

# Aufgabenmodellierung als Kernelement eines nutzerzentrierten Entwicklungsprozesses für Bedienoberflächen

Gerrit Meixner

Deutsches Forschungszentrum für  
Künstliche Intelligenz (DFKI) GmbH,  
Trippstadter Straße 122  
67663 Kaiserslautern  
Gerrit.Meixner@dfki.de

Daniel Görlich

TU Kaiserslautern  
Zentrum für Mensch-Maschine-Interaktion  
Gottlieb-Daimler-Str. 42  
67663 Kaiserslautern  
goerlich@mv.uni-kl.de

**Abstract:** The development of user interfaces has become a decisive factor for the acceptance of software and technical devices. Model-based user interface development (MBUID) is in research and still poses an enormous challenge. MBUID uses a multitude of different models which are related to each other. The analysis of the users and their tasks, as well as technical functionalities, ranges and fields of applications is instrumental in the development of model-based user interfaces. In the Useware-Engineering process, data gained from such an analysis can be applied systematically to the conception and structuring of the user interface to be designed. One of the results in the analysis phase is a generic task model. This paper introduces the Useware-Engineering process focussing the task model.

## 1 Einleitung

In den letzten Jahren haben die Entwickler von Bedienoberflächen erkannt, dass die Integration von Endnutzern und deren Aufgaben ein sehr wichtiger Faktor für den Erfolg des Software-Produktes ist. Für die Entwicklung nutzerzentrierter Software sind herkömmliche Entwicklungsprozesse nur bedingt geeignet, da explizit nicht der Fokus auf gebrauchstauglicher Seite, sondern eher auf Funktionsseite liegt. Da noch kein vollständiger Entwicklungsprozess existiert, der eine Integration zwischen Nutzerzentrierung und klassischer Software-Entwicklung vorweisen kann, besteht eine Verbindungsmöglichkeit in der Benutzung formaler Modelle. In der Software-Entwicklung ist die Modellierung der Applikationslogik, z.B. mit UML [OMG07], gängige Praxis. Bei der Entwicklung von Bedienoberflächen wird seit mehreren Jahren an der modellbasierten Entwicklung geforscht [Pue97]. Die Vorgehensweise entspricht dabei weitgehend den modellbasierten Ansätzen, die bei der Entwicklung der Applikationslogik von Software verwendet werden. Ansätze, wie z.B. durchgängige Transformation von Modellen, die im Rahmen der modellgetriebenen Entwicklung von Software (MDA) [Gru06] verfolgt werden, sind auch bei der modellbasierten Entwicklung von Bedienoberflächen ein wichtiges und auch

zentrales Element um den Entwicklungsprozess effizienter zu gestalten. Ebenfalls beachtet werden muss die starke Diversifikation von Computer in den letzten Jahren, z.B. Desktop PC, PDA oder Mobiltelefon, für die Bedienoberflächen entwickelt werden müssen. Da die Geräte verschiedene Spezifikationen, wie z.B. Bildschirmgröße oder Betriebssystem, vorweisen und auch eine Vielfalt an Programmiersprachen und Grafikbibliotheken existieren, ist das Entwickeln von konsistenten Bedienoberflächen für verschiedene System sehr zeit- und kostenaufwändig. Mit Hilfe der modellbasierten Entwicklung kann der Entwicklungsprozess für Bedienoberflächen über Plattformgrenzen hinweg effizienter gestaltet werden. Basis eines nutzerzentrierten Entwicklungsprozesses ist die Fokussierung auf die – evtl. zukünftigen, bisherigen oder potenziellen – Nutzer sowie deren Aufgaben, Vorgehensweisen und Arbeitsbedingungen. Die Aufgaben und ihre Strukturen werden durch Aufgabenmodelle spezifiziert, die relevanten Charakteristika der Nutzer gewöhnlich in Nutzergruppen, Personas, oder vergleichbaren Archetypen abgebildet. Wesentliches Ziel einer nutzerzentrierten Entwicklung ist die Orientierung der Technikgestaltung an den Fähigkeiten, Anforderungen sowie physischen und kognitiven Kapazitäten der Nutzer [vgl. Züh04]

