Logische Modellierung von Anwendungswelten aus Benutzersicht

Maritta Heisel, Heidi Krömker

Technische Universität Ilmenau

Zusammenfassung Der Softwareentwicklung fehlt oft eine detaillierte methodische Unterstützung von technischen Softwareentwicklungsaktivitäten. Eine Autorin dieses Papiers hat das Konzept der Agenda entwickelt, das zum Ziel hat, Softwareentwicklungswissen als "methodische Essenzen" von Softwareentwicklungsaktivitäten explizit zu repräsentieren. Zur logischen Modellierung von Anwendungswelten aus Benutzersicht wird eine Agenda entwickelt, die es erlaubt diese Anwendungswelt methodisch in Konzepten der Handlungspsychologie zu erheben.

1 Gründe

Das Modellieren der Anwendungwelten geschieht im Softwareentwicklungsprozess in der Phase der Anforderungsermittlung. Obwohl sich eine Vielzahl von Disziplinen, wie z.B. das Marketing, das Requirements Engineering und Usability Engineering oder die Softwareergonomie, mit der Ermittlung von Anforderungen befassen, hat sich in Theorie und Praxis keine „methodische Essenz“ herausgebildet, mit der dies zu bewerkstelligen ist. Der Softwareentwickler steht einem weiten Spektrum von analytischen und empirischen Methoden unterschiedlichster Provenienz gegenüber.
Die präzise Antwort auf die Frage „Was eigentlich wie erhoben werden soll“ bleibt offen. Ein Pflichtenheft soll das Ergebnis der Aktivitäten sein, und das definiert die DIN 69905 so: „Ein ausführliche Beschreibung der Leistungen (z.B. technische, wirtschaftliche, organisatorische Leistungen), die erforderlich sind oder gefordert werden, um die Ziele des Projekts zu erreichen.“ Die Tatsache, dass die Anforderungsermittlung ein methodisches Stiefkind geblieben ist, wird seit Jahren mannigfaltig beklagt [FUSCH98]. Zahlreiche Analysen belegen die Notwendigkeit, praktikable Ansätze zu finden:

- Die Standish Group [STGR95] sieht in dem mangelnden Einbezug von Benutzern sowie der ungenauen Beschreibung von Anforderungen eine Hauptursache für Scheitern von Projekten. In den USA werden 31,1% der Softwareprojekte eingestellt, bevor sie beendet sind. Falls sie beendet werden, liegen 52,7% der Softwareprojekte um 189% über den ursprünglich geplanten Kosten. Vermutlich zeigen die Untersuchungen nur die Spitze des Eisbergs. Weit höhere Kosten entstehen durch die entgangenen Geschäftsmöglichkeiten.

- Nach Schätzung von Cymfony [CYM01] finden 56% der Benutzer die gewünschte Information in Applikationen nicht. Nielsen und Tahir [NITA01] fanden heraus: “In our recent E-Commerce study (see "E-Commerce User Experience"), the most common factor that stopped users from buying on a site was that they couldn't find the item they were looking for. This accounted for 27 percent of all lost sales in our study. And when they used a site's search function to try to find items, the failure rate was even higher—a full 36 percent of users couldn't find what they wanted.”. Firmen, denen es gelingt, Applikationen benutzergerecht zu gestalten, sparen somit Kosten und steigern die Zufriedenheit der Anwender.

Abgesehen von der Unzufriedenheit der Anwender mit den Softwareprodukten tritt auch bei etablierten und erfolgreichen Produkten die Notwendigkeit auf, bei der Entwicklung neuer Produktgenerationen, Anforderungen neu zu definieren. Fortgeschrittene Produktgenerationen sind oft mit Funktionen überfrachtet, die kaum oder nicht genutzt werden. Die Funktionen müssen auf die aus Benutzersicht Wesentlichen reduziert werden [KK95]. Um auf effiziente Weise eine Variantenvielzahl zu erzielen, werden Produkte modularisiert, d.h. in Bausteine zerlegt, die aus Benutzersicht eine sinnvolle funktionale Einheit bilden. Auch hier muß die Aufgaben- und Denkwelt des Benutzers systematisch durchdrungen werden, um die jeweils richtigen Funktionen zu einem Baustein zu kombinieren. Da fast alle Produkte auf dem internationalen Markt verkauft werden, muß die Software auch in Hinblick auf unterschiedliche kulturspezifische Kundenbedürfnisse gestaltet werden. Kulturspezifische Unterschiede der Benutzergruppen, der Anforderungen an das Produkt und in der Produktnutzung müssen systematisch identifiziert werden [KK00].

