits jeden Mitarbeiter in die Lage zu versetzen, erkannte Risiken weiterzuleiten und andererseits die Nichtbehandlung erkannter Risiken für alle erkennbar zu machen.

5 Beispiele

Es existieren verschiedene Studien über fehlgeschlagene Projekte und fehlerhafte Systeme. Die Internetseite www.risks.org [8] ist des Weiteren eine Fundgrube für Beispiele. Insgesamt wird jedoch wenig über problematische Projekte und die genauen Ursachen des Scheiterns bzw. der teilweisen Nichterfüllung berichtet, insbesondere aus geschäftspolitischen Gründen. Auffällig ist aber die Tatsache, dass besonders IT-Großprojekte des öffentlichen Bereiches viele Misserfolge vorzuweisen haben. Das betrifft nicht nur die US-Bundesverwaltung [9], sondern wird auch an aktuellen Beispielen der deutschen Behörden deutlich [5]. Im Folgenden charakterisieren wir kurz einige dieser deutschen Projekte. Es wird dabei auf die Wiederholung von unsicheren – weil nicht verifizierbaren – Aussagen, wie sie zu allen genannten Projekten in den verschiedensten Quellen ([5, 8, 9] usw.) zu finden sind, verzichtet.

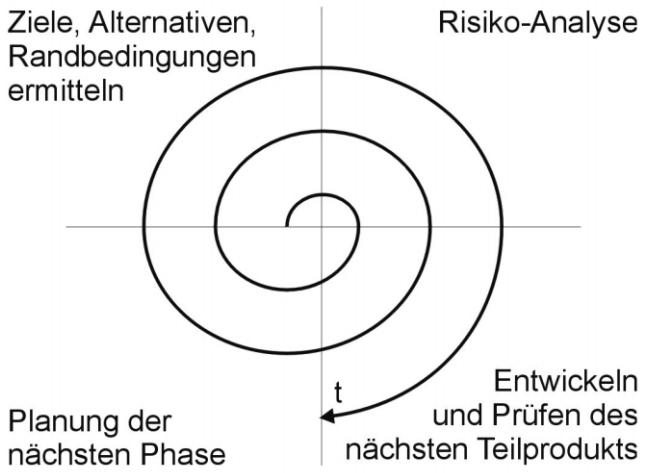

Bild 2. Spiralmodell nach BOEHM [2]

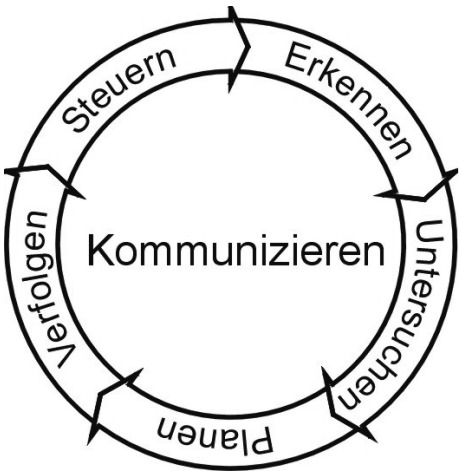

Bild 3. Continuous Risk Management Paradigma der CMU [4]

5.1 Tollcollect

Das am 1. Januar 2005 in Deutschland eingeführte System zur Erfassung einer streckenbezogenen Maut auf den ca. 12 000 km Autobahn kann auf eine bewegte Geschichte zurückblicken. Bereits 1994 wurden erstmalig Ideen zur automatischen Mauterfassung basierend auf dem satellitengestützten Ortungssystem GPS und GSM-basierten Mobilfunknetzen auf einem Pressekolloquium präsentiert. Im Jahre 2002 wurde ein Konsortium aus der Deutschen Telekom, DaimlerChrysler und der französischen Cofiroute nach einer Ausschreibung mit der Realisierung eines solchen Systems, das im Wesentlichen auf derselben Architektur basiert, beauftragt.