In [Car83] werden Aufgaben mittels des GOMS-Modells (Goals, Operators, Methods and Selection Rules) definiert. GOMS ist eine Methode zur Modellierung der Aufgabebearbeitung mit einem gegebenen System. Die kognitive Struktur eines Benutzers wird durch Ziele, Operatoren, Methoden und Auswahlregeln beschrieben. Das Ziel der Modellierung liegt hierbei auf der Abschätzung des Lernaufwandes für den Erwerb des prozeduralen Benutzerwissens und der Ermittlung von Bearbeitungszeiten für die Aufgabendurchführung. Mittels GOMS ist es nicht möglich Aufgabenstrukturen, im Hinblick auf einen durchgängigen nutzerzentrierten und teilautomatisierbaren Entwicklungsprozess, im Sinne der MDA zu modellieren.

Paternò [Pat00] entwickelte mit der Sprache Concur Task Trees (CTT) eine Notation für die Spezifizierung von Aufgabenmodellen. Bedingt durch die Strukturierung der Aufgabentypen und einer groben Aufteilung der interaktiven Nutzeraufgaben (Interaction Task) eignet sich CTT, ohne eine feingranularere Aufteilung, nur bedingt für einen durchgängigen nutzerzentrierten modellbasierten Entwicklungsprozess.

Um die klassisch hierarchische Aufgabenanalyse mit dem Workflow-Management zu verbinden, hat [Træ02] nachgewiesen, dass die vier Basiskonzepte zur Beschreibung von Workflows, nämlich Handlungsstruktur, Akteure, Werkzeuge und Informationen, direkte Entsprechungen in der Domäne der Aufgabenmodellierung besitzen. Darauf aufbauend hat er mit TaskMODL eine Sprache geschaffen, die gleichzeitig Aufgaben-hierarchien wie auch workflow-typische Compartments (ähnlich einem UML-Aktivitätsdiagramm) darstellen kann.

Dieser Beitrag ist wie folgt aufgebaut: Das nächste Kapitel beschreibt den Useware-Engineering-Prozess. Im Anschluss wird näher auf das Konzept des MBUID eingegangen. Anschließend wird das generische Aufgabenmodell des Useware-Engineering-Prozesses beschrieben. Der Abschluss dieses Beitrages bildet eine Zusammenfassung sowie ein Ausblick auf zukünftige Arbeiten.

## 2 Useware-Engineering Prozess

Während die Nutzerorientierung von Bedienoberflächen längst zu einem entscheidenden Faktor für die Akzeptanz von Software geworden ist, stellt deren Entwicklung heutzutage noch immer eine enorme Herausforderung dar. Der steigende Aufwand bei der Bedienoberflächenentwicklung auf Grundlage wissenschaftlicher Methoden hat zu einem Entwicklungsanteil geführt, der ähnlich groß wie in den traditionellen Feldern des Software-Engineerings ist [Züh04]. 1999 wurde in Analogie zu den Begriffen Hard- und Software der Begriff „Useware“ als Sammelbegriff für alle Komponenten, die der Benutzung dienen, eingeführt [Züh99]. Mit ihm verbindet sich eine Fokussierung der Technikgestaltung auf menschliche Fähigkeiten und Bedürfnisse.