2 Praxis des Erhebens

Die Heterogenität der Theorielandschaft der Anforderungsermittlung spiegelt sich in der Vielfalt der praktischen Vorgehensweisen wider. Die Situation, in der sich der Entwickler befindet, charakterisiert Schinzel [SCH98, S. 5]:
„Tatsächlich aber findet ein Kommunikationsprozeß zwischen fremden Welten ….statt, in dessen Verlauf die Vielgestaltigkeit der Organisationsrealität, Differenzen und Ambiguitäten in Wahrnehmung, wie persönliche und fachspezifische Annahmen, theoretische Konstrukte, Denkstile, Betroffenheit und Interessen, wie Untersuchungs- und Firmeninteressen und sozialen Praktiken zutage treten können.“ Die im Laufe des Softwareentwicklungsprozesses erworbenen Verständnismuster basieren meist auf unkontrollierten individuellen Interpretationen der Beteiligten. Verstärkt wird dieses Phänomen noch durch die generelle Wertschätzung der formalen und technischen Anteile im Softwareentwicklungsprozess gegenüber den ermittelnden und gestaltenden Anteilen der Softwareentwicklung [SCH95].
Als Lösungsansatz soll ein Konzept zur systematischen Durchführung von Aktivitäten im Softwareentwicklungsprozess angewendet werden. In diesem Konzept sollen Methoden zur logischen Modellierung von Anwendungswelten ausgedrückt werden.

3 Konzept zur systematischen Durchführung von Aktivitäten im Softwareprozess

Gängige Prozessmodelle, wie das V-Modell [VER00] oder der Rational Unified Process [JAC+99], bieten eher einen *organisatorischen* Rahmen zur Durchführung von Softwareentwicklungsprojekten. Methodische Unterstützung jedoch fehlt oftmals. Softwareentwicklungswissen wird nicht nur bei der Anforderungserhebung, sondern auch in späteren Phasen nicht explizit gemacht. Wissen, das nicht explizit repräsentiert ist, kann aber weder für Neulinge vermittelt noch systematisch wiederverwendet und weiterentwickelt werden.
Um diesen Zustand verbessern zu helfen, hat eine Autorin dieses Papiers das Konzept der *Agenda* entwickelt [HEI98]. Ziel dieses Konzeptes ist es, Softwareentwicklungswissen explizit machen, d.h. "methodische Essenzen" von Softwareentwicklungsaktivitäten explizit zu repräsentieren. Das Agendenkonzept baut auf der Beobachtung auf, daß erfahrene Softwareingenieure die verschiedenen Aktivitäten besser und schneller ausführen können als Neulinge, da sie mit der Zeit ein fundiertes problemspezifisches *Wissen* erworben haben. Eine Agenda gibt Anleitung, wie eine spezifische Softwareentwicklungsaktivität durchzuführen ist. Agenden können für viele verschiedene Aktivitäten und Kontexte aufgestellt werden. Beispiele sind Anforderungsermittlung (siehe Abschnitt 4), Spezifikationsakquisition, Softwareentwurf mittels Architekturstilen, objektorientierte Analyse und Entwurf, sowie die Entwicklung von Code aus Spezifikationen. Technisch gesehen besteht eine Agenda aus einer Liste von Schritten, die auszuführen sind, um eine Softwareentwicklungsaktivität durchzuführen. Das Ergebnis der Aktivität ist ein Dokument, das in einer bestimmten Sprache (sei sie formal oder informell) ausgedrückt ist.
Agenden dienen aber nicht nur zur Anleitung von Softwareentwicklungsaktivitäten, sondern auch zur Qualitätssicherung der entwickelten Dokumente. *Validierungsbedingungen*, die den einzelnen Schritten zugeordnet sein können, repräsentieren notwendige semantische Bedingungen, die das Dokument erfüllen muß, um seinen Zweck zu erfüllen. Agenden werden als Tabellen dargestellt, wie in Tabelle 1 gezeigt. Die Schritte einer Agenda müssen nicht unbedingt in der angegebenen Reihenfolge bearbeitet werden. Zur Definition einer Agenda gehört auch ein Abhängigkeitsgraph, der angibt, welche Schritte voneinander abhängen und welche nicht.

