Modellbasierte Entwicklung von Benutzungsschnittstellen

Gerrit Meixner

Einleitung

Die Optimierung der Interaktion zwischen Menschen und Computern ist seit vielen Jahrzehnten ein essenzielles Thema bei der täglichen Arbeit. Auch heute erfolgt die Interaktion in den meisten Fällen noch über grafische Benutzungsschnittstellen. Die Interaktion des Menschen mit einem Computer über multimodale Benutzungsschnittstellen (Kombination und parallele Nutzung verschiedener Sinneskanäle zur Übermittlung von Informationen) gewinnt allerdings zunehmend an Bedeutung bei der Bedienung von Computern bzw. interaktiven Systemen.

Dass die Entwicklung von Benutzungsschnittstellen für interaktive Systeme eine zeitaufwendige und somit kostspielige Aufgabe ist, zeigt eine Untersuchung von Myers und Rosson [8]. Bei der Analyse einer Reihe verschiedener Applikationen fanden sie heraus, dass

- ca. 48 % des Quellcodes,
- ca. 45 % der Entwicklungszeit,
- ca. 50 % der Implementierungszeit und
- ca. 37 % der Wartungszeit

für Aspekte der Benutzungsschnittstelle benötigt werden. Dass der zeitliche Aufwand für die Implementierung der Benutzungsschnittstelle auch heute noch bei mindestens 50 % liegt, wird durch Petrasch [10] darin begründet, dass die Verbreitung interaktiver Systeme und die Anforderungen an die Benutzungsschnittstelle stark gestiegen sind. Des Weiteren muss betont werden, dass immer kürzere Softwareentwicklungszyklen in der Industrie gefordert werden, um einerseits Kosten zu sparen und andererseits schnell neue Produkte auf den Markt zu bringen.

Zudem stellt die stetig wachsende Zahl verschiedenster (mobiler) Geräte eine Herausforderung für Entwickler dar. Es muss nicht nur eine Applikation bzw. eine Benutzungsschnittstelle für eine Plattform, sondern eine Applikation mit einer Vielzahl von Benutzungsschnittstellen (mit verschiedenen Bildschirmgrößen, Grafikbibliotheken, Ressourcen etc.) für verschiedene Geräte entwickelt werden [5]. Erstens ist es äußerst schwierig, definierte Anforderungen zu erfüllen und gleichzeitig, über mehrere Plattformen hinweg, konsistent in Bezug auf Funktionalität bzw. Präsentation zu sein. Zweitens impliziert die Entwicklung von Anwendungen für verschiedene Plattformen auch, dass Entwickler eine Vielzahl von Programmiersprachen und Methoden beherrschen müssen.

Um den o. g. Anforderungen gerecht zu werden, spielen Aspekte wie Wiederverwendbarkeit, Flexibilität und Plattformunabhängigkeit eine entscheidende Rolle bei der Entwicklung aktueller Benutzungsschnittstellen [1]. Da der immer wiederkehrende Entwicklungsaufwand für Einzellösungen für eine spezifische Plattform oder Modalität zu

DOI 10.1007/s00287-011-0546-7

Gerrit Meixner
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI),
Forschungsbereich Innovative Fabriksysteme (IFS),
Trippstadter Strasse 122, 67663 Kaiserslautern
E-Mail: Gerrit.Meixner@dfki.de

*Vorschläge an Prof. Dr. Frank Puppe
<puppe@informatik.uni-wuerzburg.de> oder
Prof. Dr. Dieter Steinbauer <dieter.steinbauer@schufa.de>

Alle „Aktuellen Schlagwörter" seit 1988 finden Sie unter:
www.ai-wuerzburg.de/as

hoch ist, bietet sich ein modellbasierter Ansatz (engl. Model-Based User Interface Development, MBUID) für die Entwicklung von Benutzungsschnittstellen an [11].