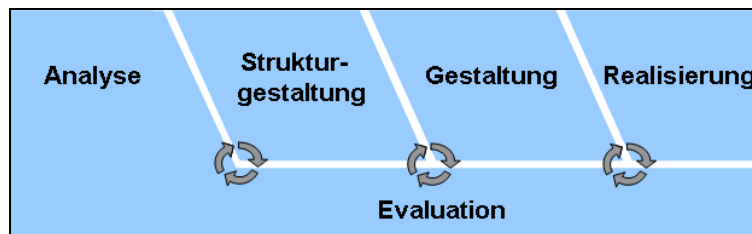


Abbildung 1. Useware-Engineering-Prozess

Die Entwicklung von nutzergerechten Bedienoberflächen kann nur durch einen systematischen Prozess sichergestellt werden [Züh04]. Die wichtigsten Daten, die in diesem Entwicklungsprozess erhoben und verwendet werden, sind die Anforderungen der Nutzer an die Bedienoberfläche und somit auch an Teile der Software. Nur mit einer nutzergerechten Entwicklung kann eine effiziente Nutzung der Software für den Anwender garantiert werden. Der Entwicklungsprozess gliedert sich in die Phasen: Analyse, Strukturgestaltung, Gestaltung, Realisierung und Evaluation (siehe Abb. 1). Jede dieser Phasen darf nicht isoliert, sondern muss überlappend, betrachtet werden. Die Evaluierung wird im gesamten Prozess parallel durchgeführt und stellt so eine Weiterführung der Analyse dar.

Durch die parallele Evaluierung z.B. durch Struktur- oder Funktionsprototypen können die Nutzer in den Prozess integriert werden, um erarbeitete Konzepte zu testen und zu bewerten. Die Durchgängigkeit des Prozesses sowie die Verwendung geeigneter Werkzeuge, z.B. auf Basis der Extensible Markup Language (XML), ermöglichen die Vermeidung von Informationsverlusten und Medienbrüchen.

Die Grundlage des gesamten Entwicklungsprozesses bildet die Analysephase mit der Erhebung von Anforderungen bzw. Aufgaben der Nutzer an die Bedienoberfläche. Die erfassten Aufgaben werden formal mit Hilfe eines Aufgabenmodells im Rahmen des MBUID modelliert.

### 3 Modellbasierte Entwicklung von Bedienoberflächen

MBUID stellt einen Rahmen zur Entwicklung konsistenter plattformunabhängiger Bedienoberflächen zur Verfügung. MBUID kann unter anderem zur Komplexitätsbeherrschung verwendet werden [Mye00]. Modellbasiert bedeutet in diesem Kontext, dass eine abstrakte Darstellung von Bedienoberflächen verwendet wird, welche dann (semi-) automatisch in eine konkrete Repräsentation der jeweiligen Plattform umgesetzt wird.

Zur Erzeugung der Bedienoberflächen werden beim MBUID verschiedene, formal definierte Modelle in den Prozess mit einbezogen, die das Aussehen (Look&Feel), die Aufgaben, die Domäne und die Benutzer beschreiben [Van03]. Das Interface-Modell ist eine Sammlung verschiedener Modelle, die zusammen die gesamte Bedienoberfläche beschreiben. Allerdings decken diese Modelle jeweils nur einzelne Aspekte der Bedienoberfläche ab [Pue97]. Bisher gibt es noch keine normierte Menge an Modellen, die zusammen das Interface-Modell ergeben. Umsetzungen wie in [Luy04] haben allerdings gezeigt, dass der Kern des Interface-Modells, welches zur Erzeugung der Bedienoberfläche benötigt wird, aus Aufgaben-, Dialog- und Präsentationsmodell besteht.

Neben dieser rein abstrakten Definition von Bedienoberflächen ist eine strukturierte Entwicklung von Bedienoberflächen möglich, da verschiedene Modelle mit unterschiedlichen Abstraktionsstufen zur Entwicklung vorliegen [Mor04]. Dabei kann zwischen abstrakten und konkreten Modellen unterschieden werden. Das Aufgabenmodell ist ein abstraktes Modell, da die Elemente des Aufgabenmodells kein direktes Pendant in der Bedienoberfläche aufweisen. Dialog- und Präsentationsmodell sind konkrete Modelle, da deren Elemente direkt auf Elemente der Bedienoberfläche transformiert werden können [Luy04] [Mor04] [Pue99].