Der Verlauf des Projektes, insbesondere bei Annäherung an das vorgesehene Projektende, muss Außenstehenden zuweilen wie ein Lehrstück in Sachen Risikomanagement erscheinen. Nachdem noch kurz vor der geplanten Inbetriebnahme am 31. August 2003 versichert wurde, dass der Projektfortschritt den Erwartungen entspräche, wurde das System erst am 1. Januar 2005 mit eingeschränkter Funktionalität in Betrieb genommen.

Auch wenn ein Teil der Probleme schlichtweg der erheblichen Komplexität des ambitionierten Systems zuzuschreiben ist, traten auch Probleme bei der Softwareentwicklung zu Tage, die zum Teil sicherlich vermeidbar gewesen wären. Das wird an Verspätungen der Auslieferung der Geräte zur automatischen Mautabrechnung in den LKWs (On Board Unit, OBU) sichtbar. So stellte sich zum Beispiel die automatische Software-Nachladefunktion (Update-Funktion) der OBUs als fehlerhaft heraus. Die Umstände deuten darauf hin, dass man die Risiken des Integrationstests unterschätzt hatte. Gerade bei komplexen Projekten birgt das Zusammenschalten der Komponenten enorme Risiken; Verteilung, realistische Last und unerwartetes Benutzerverhalten erzeugen unvorhergesehene Situationen, die neu konfigurierte Systeme zusammenbrechen lassen können. Sicher wäre auch der Einsatz eines iterativen Vorgehensmodells mit Prototypen angebracht gewesen. Gerade das Spiralmodell ist für solche großen Projekte konzipiert.

Die Konsequenzen für die an Tollcollect beteiligten Unternehmen waren ernst und wären sicherlich existenzbedrohend gewesen, hätte nicht die Vertragsgestaltung an dieser Stelle einseitig schadensbegrenzend gewirkt. Mithin muss dem Konsortium ein gewisses Maß an Risikomanagement bei der Ausarbeitung des Vertragswerkes anerkannt werden. Leider wurden die Möglichkeiten des Risikomanagements bei der Softwareentwicklung nicht hinreichend genutzt, was in einem erheblichen Image-Schaden für die beteiligten Unternehmen resultierte. Und auch der Bund betrieb kein hinreichendes Risikomanagement: Die Steuerausfälle führten zur Verzögerung weiterer Projekte.

5.2 A2ll

Die in Deutschland kontrovers diskutierte Hartz IV-Reform und die damit verbundene Gewährung des Arbeitslosengeldes II ab 1. Januar 2005 hat ihr Software-Pendant in der webbasierten Anwendung mit dem Namen A2ll gefunden. A2ll gilt derzeit mit bis zu 40 000 gleichzeitigen Nutzern (Sachbearbeiter der bisherigen Sozialämter und des Arbeitsamtes) als die größte E-Government-Anwendung Europas. Die für den 4. Oktober 2004 geplante Software zur Datenerfassung und -bearbeitung von ca. 2,5 Millionen Anträgen konnte im größeren Stil erst mit drei Wochen Verspätung und nur mit gravierenden Mängeln an den Start gehen. In den Medien sorgte vor allem ein Fehler in den Überweisungen des Arbeitslosengeldes an die Empfänger für Aufruhr. Kontonummern, die nicht 10-stellig sind, wurden rechtsbündig statt linksbündig aufgefüllt, wodurch Überweisungen den Empfängern nicht zugeordnet werden konnten und dadurch eine Verzögerung der Zahlung des Arbeitslosengeldes eintrat. Da dies Menschen trifft, für die die rechtzeitige Zahlung eines jeden Euro von existenzieller Bedeutung sein kann, hat dieses Softwareproblem zum Imageschaden der Regierung beigetragen. Des Weiteren geriet, wie in [3] berichtet, die Arbeit mit A2ll für die Sachbearbeiter zu einem „wahren Alptraum“.