Eine andere Art der Repräsentation von Softwareentwicklungswissen sind *Muster*. Muster sind Abstraktionen von *Artefakten*, die jeweils geeignet instantiiert werden können. Muster sind auf verschiedenen Abstraktionsebenen einsetzbar: *Problem Frames* [JACK01] dienen der Charakterisierung und Strukturierung von Softwareentwicklungsproblemen. *Software-Architekturstile* [SG96] dienen der Charakterisierung und Strukturierung von Softwaresystemen. *Entwurfsmuster* [GHJV95] kommen beim Feinentwurf zum Einsatz, und *Idiome* [BMRSS96] dienen der Programmierung. Muster und Agenden ergänzen einander, da das eine Konzept eher an Produkten und das andere eher an Abläufen orientiert ist. Beide Konzepte sollten bei der systematischen Softwareentwicklung zum Einsatz kommen.

4 Agenda zur logischen Modellierung von Anwendungswelten

Der Vorschlag einer Agenda zur logischen Modellierung von Anwendungswelten basiert auf folgenden Überlegungen. Das logische Durchdringen der Anwendungswelt soll sich eng an dem mentalen Modell orientieren, das der Benutzer von der Anwendungswelt hat. Bestimmendes Element dieser Anwendungswelt ist aus Sicht des Benutzers die *Aufgabe*. Nach Norman [NO86] besteht die Bearbeitung einer Benutzeraufgabe aus den Schritten, wie sie Abb. 1 zeigt.

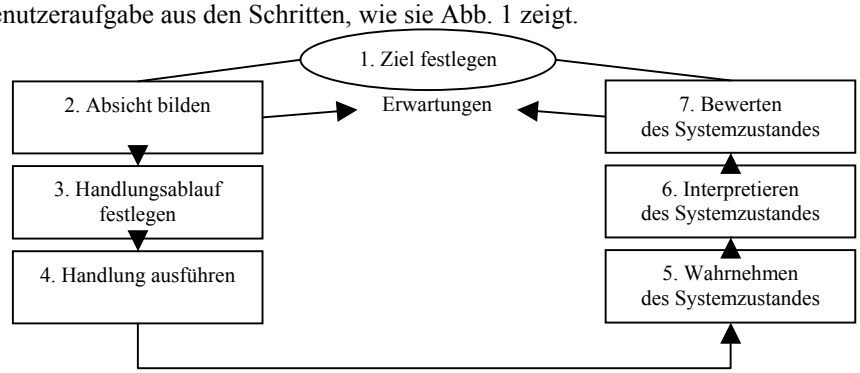


Abb.1. Aufgabenbearbeitung nach Norman [NO86]

Jede Aufgabe lässt sich wiederum hierarchisch in Teilaufgaben zerlegen, deren Abarbeitung sich in einem hierarchisch sequentiellen Handlungsmodell (vgl. Abb. 2) abbilden lässt. [VO83].

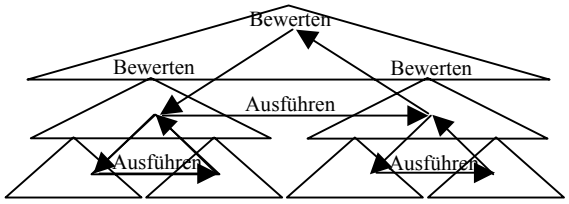


Abb. 2. Hierarchisch sequentielles Handlungsmodell nach Volpert [VO83]

Für das Beispiel eines Bordcomputers für ein Kraftfahrzeug könnte ein Modell aussehen, wie es Abb. 3. zeigt. Die Zerlegung der Teilhandlungen ist unabhängig von einer möglichen technischen Lösung vorzunehmen. Sie soll ausschließlich die Handlungslogik aus Benutzersicht repräsentieren. Die Handlungen werden dabei in ihrer Objekt-Aktions-Beziehung beschrieben. Das heißt, aus Sicht des Benutzers gibt es das Objekt „Vorschlag“, und mit ihm müssen die Aktionen „prüfen“, „modifizieren“ und „akzeptieren“ ausgeführt werden.