Das Dialogmodell beschreibt die Interaktion des Nutzers und deren Auswirkung an der Bedienoberfläche. Für die Modellierung von Dialogen gibt es eine Vielzahl verschiedener Notationen z.B. Endliche Automaten, Petri-Netze, UML-Aktivitätsdiagramme oder die Backus-Naur-Form (BNF). Das Dialogmodell verbindet das Aufgabenmodell mit dem Präsentationsmodell. Es stellt den dynamischen Teil bei der Ausführung von Aufgaben des Nutzers (Aufgabenmodell) dar. Es ist daher eng mit der (visuellen) Darstellung des Präsentationsmodells verknüpft, das die konkrete Realisierung der Bedienoberfläche repräsentiert. Die Art und Weise, wie Oberflächen dargestellt werden, hängt allerdings vom Ausgabemedium ab (z.B. Displaygröße, Farbdarstellung, etc.). Dabei kann die Interaktion z.B. über eine grafische Oberfläche eines PCs oder durch Sprachsteuerung mit einem Mobiltelefon stattfinden. Das Präsentationsmodell setzt sich aus dem abstrakten und konkreten Präsentationsmodell zusammen. Typischerweise bestehen nach [Van93] Bedienoberflächen aus einer Menge abstrakter Interaktionsobjekte (AIO), die im nächsten Schritt auf korrespondierende konkrete Interaktionsobjekte (CIO) abgebildet werden. Die Gesamtheit der AIO's ergeben das abstrakte und die Gesamtheit der CIO's das konkrete Präsentationsmodell. Ein CIO ist demnach die konkrete Darstellung eines Objektes.

Das folgende Kapitel beschreibt das verwendete Aufgabenmodell im Ueware-Engineering Prozess.

## 4 Das Benutzungsmodell

Ein Aufgabenmodell ist ein Modell, um Aufgaben, die in der Analysephase identifiziert wurden, zu strukturieren. Nach [Pat00] ist die Bildung eines Aufgabenmodells für folgende Punkte nützlich:

- **Verständnisbildung innerhalb der Anwendungsdomäne:** Ein Aufgabenmodell beschreibt u.a., welche Funktionen in einer Anwendungsdomäne benutzt werden. Als Resultat gibt das Aufgabenmodell dem Software Entwickler nützliche Hinweise, welches Features bzw. Funktionen die zu erstellende Applikation unterstützen sollte.
- **Interdisziplinäre Darstellung der Resultate:** Ein Aufgabenmodell ist abstrakt genug, um von Personen aus verschiedenen Abteilungen, die im Software Entwicklungsprozess involviert sind, verstanden zu werden. Daher ist es nützlich, im Rahmen der Software Anforderungsanalyse, ein Aufgabenmodell zu spezifizieren.
- **Bessere Integration von Nutzeranforderungen:** Wird das Aufgabenmodell während des ganzen Software Entwicklungsprozesses als Informationsquelle genutzt, können Applikationen entstehen, die Nutzeranforderungen deutlich besser abdecken.
- **Analyse und Evaluation:** Aufgabenmodelle können zur Analyse, Validierung und Identifikation von Usability Problemen genutzt werden.

Die Strukturierung der Aufgaben zu einem Aufgabenmodell erfolgt innerhalb des Ueware-Engineering-Prozesses in der Strukturgestaltungsphase (siehe Abb. 1). Um die zentrale Position des Nutzers hervorzuheben, wurde von [Reu03] das Aufgabenmodell des MBUID im Ueware-Engineering Prozess als Benutzungsmodell definiert. Dementsprechend werden die im Benutzungsmodell beschriebenen plattformunabhängigen Tätigkeiten, Handlungen und Operationen abstrahiert als Benutzungsobjekte bezeichnet.