Die Kernmodelle

Ein Modell hat dabei nach Charwat [2] das Ziel, Zusammenhänge und Zusammenwirken von Entitäten in der Realität durchschaubar und Entwicklungen bzw. Verhaltensweisen vorhersehbar zu machen. Dies führt dazu, dass komplexe Sachverhalte gegliedert und in eine Ordnung gebracht werden müssen. Insbesondere werden Abstraktionen oder Vereinfachungen (z. B. durch Vernachlässigen von Faktoren) der Realität ohne Übernahme der Komplexität durchgeführt.

Im Rahmen eines modellbasierten Ansatzes werden dabei verschiedene Aspekte einer Benutzungsschnittstelle mit unterschiedlichen (meist) deklarativen Modellen abgebildet [12]. Die Gesamtheit der zu verwendenden Modelle wird von Puerta als das „Interface Model" beschrieben [11] und kann aus Sicht der Benutzungsschnittstelle in abstrakte und konkrete Modelle eingeteilt werden. Abstrakte Modelle sind nicht direkter Bestandteil der Benutzungsschnittstelle, sondern beinhalten Informationen, die zur automatischen Generierung von Benutzungsschnittstellen erforderlich sind. Konkrete Modelle hingegen spezifizieren direkt Teile der sicht- und erlebbaren Benutzungsschnittstelle.

Die Entwicklung einer Benutzungsschnittstelle wird im Wesentlichen durch drei partielle Kernmodelle des „Interface Models" getrieben: Aufgaben-, Dialog- und Präsentationsmodell [5]. Das Aufgabenmodell beschreibt eine Struktur, welche die Aufgaben, Tätigkeiten und Handlungen einzelner Nutzer der Benutzungsschnittstelle hierarchisch beschreibt, die Abfolge der Aufgaben zeitlich mittels Temporaloperatoren in Relation setzt und durch Konditionen feiner spezifiziert. Das Dialogmodell beschreibt die Interaktionen des Nutzers mit der Benutzungsschnittstelle und deren Auswirkung ohne eine konkrete Technik bzw. „Look&Feel" zu berücksichtigen. Das Dialogmodell verbindet das Aufgabenmodell mit dem Präsentationsmodell. Es stellt den dynamischen Teil bei der Ausführung von Aufgaben des Nutzers (Aufgabenmodell) an der Benutzungsschnittstelle dar und ist daher eng mit dem Präsentationsmodell verknüpft. Die eigentliche Darstellung bzw. Präsentation der Benutzungsschnittstelle wird durch eine hierarchische Komposition von Interaktionsobjekten (widgets), dem sogenannten Präsentationsmodell, dargestellt.

Weitere relevante Bestandteile des „Interface Models" sind das Domänenmodell (kann als Teil der Applikationslogik angesehen werden, da es Objekttypen und Objektmethoden repräsentiert, die genutzt oder durch die Benutzungsschnittstelle unterstützt werden sollen), das Plattformmodell (spezifiziert die Zielplattform, auf der die Benutzungsschnittstelle eingesetzt werden soll. Merkmale einer Plattform sind u. a. Eingabe- und Ausgabemöglichkeiten, Softwarestrukturen oder Hardwarerahmenbedingungen), das Benutzermodell (repräsentiert verschiedene Benutzergruppen oder Personen) sowie das Kontextmodell (beschreibt Kontextfaktoren, wie z. B. die Lokalisation des Benutzers, die zur Adaption der Benutzungsschnittstelle zur Laufzeit führen).

