Entwicklung eines Transformationsprozesses zur modellbasierten Entwicklung von multimodalen Benutzungsschnittstellen

Marc Seißler und Gerrit Meixner

Schlüsselwörter: Modellbasierte Entwicklung von Benutzungsschnittstellen, useML, DISL, Multimodalität, Transformationsprozess

Zusammenfassung

Schon seit langem wird versucht, die Kommunikation zwischen Mensch und Maschine zu vereinfachen. Vor allem durch die gezielte Einbeziehung aller Sinneskanäle des Menschen soll die Interaktion zwischen Mensch und Maschine erleichtert werden. Modelle, wie z.B. das Aufgaben-, Dialog- und Präsentationsmodell, erweisen sich hier als geeignetes Mittel für die plattform- und modalitätenunabhängige – sowie vor allem nutzerzentrierte – Beschreibung von zukünftigen Benutzungsschnittstellen. Zusätzlich ermöglichen Modelle die automatisierte Erzeugung von ersten generischen Benutzungsschnittstellen-Prototypen. Im Rahmen dieses Beitrages wird die Anbindung des, in der Useware-Markup Language (useML) spezifizierten, Benutzungsmodells an die Ebene der abstrakten Benutzungsschnittstelle, welche in der modalitäten- und plattformunabhängigen Dialog and Interface Specification Language (DISL) definiert ist, sowie deren praktische Umsetzung in einem anwendungsnahen Werkzeug vorgestellt.

Abstract

To improve Human-Machine Interaction future User Interfaces (UI) will utilize many different perceptional channels of the users sensory system for information exchange. To ensure the usability of those multimodal UIs a systematic engineering process is demanded. Models, such as the task-, dialog- and presentation model can be used for a platform-, modality-, as well as user-centered, UI description. Additionally models enable the generation of generic UI prototypes. In this paper a transformation-process is introduced, which transforms a use model, specified with the Useware-Markup Language (useML), into a platform- and modality-independent dialog model, described with the Dialog and Interface Specification Language (DISL).

Einleitung

Mit der Useware-Markup Language (useML) (Reuther, 2003) wurde eine XML-basierte Beschreibungssprache für die frühe Phase des Useware-Engineering-Prozesses (Zühlke, 2004) vorgestellt. Diese erlaubt die Aufgaben und Handlungen, die ein Benutzer mit der Benutzungsschnittstelle ausführt, in einem Benutzungsmodell (BM), mit Hilfe von Benutzungsobjekten (BOs) und elementaren Benutzungsobjekten (eBOs), hierarchisch zu beschreiben. Mit der Einführung von useML 2.0 (Meixner & Görlich, 2008) wurde eine semantisch erweiterte Version der Sprache vorgestellt. Eine der wichtigsten Neuerungen in useML 2.0 stellt die Einführung der Temporaloperatoren dar. Mit diesen lassen sich komplexe Ausführungsreihenfolgen der im BM repräsentierten Aufgaben spezifizieren. Insgesamt existieren fünf verschiedene Temporaloperatoren mit folgender, absteigender Priorität: Auswahl, Reihenfolgeunabhängigkeit, Nebenläufigkeit, Deaktivierung, Sequenz. Durch die Einführung der Temporaloperatoren wurde die Sprache für die semi-automatische Transformation von Benutzungsmodellen vorbereitet.

In diesem Beitrag wird ein neu entwickelter Transformationsprozess vorgestellt, der die Anbindung von useML an die Dialog and Interface Specification Language (DISL) (Bleul et al., 2004) umsetzt. Weiterhin wird die weiterentwickelte Version des useML-Editors und Simula-

tors (Udit) vorgestellt, der den vorgestellten Transformationsprozess implementiert. Abschließend wird ein Ausblick auf die weiteren Entwicklungen im Bereich der modellbasierten Werkzeugkette gegeben.