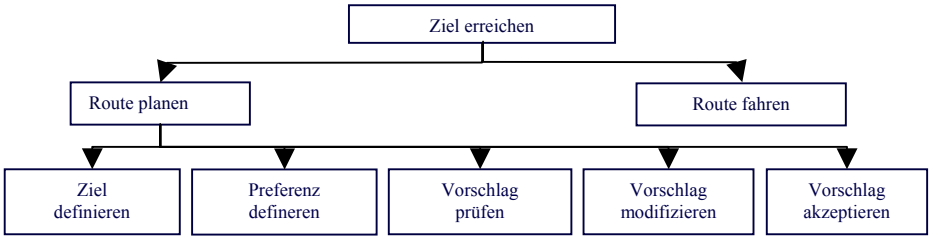


Abb. 3. Beispiel für die Bedienung eines Navigationssystems im Kraftfahrzeug

Komplexe Systeme werden im Allgemeinen von Benutzern mit unterschiedlichen Aufgaben bedient. Benutzer mit ähnlichen Aufgaben haben eine „Rolle“. Die Rolle ist ebenfalls ein bestimmender Faktor für die Sicht der Anwendungswelt [OB88].
Eine Autorin schlägt daher für die erste Phase der Anforderungserhebung die systematische Ermittlung der genannten Konzepte vor. Sie sollten Ausgangspunkt, aber auch Referenz für alle weiteren Aktivitäten im Softwareentwicklungsprozess sein. Die Konzepte lassen sich zu einer Agenda zusammenfassen, wie sie Tabelle 1 zeigt.

Nr.	Schritt	Validierungsbedingungen
1	Rollenidentifikation	Jede Rolle muß im Handlungsmodell vorkommen.
2	Begriffsbestimmung	Jeder Begriff muß beschrieben sein, der im hierarchischen Handlungsmodell und bei den Objekt-Aktionsbeziehungen vorkommt.
3	Objekt-Aktionsbeziehungen	Alle Objekte und Aktionen müssen im Handlungsmodell vorkommen.
4	Handlungsmodelle	Alle Objekte und Aktionen des Handlungsmodells müssen in den Objekt-Aktionsbeziehungen vorkommen.

Tab. 1 Agenda für die Anforderungserhebung.

Der folgende Abschnitt beschreibt die Methoden, die sich in der Praxis als effektiv und effzient herausgestellt haben, um diese Schritte durchzuführen.

5 Psychologische Methoden der Anforderungserhebung

Bei der Auswahl der Methoden, um Information aus der Anwenderwelt für diese Konzepte zu erhalten, wird von folgenden Kriterien ausgegangen:

- Benutzerzentrierte Gestaltung findet fast immer in einem interdisziplinären Entwicklungsteam, bestehend aus Mitarbeitern aus Marketing, Entwicklung und Vertrieb, statt. Für die Wahl der Methoden bedeutet dies auch, dass sie eine systematische Kommunikationsgrundlage für alle Beteiligten bilden müssen. Dies ist wich-

tig, da das Erfahrungswissen aller in eine Gestaltung einfließen muß.

- Das Ziel, Zeit und Kosten in den Produktlebenszyklen zu reduzieren, erfordert eine hohe Effizienz der Methoden.
- Der Innovationsbedarf führt dazu, besonderen Schwerpunkt auf Methoden zu legen, mit denen es möglich ist, neue Funktions- und Gestaltungsideen zu gewinnen und zu bewerten. Ein gezielter Einsatz solcher Methoden hat besondere Bedeutung, da über die Hälfte aller Produktinnovationen auf Kundenanregungen zurückgeht.
- Um die Anwendungswelt in ihrer Gesamtheit zu verstehen, sollten in dieser Phase vor allem qualitative Methoden verwendet werden. Beispiele für qualitative Methoden sind etwa strukturierte Interviews, Gruppendiskussionen (z.B. Focus Groups) sowie teilnehmende Bebachtung (z.B. Contextual Task Analysis) [MAY99]. Ziel der Analyse der erhobenen Daten ist das Verstehen der Anwendungswelt mit ihren Aufgaben [LA95].