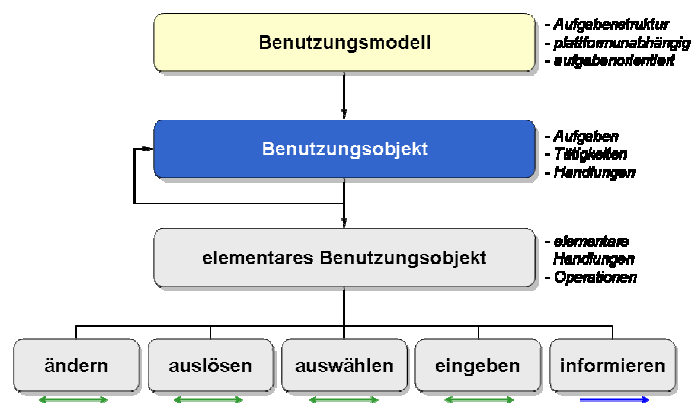


Abbildung 2. Aufbau eines Benutzungsmodells

Das hierarchisch zu gliedernde Benutzungsmodell wird mit Hilfe einfacher Konstrukte erstellt (vgl. Abb. 2). Seine Grundstruktur basiert auf Benutzungsobjekten. Diese enthalten Informationen über z.B. Zuständigkeiten, Rechte und Beziehungen. Den jeweiligen,

durch Benutzungsobjekte beschriebenen Aufgaben, werden elementare Benutzungsobjekte zugeordnet, welche die fünf elementaren Teilaufgaben „Informieren“, „Auslösen“, „Auswählen“, „Ändern“ und „Eingeben“ repräsentieren. Die Plattformunabhängigkeit und die generische Struktur des Benutzungsmodells ermöglichen so eine von konkreten Objekten unabhängige Aufgabenmodellierung.

Benutzungsobjekte bilden durch Aggregationsbeziehungen die Knoten der Struktur des Benutzungsmodells. Sie stellen die in der Analyse identifizierten Aufgaben der Benutzer dar. Neben den Aufgabenstrukturen enthalten Benutzungsobjekte zudem Informationen über Nutzergruppen, Personas [Thi06], Gerätetypen oder mögliche Zugriffsorte [Gör07].

Den Benutzungsobjekten werden elementare Benutzungsobjekte zugeordnet. Diese repräsentieren atomare Teilaufgaben. Die elementaren Benutzungsobjekte werden in fünf unterschiedliche Objekte eingeteilt:

- „Informieren“: Informieren bezeichnet die elementare Aufgabe des Benutzers, Informationen über einen Sachverhalt zu erlangen, z.B. durch Ablesen eines Wertes.
- „Auslösen“: Das Auslösen repräsentiert die direkte Auslösung genau einer Funktion, z.B. „Zähler zurücksetzen“.
- „Auswählen“: Beim Auswählen kann der Benutzer auf eine bestimmte Menge vorgegebener Auswahlmöglichkeiten zurückgreifen, z.B. „Betriebsmodus auswählen“.
- „Eingeben“: Der Benutzer übermittelt dem Computer einen absoluten Wert, z.B. „Suchbegriff eingeben“.
- „Ändern“: Beim Ändern wird eine relative Wertmanipulation vorgenommen. Der Benutzer kann den Wert lediglich inkrementieren bzw. dekrementieren.

Die Umsetzung des Benutzungsmodells wurde auf Basis der XML-Technologie durchgeführt. In [Reu03] wurde die Ueware Markup Language (useML) entwickelt, mit der es möglich ist, Benutzungsmodelle zu definieren und zu serialisieren.

Grundlage für die Beschreibung des Benutzungsmodells mit Hilfe von useML ist ein XML-Schema. In diesem werden die Elemente, Attribute und Beziehungen des Benutzungsmodells festgelegt. Das Benutzungsmodell und damit die Struktur des useML-Schemas sind dabei so aufgebaut, wie in Abb. 2 grafisch dargestellt.

Wie in Abb. 3 zu erkennen ist, besteht das Element „benutzungsmodell“ neben einer „bezeichnung“ im Wesentlichen aus einer beliebigen Anzahl von „benutzungsobjekten“. Jedes „benutzungsobjekt“ kann wiederum weiter in elementare Benutzungsobjekte oder Benutzungsobjekte strukturiert werden. Anhand dieser hierarchischen Struktur wird die natürliche Bearbeitungsweise von Aufgaben durch den Menschen abgebildet [Hac95].