Der Transformationsprozess

Um den Entwickler beim Übergang von der Strukturgestaltungsphase in die Gestaltungsphase des Useware-Entwicklungsprozesses adäquat zu unterstützen wurde der – in Abbildug 1 dargestellte – vier-phasige, semi-automatische Transformationsprozess entwickelt. Dieser Prozess adaptiert dabei den in der TERESA-Entwicklungsmethodik (Berti et al., 2004) eingesetzten Transformationsprozess und ermöglicht die Generierung von ersten, multimodalen Benutzungsschnittstellenprototypen.

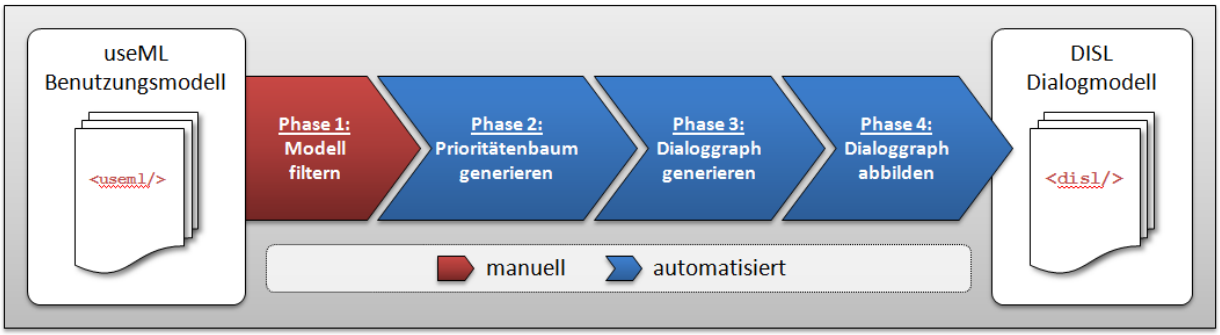

Abb. 4: Der vier-phasige Transformationsprozess

Phase 1: Filtern des Benutzungsmodells

In der ersten Phase des Transformationsprozesses findet eine Verfeinerung des BMs statt. Hier kann das BM manuell mit Hilfe von Filtern beispielsweise für eine bestimmte Zielplattform, oder Benutzergruppe, anpasst werden. In useML können Filterregeln mit Hilfe der Zuordnungen spezifiziert werden.

Phase 2: Erstellen des binären Prioritätenbaums

Die zweite Phase dient der Erzeugung einer, für die Transformation effizienteren, Datenstruktur – dem sogenannten „binären Prioritätenbaum“ (Mori et al., 2002). Der binäre Prioritätenbaum ist ein BM, bei dem jedes BO genau zwei Kinder besitzt. Weiterhin gilt, dass die Blätter dieses BMs durch eBOs repräsentiert werden. Die hierarchische Struktur des binären Prioritätenbaums leitet sich dabei aus den Prioritäten der Temporaloperatoren des zugrundeliegenden BMs ab. Im linken Teil von Abbildung 2 wird ein binärer Prioritätenbaum für ein einfaches BM einer „Pumpe“ dargestellt.

Phase 3: Generierung des Dialoggraphen

In der dritten Phase wird aus dem übergebenen binären Prioritätenbaum der – im rechten Teil der Abbildung 2 dargestellte – Dialoggraph generiert. Der Dialoggraph repräsentiert alle möglichen Ausführungsreihenfolgen der eBOs und somit das dynamische Verhalten des BMs. Die vom Benutzer zu einem Zeitpunkt ausführbaren eBOs werden in einem Dialogzustand zusammengefasst. Mit Hilfe von gerichteten Transitionen wird die Navigation zwischen den Zuständen beschrieben. Für die Erzeugung des Dialoggraphen wurde ein Algorithmus entwickelt, welcher den Dialoggraphen schrittweise, rekursiv erzeugt. Der Algorithmus setzt sich dabei aus zwei Phasen zusammen: einer „Top-Down-Analyse“ des BMs für die Identifizierung eines

Dialogzustandes, sowie einer „Bottom-Up-Aktualisierung“, um den Ausführungszustand des BMs zu aktualisieren.