6 Integration in den Softwareentwicklungsprozess

Die Erhebung von Benutzeranforderungen, wie in Abschnitt 4 beschrieben, stellt einen ersten sehr wichtigen Schritt dar, der die Voraussetzung für die darauffolgende Softwareentwicklung ist. Es ist nun wichtig, die Anforderungmodellierung geeignet in die die weiteren Phasen des Softwareentwicklungsprozesses zu integrieren. Hierbei sind folgende Problemfelder zu beachten:

Geeignete Repräsentation der Anforderungen

In der Phase der Anforderungserhebung müssen die Benutzeranforderungen in einer Form dargestellt werden, die für *alle Beteiligten am Softwareentwicklungsprozess* verständlich ist. In den weiteren Phasen des Softwareentwicklungsprozesses müssen die Anforderungen zunächst in eine Softwarespezifikation und dann in ein laufendes System überführt werden. Hier müssen sie in erster Linie für die *Softwareentwickler* verständlich sein und zudem so unmissverständlich ausgedrückt werden, dass die Vorgaben für die Systementwicklung eindeutig sind. Es ist zu untersuchen, ob mit dem Kontextwechsel der Anforderungen auch ein Repräsentationswechsel nötig ist, oder ob eine Repräsentation der Anforderungen gefunden werden kann, die sowohl verständlich als auch genau und gut analysierbar ist. Diagrammnotationen wie z.B. die *Unified Modeling Language* [UML99] könnten hier einen Ansatz bieten.

Analyse und Herstellung der Kohärenz sowie Umgang mit Widersprüchen

Es liegt in der Natur der Sache, dass Benutzeranforderungen widersprüchlich oder in anderer Weise unverträglich sein können. Dies gilt insbesondere für multimediale Informations- und Kommunikationssysteme, die für viele Benutzer ausgelegt sind, welche jeweils unterschiedliche Bedürfnisse haben können. Zur Aufdeckung von Widersprüchen und Unverträglichkeiten gibt es verschiedene Ansätze, z.B. Model Checking [PR98] oder heuristische Algorithmen [HS00].

Umsetzung der Benutzeranforderungen in Softwarespezifikationen.

Benutzeranforderungen sind nicht mit Softwarespezifikationen identisch [JACK95]. Anforderungen beschreiben, wie die Benutzer bestimmte Arbeitsgänge ausführen wollen. Spezifikationen schreiben, wie die Software diese Arbeitsgänge unterstützen

sollte. Wir schlagen vor, auch den Übergang von Benutzeranforderungen zu Softwarespezifikationen systematisch zu gestalten und mit Agenden und Tests zu unterstützen. Ansätze für das grundsätzliche Vorgehen hierbei existieren bereits [JZ95, HS99].

Analyse der Auswirkungen von Änderungen auf die erhobenen Benutzeranforderungen

Mit dem raschen Fortschritt bei multimedialen Techniken geht auch eine rasche Veränderung der Benutzerbedürfnisse einher. Bestehende Software muß in immer kürzeren Zeitabständen an neue Bedürfnisse angepasst werden. Im Fachgebiet Softwaretechnik und Programmiersprachen der TU Ilmenau wurde mit Forschungsaktivitäten begonnen, die das Ziel haben, die Evolution von Software, also die Anpassung an neue Bedürfnisse, systematisch und kostengünstig durchzuführen. Erste Ideen hierzu sind in [HEI00] beschrieben.

Einsatz von Mustern

Wie schon in Abschnitt 3 angerissen, sind neben Agenden auch Muster ein wichtiges Hilfsmittel in allen Phasen der Softwareentwicklung. Allerdings existieren auf dem Gebiet der Modellierung von Anwendungswelten aus Benutzersicht bisher keine solchen Muster. Es ist daher ein lohnendes Forschungsvorhaben, nach Mustern zu suchen, die dabei helfen, Anwendungswelten zu modellieren.

Literatur

[BMRSS96] Buschmann, F.; Meunier, R.; Rohnert, H.; Sommerlad, P.; Sal, M.: Pattern-Oriented Software Architecture: A System of Patterns. John Wiley & Sons, 1996

[CYM01] Cymfony (2001). Cymfony-about, -- Retrieved August 26, 2001, from http://www.cymfony.com/about.html

[FUSCH98] Funken Ch., Schinzel B.: Die Anforderungsermittlung als Fehlerquelle. In: Otto, W.H. (Hrsg.): „Faszination Computer“ – Die Informationstechnologie im Expertenspiegel. Festschrift für Konrad Zuse; München 1998

[GHJV95] Gamma, E.;Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison Wesley, 1995

[HEI00] Heisel, M.: Toward an Evolutionary Software Technology. In: Reggio, G.; Astesiano, E.: Modelling Software System Structures in a fastly moving scenario. Dipartimento di Informatica e Scienze dell'Informazione, Universita' di Genova, 2000; http://www.disi.unige.it/person/ReggioG/PROCEEDINGS