UseML wurde bereits in zahlreichen Projekten (z.B. [Mei07]) über verschiedene Domänen hinweg erfolgreich eingesetzt. Dabei stellten sich Probleme, bzgl. der Modellierungsmächtigkeit heraus, z.B. verfügt useML in der Version von [Reu03] weder über Temporaloperatoren zur zeitlichen Gliederung der Aufgaben noch über Konditionen. In der erweiterten Version von useML (Abb. 3) können Temporaloperatoren zwischen Ele-

menten gleicher Ebene definiert werden. Ebenfalls können nun Mehrfachausführungen definiert, oder Bedingungen (Vor-, Während-, Nachbedingung) an die Aufgaben angehängen werden.

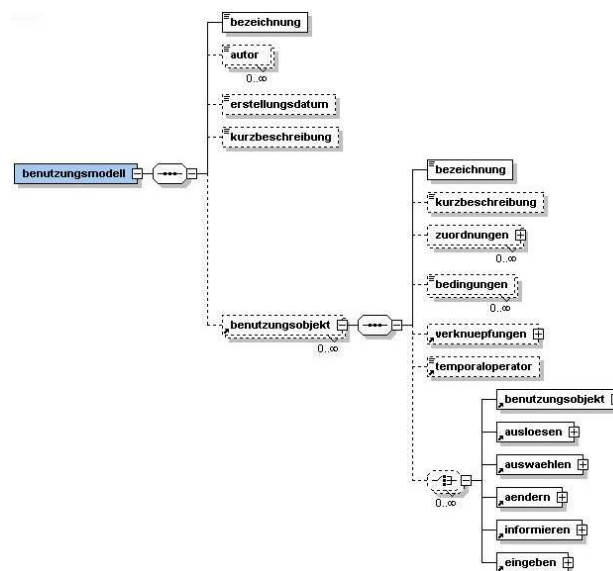


Abbildung 3. Aufbau des erweiterten Benutzungsmodells

## 5 Zusammenfassung und Ausblick

Die Modellierung der Applikationslogik während des Softwareentwicklungsprozesses ist mittlerweile als Standard anzusehen, wobei im Gegensatz dazu die Modellierung von Bedienoberflächen noch kein alltägliches Thema im Rahmen aktueller Softwareentwicklungsprozesse ist. Wie auch bei der Modellierung der Applikationslogik ist es mit einem MBUID Ansatz bei Bedienoberflächen möglich, plattformübergreifend und damit abstrakt zu modellieren. Klarer Vorteil der abstrakten Modellierung ist eine automatisierbare Erstellung von Quellcode. Bei Bedienoberflächen ist es somit möglich, diese aufbauend auf dem Aufgabenmodell, z.B. in useML spezifiziert, in den späteren Phasen in Quellcode zu transformieren. Durch Verwendung des nutzerzentrierten Useware-Engineering-Prozesses, sowie der Verwendung formaler Modelle ist es somit möglich, konsistente Bedienoberflächen für verschiedene Plattformen zu entwickeln.

Bei aktuellen modellbasierten Entwicklungsprozessen wird das Dialogmodell manuell aus dem Aufgabenmodell extrahiert. Der Ansatz, der automatischen Generierung des Dialogmodells aus dem Aufgabenmodells, wie in [Luy03] gezeigt, kann an useML adaptiert werden. Ein wesentlicher Vorteil liegt dabei in der konsistenten und korrekten Umsetzung der Informationen aus dem Aufgabenmodell. Um die automatisierte Ableitung des Dialogmodells aus dem Aufgabenmodell zu erzeugen, wurde die Sprache useML u.a.

um temporale Operatoren erweitert. Des Weiteren muss zukünftig eine Integration zwischen den Entwicklungsprozessen von Software und Useware geschaffen werden. Dies könnte z.B. über Modelltransformationen nach UML [Nöb05] realisiert werden.