Vorgehensweise und Referenzarchitektur

Im Bereich der Softwareentwicklung liegt mit der Model Driven Architecture (MDA) ein Standard der Object Management Group (OMG) vor, der die Repräsentation von Software von der Programmcodeebene auf die Modellebene hebt. Um die Komplexität auf Modellebene zu reduzieren, werden Modelle verschiedener Abstraktionsebenen unterschieden, die aufeinander aufbauen. Der Ansatz folgt dem Grundsatz, dass die Spezifikation einer Komponente unabhängig von deren technischen Umsetzung zu beschreiben ist. Dabei kann die Realisierung eines plattformunabhängigen Modells durch die Wahl einer Plattform, also der konkreten technischen Umsetzung, teilweise oder vollständig automatisiert erfolgen. Mit der MDA wird das Modell zum zentralen Element des Softwareentwicklungsprozesses.

Aufgrund fehlender Referenzarchitekturen bei der modellbasierten Entwicklung von Benutzungsschnittstellen und dem fehlenden Konsens über die Verwendung geeigneter Modelle [5], wurde die Arbeit an einer entsprechenden Metaarchitektur im Rahmen des europäischen CAMELEON-Projektes (*C*ontext *A*ware *M*odelling for *E*nabling and *L*everaging *E*ffective interacti*ON*) begonnen. Ergebnis dieses Projektes ist das CAMELEON Reference Framework [1], was sich in den vergangenen Jahren im Bereich des MBUID etabliert hat. Vereinfacht gesehen, basiert dieses Framework auf vier ver-

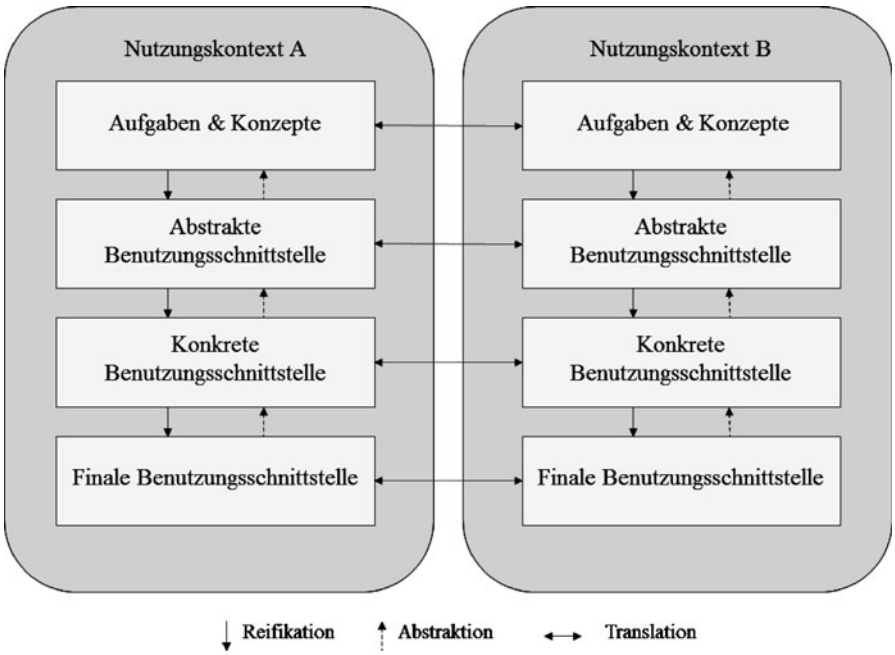

Abb. 1 Vereinfachtes CAMELEON Reference Framework (vgl. [1])

schiedenen Schichten (vgl. Abb. 1), wobei die oberen drei Schichten (Aufgaben und Konzepte, Abstrakte Benutzungsschnittstelle, Konkrete Benutzungsschnittstelle) den o. g. (Kern-)Modellen entsprechen. Die unterste Schicht des Frameworks ist die finale Benutzungsschnittstelle, die letztlich für ein bestimmtes Gerät erzeugt wird und meist im Quellcode der Zielplattform ausgedrückt ist.

Neben der vertikalen Ebene, die den Weg von abstrakten zu immer konkreteren Modellen beschreibt, gibt es auch eine horizontale Ebene, die ein Modell abhängig vom gegenwärtigen Kontext anpassen kann. So kann sich beispielsweise für ein Aufgabenmodell die Reihenfolge der Unteraufgaben ändern, wenn der Nutzer sich an unterschiedlichen Orten befindet (Kontextadaptivität).