Rückblickend lagen die Ursachen für das A2ll-Desaster in einer absolut unrealistischen Zeitplanung für das Projekt (bedingt durch politische Anforderungen, Managementprobleme und unzureichende Qualitätssicherung) sowie der unterschätzten Komplexität des Produktes (Lastanforderungen einer webbasierten Software). Dass aber für die erwähnten, immer wieder auftretenden Programmierfehler, für die fehlerhafte Zeitplanung und für die mangelnde Benutzbarkeit des Systems keine Risikobehandlung vorgesehen wurde, ist als ein grober Fehler des Risikomanagements zu werten.

5.3 ELSTER

Seit Anfang des Jahres 2005 sind Unternehmen verpflichtet, ihre Steuererklärung auf elektronischem Weg einzureichen. Diese Übermittlung der elektronischen Steuererklärung („ELSTER“) sorgte für mehrere Schlagzeilen. Zum einen wurde die mangelnde Plattformunterstützung des Programms kritisiert (es gibt nur eine Windows-Variante). Andererseits wurden Sicherheitsmängel gemeldet.

So stellte sich heraus, dass für die Übermittlung einer Steuererklärung keinerlei Authentifizierung erforderlich ist – die Angabe von Namen und Steuernummer (beides öffentlich verfügbare Daten) ist ausreichend. Die auf diese Art erreichbare Verletzung der Integrität der Daten kann für betroffene Unternehmen zumindest Unannehmlichkeiten zur Folge haben.

ELSTER ist vermutlich ein gutes Beispiel für eine Fehleinschätzung der Auswirkungen, die sich aus der Nichtbehandlung eines Risikos ergeben. Es ist nicht anzunehmen, dass das Authentifizierungsproblem den Verantwortlichen unbekannt war, stattdessen dürften die Folgen – sowohl für die Benutzer der Software als auch durch den Imageschaden für die öffentlichen Institutionen – unterschätzt worden sein. Auch wurden keinerlei kompensierende Maßnahmen für den Fall des Fehlschlagens vorgesehen. Schließlich hätte man die Anwendung leicht als erweiterbar entwerfen können, was offensichtlich unterblieben ist.

5.4 Fiscus

Ein trauriges Beispiel für ein endgültig fehlgeschlagenes IT-Großprojekt des Bundes und der Länder, welches den Steuerzahler nach 13 Jahren Projektlaufzeit geschätzte 900 Millionen Euro gekostet hat, ist Fiscus.

Fiscus sollte das „Föderale integrierte standardisierte computergestützte Steuersystem“ der Zukunft in allen 650 deutschen Finanzämtern werden. Im Sommer 2004 beschlossen die Länderfinanzminister jedoch das Aus für die einheitliche Steuererhebungssoftware. Die Projektrisiken von Fiscus lagen von Anfang an in der Koordination von 17 behördlich organisierten Auftraggebern (16 Länder und der Bund) sowie der Modellierung des komplexen deutschen Steuersystems. Das Projekt war über die lange Projekt-

laufzeit hinweg gekennzeichnet durch zahlreiche Umstrukturierungen, den Wechsel zwischen unterschiedlichen Technologieansätzen und verzweifelten Versuchen, mit zum Teil ungeeignetem Personal ein komplexes Softwareprodukt zu erstellen. Die zuletzt praktizierte gute Idee, dass fachlich versierte Beamte mit technisch versierten Entwicklern zusammenarbeiten, konnte aufgrund der bürokratisch reglementierten Kommunikation beider Seiten nicht funktionieren (Papieraustausch statt persönlicher Kommunikation im Sinne agiler Softwareentwicklungsansätze). Es gibt dazu zahlreiche anonyme Erlebnisberichte, leider jedoch keine ernsthafte Analyse aus softwaretechnischer Sicht. Indes lassen sich zum Beispiel interessante finanzielle Details entnehmen [7].