## Literaturverzeichnis

- [Car83] Card, S.; Moran, T.; Newell, A.: „The Psychology of Human Computer Interaction“. Lawrence Erlbaum Associates, Hillsdale (USA) (1983).
- [Gör07] Görlich, D.; Breiner, K.: „Useware modelling for ambient intelligent production environments“. Workshop: Model-Driven Development of Advanced User Interfaces, MODELS 2007 (2007).
- [Gru06] Gruhn, V.; Pieper, D.; Röttgers, C.: „MDA - effektives Software-Engineering mit UML 2 und Eclipse“. Springer (2006).
- [Hac95] Hacker, W.: „Arbeitsfähigkeitsanalyse - Analyse und Bewertung psychischer Arbeitsanforderungen“. Heidelberg: Roland Asanger Verlag (1995).
- [Luy03] Luyten, K.; Clerckx, T.; Coninx, K.; Vanderdonckt, J.: „Derivation of a Dialog Model from a Task Model by Activity Chain Extraction“. In: Interactive Systems: Design, Specification and Verification, 10<sup>th</sup> International Workshop DSV-IS (2003).
- [Luy04] Luyten, K.: „Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development“. PhD thesis, transnational University Limburg: School of Information Technology (2004).
- [Mei07] Meixner, G.; Thiels, N.; Klein, U.: „SmartTransplantation – Allogeneic Stem Cell Transplantation as a Model for a Medical Expert System“. In: Proceeding of Usability & HCI for Medicine and Health Care (USAB) 2007, LNCS 4799, S. 306-317 (2007).
- [Mor04] Mori, G.; Paternò F.; Santoro, C.: „Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions“. In: IEEE Transactions on Software Engineering, Bd. 30, Nr. 8, S. 507-520 (2004).
- [Mye00] Myers, B.; Hudson, S.; Pausch R.: „Past, Present, Future of User Interface Tools“. In: ACM Trans. Computer-Human Interaction, Bd. 7, Nr. 1, S. 3-28 (2000).
- [Nöb05] Nöbrega, L.; Nunes, N.J.; Coelho, H.: „Mapping ConcurTaskTrees into UML 2.0“. In: Interactive Systems: Design, Specification and Verification, 12<sup>th</sup> International Workshop DSV-IS (2005).
- [OMG07] Object Management Group (OMG): UML Resource Page. [www.uml.org](http://www.uml.org), 02.10.2007.
- [Pat00] Paternò, F.: „Model-based design and evaluation of interactive applications“. Springer (2000).
- [Pue97] Puerta, A.: „A Model-Based Interface Development Environment“. In: IEEE Software, Bd. 14, Nr. 4, S. 40-47 (1997).
- [Pue99] Puerta, A.; Eisenstein, J.: Towards a General Computational Framework for Model-Based Interface Development Systems. In: Proc. of the 4<sup>th</sup> International Conference on Intelligent User Interfaces (1999).
- [Reu03] Reuther, A.: „useML – Systematische Entwicklung von Maschinenbediensystemen mit XML“. Fortschritt-Berichte pak, Bd. 8, TU Kaiserslautern (2003).
- [Thi07] Thiels, N.; Zühlke, D.: „Personalisation of user interfaces by elevating individual differences“. In: Ergonomic Society Annual Conference 2007, S. 266-271 (2007).
- [Træ02] Trættemberg, H.: Model-based User Interface Design. (Ph.D. thesis) Trondheim/Norwegen: Information Systems Group, Department of Computer and Information Sciences, Norwegian University of Science and Technology, 2002.
- [Van93] Vanderdonckt, J.; Bodart, F.: „Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection“. In: ACM Conference on Human Aspects in Computing Systems InterCHI'93, Addison Wesley, S. 424-429 (1993).
- [Van03] Vanderdonckt, J.; Limbourg, Q.; Florins, M.: „Deriving the Navigational Structure of a User Interface“. Human-Computer Interaction INTERACT '03: IFIP TC13 International Conference on Human-Computer Interaction (2003).
- [Züh99] Zühlke, D.; Wahl, M.: „Hardware, Software – Useware“. In: Elektronik, Nr. 23, S. 54-62 (1999).
- [Züh04] Zühlke, D.: „Useware-Engineering für technische Systeme“. Springer (2004).
- [Züh07] Zühlke, D.: „Model-based Development of User Interfaces: A New Paradigm in Useware Engineering“. Keynote Speech, 10<sup>th</sup> IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine-Systems (2007).