Die durchgängige Transformation von Modellen, die im Rahmen der modellgetriebenen Entwicklung von Software verfolgt wird, ist auch bei MBUID ein wichtiges und zentrales Element, um

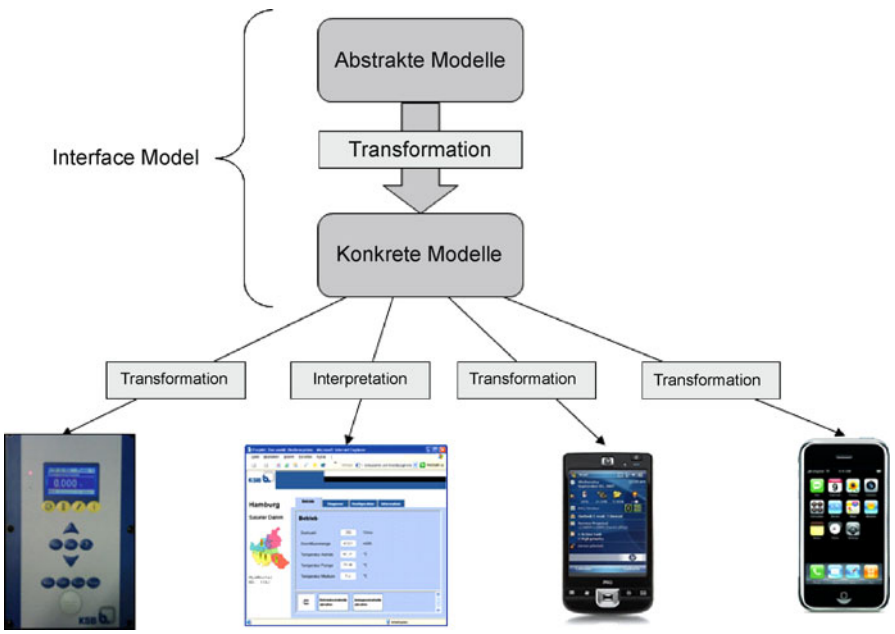

Abb. 2 Entwicklung von Benutzungsschnittstellen für verschiedene Plattformen mittels MBUID