6 Zusammenfassung

Die Wichtigkeit des Risikomanagements in der Softwareentwicklung ist unstrittig. Die angegebene Klassifikation von Risiken kann bei der Erkennung, Beurteilung und Vermeidung von Risiken aller Art helfen. Die genannten Integrationen in die Prozesse der Softwareentwicklung erlauben darüber hinaus eine systematische Behandlung von Risiken in Projekten und bei der Produktentwicklung.

Die interessante Tatsache, dass für gescheiterte Projekte selten genaue Problemanalysen veröffentlich werden, führt einerseits zu einem interessanten Forschungsgebiet, sollte aber andererseits auch jeden Ingenieur zu seinen eigenen Analysen jenseits der öffentlichen Polemik motivieren. Ein weiterer bisher nicht beachteter Aspekt für die Beschäftigung mit dem Risikomanagement ist der zunehmende Einsatz von Software in eingebetteten Systemen. Eines der bekanntesten Beispiele für ein Desaster eines eingebetteten Systems ist die Zerstörung der europäischen Ariane-5-Rakete in der Startsequenz. Diese ist auf einen Programmierfehler und falsche Anforderungsspezifikationen zurückzuführen. Die Schnittstelle zu den „fassbaren“ technischen Produkten begründet damit die Notwendigkeit des Wissens um Risiken der Softwareentwicklung auch in den klassischen Ingenieurdisziplinen.

Literatur

[1] *Boehm, B. W.:* Software Risk Management: Principles and Practices. In: IEEE Software **8** (1991) 1, S. 32 – 41

[2] *Boehm, B. W.:* A Spiral Model of Software Development and Enhancement. In: IEEE Computer Society Press **21** (1998) 5, S. 61 – 72

[3] *Borchers, Detlef:* Ein Hartz für alle. In: iX 1 (2005), S. 112 – 113

[4] *Dorofee, A.; Walker, J.; Alberts, C.; Higuera, R.; Murphy, R.; Williams, R.:* Continuous Risk Management Guidebook. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1996

[5] *Krempl, Stefan:* Das Casino-Prinzip. Warum so viele IT-Großprojekte scheitern. In: c´t **23** (2004), S. 218 – 223

[6] *Naur, P.; Randell, B.:* Software Engineering. Report on a conference sponsored by the NATO Science Committee. Garmisch, Germany, 7th to 11th October 1968, Scientific Affairs Division, NATO, Brussels, 1969, S. 138 – 155

[7] *Rechnungshof Baden-Württemberg:* Denkschrift 2005. www.rh.bwl.de/sixcms/media.php/976/Denkschrift_2005.pdf

[8] The Risks Digest. www.risks.org

[9] *Wallmüller, Ernest:* Software-Qualitätsmanagement in der Praxis. 2., völlig überarb. Aufl. München/Wien: Carl Hanser, 2001

Aßmann, Uwe
Prof. Dr. rer. nat. habil.
Studium Informatik von 1983 bis 1989 an der TU Karlsruhe ♦ 1995 Promotion zum Dr. rer. nat. ♦ 2002 Habilitation zum Dr. rer. nat. habil. ♦ von 2001 bis 2004 Assoz. Professor für Softwaretechnik und seit 2004 Professor für Softwaretechnologie am Institut für Software- und Multimediatechnik, Fakultät Informatik der TU Dresden

Demuth, Birgit
Dr.-Ing.
Studium Informationsverarbeitung von 1975 bis 1980 an der TU Dresden ♦ 1983 Promotion zur Dr.-Ing. ♦ wissenschaftliche Mitarbeiterin am Institut für Software- und Multimediatechnik, Fakultät Informatik der TU Dresden

Hartmann, Falk
Dipl.-Inform.
Studium Informatik von 1991 bis 1998 an der TU Dresden ♦ 1998 Studienabschluss als Diplominformatiker ♦ wissenschaftlicher Mitarbeiter bei SAP Research, SAP AG, Dresden