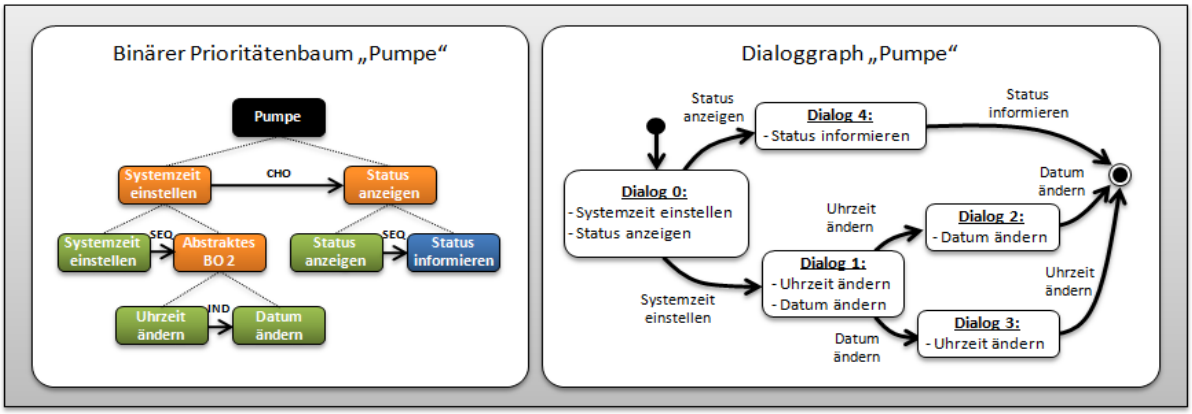

Abb. 5: Binärer Prioritätenbaum (links) und Dialoggraph (rechts) eines Beispielmodells

Im ersten Schritt wird in der Top-Down-Analyse der Prioritätenbaum von der Wurzel bis in die Blätter nach ausführbaren eBOs durchsucht. Dabei entscheidet die Semantik der Temporaloperatoren, welche eBOs in den aktuellen Dialogzustand aufgenommen werden müssen. Das Ergebnis eines Top-Down-Durchlaufs des Algorithmus ist ein eindeutiger Dialogzustand des Dialoggraphen.

Im darauffolgenden Schritt findet eine „simulierte“ Ausführung der eBOs des Dialogzustandes statt. Durch diese werden die Nachfolgedialogzustände und die Transitionen zu diesen bestimmt. Nach der simulierten Ausführung eines eBOs muss der Zustand des Prioritätenbaums von den Blättern bis in die Wurzel aktualisiert werden. Durch den anschließenden rekursiven Aufruf der „Top-Down-Analyse“ kann der Nachfolgedialogzustand identifiziert werden, der durch das ausgewählte eBO erreicht werden kann. Wurden alle eBOs im Prioritätenbaum ausgeführt, so terminiert der Algorithmus und der vollständige Dialoggraph des BMs wurde erzeugt.

Phase 4: Abbildung des Dialoggraphen nach DISL

In der letzten der vier Transformationsphasen wird der Dialoggraph nach DISL abgebildet. Durch die Verwendung generischer Interaktionsobjekte, die in verschiedensten Modalitäten eine konkrete Repräsentation besitzen, eignet sich DISL für die plattform-, sowie modalitätenunabhängige Beschreibung von Benutzungsschnittstellen.

Bei der Abbildung des Dialoggraphen nach DISL werden dessen Dialogzustände auf Interfaces abgebildet. Interfaces stellen in DISL einen Ausschnitt der Gesamt-Benutzungsschnittstelle dar und ermöglichen somit die Modularisierung der Benutzungsschnittstelle. Dies kann vor allem auf mobilen Endgeräten – wie z.B. Mobiltelefonen – ein mögliches „Überladen“ der Benutzungsschnittstelle verhindern. Anschließend werden die eBOs eines Dialogzustandes durch ein „Präsentations-Mapping“ (Puerta & Eisenstein, 1999) auf passende generische Interaktionsobjekte in DISL abgebildet.