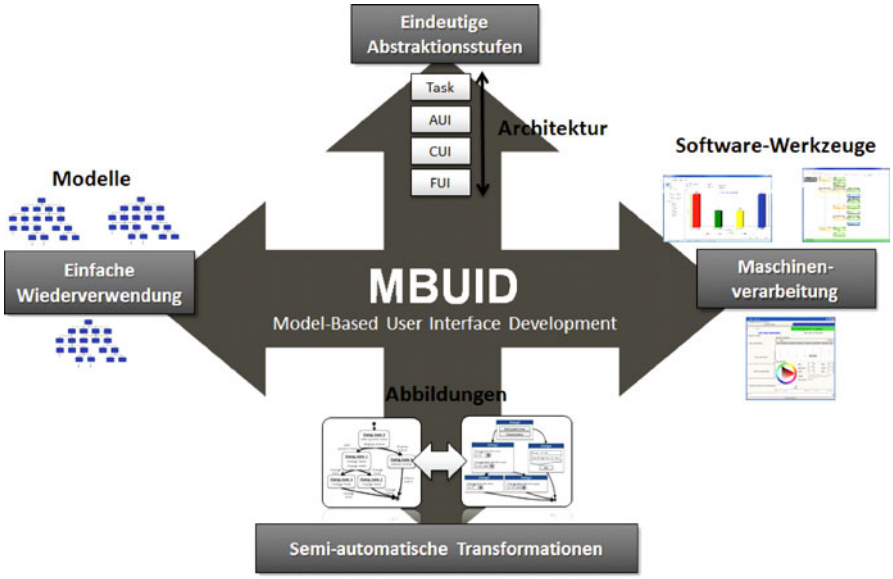

Abb. 3 Charakterisierung des MBUID

den Entwicklungsprozess effizienter zu gestalten. Das Paradigma empfiehlt einen Top-down-Ansatz, beginnend mit abstrakten Beschreibungen relevanter Informationen über die Benutzungsschnittstelle. Diese abstrakten Beschreibungen werden interpretiert oder transformiert, um eine partielle oder vollständige Generierung von Quellcode zu erreichen. Somit ist es nicht mehr erforderlich, für jede Zielplattform der Benutzungsschnittstelle separaten Quellcode zu entwickeln (vgl. Abb. 2).

Bewertung und Ausblick

Abbildung 3 fasst die charakterisierenden Eigenschaften und Aussagen über MBUID grafisch zusammen.

Durch die Aufteilung einer Benutzungsschnittstelle in verschiedene Abstraktionsstufen, Ebenen und Modelle wird eine strukturierte Entwicklung von Benutzungsschnittstellen ermöglicht, da die verschiedenen Modelle gezielt in der jeweiligen Phase eines modellbasierten Entwicklungsprozesses erstellt, bearbeitet und verwendet werden können. Ein weiterer wichtiger Punkt dieser Vorgehensweise ist die strikte Trennung zwischen der Darstellung der Benutzungsschnittstelle und der Applikationslogik, wie zum Beispiel im Model-View-Controller-Paradigma (MVC) realisiert. Eine Benutzungsschnittstelle (View) kann somit von einem Designer entwickelt und von einem Informatiker mit Funktionen (Model, Controller) angereichert werden. Dadurch können sich Designer und Informatiker auf ihre jeweiligen Spezialgebiete beschränken. Ein bekanntes Beispiel dazu ist die Verwendung von Microsofts XAML. Ein Informatiker nutzt Visual Studio zur Entwicklung der Applikationslogik, wohingegen ein Designer Blend zur Entwicklung der Darstellung (Dialog- und Präsentationsmodell) verwendet. XAML ist dabei die gemeinsame Sprache bzw. das gemeinsame Modell der verschiedenen Entwicklergruppen.

Bereits seit mehr als 20 Jahren wird nun auf dem Gebiet der modellbasierten Entwicklung von Benutzungsschnittstellen geforscht [11, 12]. Seitdem wurde eine Vielzahl an Architekturen, Sprachen und Werkzeugen entwickelt. Allerdings konnten sich nur sehr wenige Ansätze in Forschung und Praxis etablieren [3]. Viele Sprachen zur Beschreibung von Benutzungsschnittstellen bzw. deren Modelle basieren auf XML – der eXtensible Markup Language. Die Vorteile des Einsatzes von XML-basierten Sprachen liegen insbesondere in der Plattformunabhängigkeit, der klaren (hierarchischen) Struktur, der Lesbarkeit und nicht zuletzt in der einfacheren Transformierbarkeit (z. B. mit XSLT).

Zu den bekanntesten wissenschaftliche Ansätzen im Bereich der Aufgabenmodellierung zählen bspw. die Useware Markup Language (useML) [6] sowie die ConcurTaskTree-Notation (CTT) [9]. Im Bereich der Dialog- und Präsentationsmodellierung haben sich beispielsweise die Dialog and Interface Specification Language (DISL) [6] oder der OASIS Standard der User Interface Markup Language (UIML) [7] etabliert. Des Weiteren existieren andere Sprachen wie beispielsweise die User Interface Extensible Markup Language (UsiXML), die das gesamte CAMELEON Reference Framework in einem monolithischen Ansatz versucht abzudecken. Ein erweiterter Überblick zum Stand der Technik so-

wie aktuelle Entwicklungen des MBUID findet sich in [4].