[HEI98] Heisel, M.: Agendas - A Concept to Guide Software Development Activites. In: R. N. Horspool (ed.): Proc. Systems Implementation 2000. Chapman & Hall London, 1998, S. 19-32

[HS00] Heisel M.; Souquières J.: A heuristic algorithm to detect feature interactions in requirements. In: Gilmore, S.; Ryan, M.: Language Constructs for Describing Features. Springer-Verlag, 2000, S. 143-162

[HS99] Heisel, M. Souquières, J.: A Method for Requirements Elicitation and Formal Specification. In: Akoka J.; Bouzeghoub, M.; Comyn-Wattiau, I.; Métais, E.: Proc. 18th Int. Conf. on Conceptual Modeling. Springer-Verlag, 1999, S. 309-324

[JAC+1999] Jacobson, I.; Booch, G.,; Rumbaugh, J.: The Unified Software Development Process. Addison-Wessley, 1999
[JACK01] Jackson, M.: Problem Frames. Analyzing and structuring software development problems. Addison-Wesley, 2001
[JACK95] Jackson, M.: Problems and Requirements. In: Proceedings of the IEEE Second International Symposium on Requirements Engineering, ACM Press, 1995
[JZ95] Jackson, M.; Zave, P.: Deriving Specifications from Requirements: an Example. Proc. 17th Int. Conf. on Software Eng.. Seattle, USA, ACM Press, 1995, S. 15
[KK95] Krömker H. Gestaltung der Mensch-Maschine Schnittstelle in der industriellen Forschung und Entwicklung. In: ITG-Fachbericht 154, Technik für den Menschen, Gestaltung und Einsatz benutzungsfreundlicher Produkte, Vorträge der ITG-Fachtagung am 26. Und 27. Oktober 1998 an der katholischen Universität Eichstätt, VDE-Verlag GmbH Berlin, 1998, 171-179
[KK00] Krömker H. (Special Issue Editor): Ease and Joy of Use for Complex Systems at Siemens. International Journal of Human-Computer Interaction (IJHCI), Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, New Jersey, 2000
[LA95] Lamnek, S.: Qualitative Sozialforschung. Band1. Methoden und Techniken. Belz Psychologie Verlags Union, 1995, S. 259
[MAY99] Mayhew, D.J.: The Usability Engineering Lifecycle, Morgan Kaufmann Publishers, Inc., 1999
[NITA01] Nielsen J., Tahir M.: Building web sites with depth. 2001 (2) retrieves July 19, 2002 from http://www.webtechniques.com/archives/2001/02/nielsen/
[NO86] Donald A. Norman: Cognitive Engineering. In: Donald A. Norman & Stephen W. Draper (Hrg.) : User Centered System Design. New Perspectives on Human-ComputerInteraction. Hilffsdale, New Jersey: Erlbaum, 1986, S. 31-61.
[OB88] Oberquelle, H.: Role-Function/Action-Nets as a Visual Language for Cooperative Modelling. In: WHISAD 88: Proc. of the IFIP Int. Workshop on Human Factors of Information Systems Analysis and Design. Imperial Coll, 1988, S. 1-21
[PR98] Plath, M and Ryan, M.: Plug-and-Play features. In: Kimbler, K.; Bouma, W. (ed.):5th Feature Interaction Workshop, FIW'1998, S. 150-164.
[SCH98] Schinzel B.: Mit Frauen gegen die Softwarekrise: FIfF- Kommunikation, 1, 1998, S. 5 retrieved July 19, 2002 from
http://mod.iig.uni-freiburg.de/publikationen/online-publikationen/fiff43.pdf
[SG96] Shaw, M.; Garlan D.: Software Architecture. IEEE Comp. Soc. Press, 1996
[STGR95] The Standish Group: T23E-T10E Standish Group Report –Chaos, retrieved July 19, 2002, from http://www.scs.carleton.ca/~beau/PM/Standish-Report.html
[UML99] Booch G. and Rumbaugh J. and Jacobson I.: The Unified Modeling Language User Guide. Addison-Wesley, 1999
[VER00] Versteegen, G.: Das V- Modell in der Praxis. Grundlagen, Erfahrungen, Werkzeuge. dpunkt.verlag, 2000
[VO83] Volpert W.: Das Modell der hierarchisch-sequentiellen Handlungsorganisation. In W. Hacker, Kognitive und motivationale Aspekte der Handlung, 1983