Die Transitionen des Dialoggraphen werden in der Verhaltensbeschreibung des Interfaces mittels Regeln, Transitionen und Aktionen beschrieben. Da die Dialoggraph-Transitionen den „Fortschritt“ der Benutzungsschnittstelle repräsentieren, werden diese ebenfalls in DISL abgebildet. Dazu wird für jedes generische Interaktionsobjekt eine Regel angelegt, die das Nachladen eines Interfaces beschreibt, wenn der Benutzer mit dem generischen Interaktionsobjekt (widget) interagiert. Neben dem dynamischen Verhalten der Benutzungsschnittstelle kann im Verhaltensabschnitt der Benutzungsschnittstelle ebenfalls die Anbindung an die Applikationslogik der Anwendung umgesetzt werden.

Udit – Der useML-Editor

Um die Entwickler bei der modellbasierten Benutzungsschnittstellenentwicklung mit einem adäquaten Werkzeug zu unterstützen, wurde in (Meixner et al., 2009) der graphische useML-Editor (Udit) vorgestellt. Der in diesem Artikel vorgestellte Transformationsprozess wurde vollständig in die aktuelle Version des Editors integriert. Durch die Einführung einer Filterfunktion lässt sich nun ein spezifisches Benutzungsmodell automatisch ableiten. Dieses kann anschließend weiter vom Entwickler verfeinert und dann automatisch nach DISL exportiert werden. Für die Darstellung des dynamischen Verhaltens des Benutzungsmodells wurde zusätzlich ein Benutzungsmodell-Simulator in Udit integriert. Dieser erlaubt dem Entwickler die frühzeitige Evaluierung des entwickelten Benutzungsmodells, unabhängig von der späteren Plattform und Modalität der finalen Benutzungsschnittstelle.

Zusammenfassung und Ausblick

In diesem Artikel wurde ein semi-automatischer Transformationsprozess zur Generierung von modalitätenunabhängigen Benutzungsschnittstellen-Prototypen vorgestellt. Als Zielsprache für die Beschreibung der Benutzungsschnittstellen-Prototypen wird die Dialog and Interface Specification Language (DISL) eingesetzt. Um den Entwicklern ein passendes Werkzeug zur Hand zu geben, wurde der Transformationsprozess vollständig in den useML-Editor und Simulator integriert.

Als nächstes Ziel ist der Übergang von der frühen Gestaltungsphase des Useware-Prozesses in Richtung der Realisierungsphase geplant. Durch diesen Schritt soll die Verfeinerung der abstrakten, in DISL beschriebenen, Benutzungsschnittstelle unterstützt werden.

Literatur

Berti, S., Correani, F., Mori, G., Paternò, F. & Santoro, C. (2004). TERESA: A Transformation-based Environment for Designing and Developing Multi-Device Interfaces. In ACM (Hrsg.), *CHI '04 extended abstracts on Human factors in computing systems* (S. 793-794). New York, NY, USA.

Bleul, S., Schäfer, R. & Müller, W. (2004). Multimodal Dialog Description for Mobile Devices. In *Workshop on XML-based User Interface Description Languages at AVI 2004*. Gallipoli, Italy.

Meixner, G. & Görlich, D. (2008). Aufgabenmodellierung als Kernelement eines nutzerzentrierten Entwicklungsprozesses für Bedienoberflächen. In *Fachtagung Modellierung*. Berlin.

Meixner, G., Seißler, M. & Nahler, M. (2009). Udit – A Graphical Editor For Task Models. In *Proc. of the 4th International Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI)*. Sanibel Island, USA.

Mori, G., Paternò, F. & Santoro, C. (2002). CTTE: support for developing and analyzing task models for interactive system design, *28*(8), 797-813.

Puerta, A. R. & Eisenstein, J. (1999). Towards a general computational framework for model-based interface development systems. In *IUI '99: Proceedings of the 4th international conference on Intelligent user interfaces* (S. 171-178). Redondo Beach California, USA.

Reuther, A. (2003). *useML - Systematische Entwicklung von Maschinenbediensystemen mit XML*. Fortschritt-Berichte pak, Bd. 8, Technische Universität Kaiserslautern.

Zühlke, D. (2004). *Useware-Engineering für technische Systeme*. VDI-Buch. Berlin, Heidelberg: Springer.