Im Bereich kommerziell erhältlicher Modellierungssprachen entwickelt die Firma RedWhale (http://www.redwhale.com) die eXtensible Interface Markup Language (XIML) sowie die Werkzeuge UI Pilot und UI Fin. Die Firma Harmonia (http://www.liquidappsworld.com) vertreibt das Werkzeug LiquidApps, was für die Modellierung der Benutzungsschnittstelle mittels UIML [7] eingesetzt wird.

Dem Autor dieses Beitrages sind aktuell keine vollständigen Softwarewerkzeugketten bekannt, mit denen nutzerzentriert und auf Basis von Modellen multimodale Benutzungsschnittstellen für verschiedene Plattformen durchgängig entwickelt werden können. Daher ist es wichtig, dass die zur Entwicklung genutzten Modelle theoretisch fundiert entwickelt und in der Praxis evaluiert werden. Aufgrund technischer und technologischer Fortschritte ist es weiterhin notwendig, Grundlagenarbeit im Bereich der modellbasierten Entwicklung von Benutzungsschnittstellen zu leisten.

Insbesondere ist es erforderlich, Entwicklern intuitive Softwarewerkzeuge an die Hand zu geben, die es erlauben, schnell und einfach Benutzungsschnittstellen für verschiedene Plattformen ohne hohen Mehraufwand (bei gleicher Qualität wie konventionelle Ansätze) generieren zu können. Zur Akzeptanzerhöhung des MBUID-Ansatzes ist es darüber hinaus notwendig, die Qualität der automatisch generierten Benutzungsschnittstellen zu steigern. Erfolg versprechende Ansätze und Konzepte kommen aus dem Bereich der (generativen) HCI-Patterns [10].

Aufgrund der Wichtigkeit und Aktualität der Thematik werden zurzeit Standardisierungsaktivitäten beim World Wide Web Consortium (W3C) durchgeführt, an dem der Autor dieses Beitrages aktiv beteiligt ist.

Literatur

1. Calvary G et al. (2003) A unifying reference framework for multi-target user interfaces. Interact Comput 15(3):289–308
2. Charwat HJ (1994) Lexikon der Mensch-Maschine-Kommunikation. Oldenbourg
3. Guerrero-García J et al. (2009) A Theoretical Survey of User Interface Description Languages: Preliminary Results. In: Proc of the Latin American Web Congress, Mérida, Mexico, 9–11 November 2009, pp 36–43
4. Hußmann H, Meixner G, Zühlke D (2011) Model-Driven Development of Advanced User Interfaces. Springer, Heidelberg
5. Luyten K (2004) Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development. Doctoral Thesis, Transnationale Universiteit Limburg
6. Meixner G (2010) Entwicklung einer modellbasierten Architektur für multimodale Benutzungsschnittstellen. Dissertation, Fortschritt-Berichte pak, Bd 21. Technische Universität Kaiserslautern, Kaiserslautern
7. Meixner G, Schäfer R (2009) Modellbasierte Entwicklung von Benutzungsschnittstellen mit UIML. i-com 8(1):60–67
8. Myers B, Rosson MB (1992) Survey on User Interface Programming. In: Proc of the 10th Annual CHI Conference on Human Factors in Computing Systems, Monterey, CA, USA, 3–7 May 1992, pp 195–202
9. Paternò F (2000) Model-Based Design and Evaluation of Interactive Applications. Springer, London
10. Petrasch R (2007) Model based user interface design: model driven architecture und HCI patterns. GI Softwaretechnik-Trends, Mitt Ges Inform 27(3):5–10
11. Puerta A (1997) A model-based interface development environment. IEEE Softw 14(4):40–47
12. Schlungbaum E, Elwert T (1996) Automatic user interface generation from declarative models. In: Proc of of the 2nd International Workshop on Computer-Aided Design of User Interfaces, Namur, Belgium, 5–7 June 1996, pp 3–18