Marcel Dausend und Mark Poguntke

Ausführbare UML-Modelle multimodaler Interaktionsanwendungen

Executable UML Models of Multimodal Interactive Applications

Modellbasierte Entwicklung, Unified Modeling Language, Multimodalität, Interaktive Systeme

Zusammenfassung. Komplexe interaktive Anwendungen stellen sowohl für die Nutzer als auch für die Entwickler eine große Herausforderung dar, z.B. das Infotainmentsystem moderner Fahrzeuge. Dem Nutzer wird Multimodalität (Sprachbedienung neben der grafisch-haptischen Bedienung) angeboten, um die Ablenkung während des Fahrens zu reduzieren. Da die Kombination der verschiedenen Interaktionsmöglichkeiten die Komplexität der Anwendung deutlich erhöht, ist es notwendig, schon die Entwicklung dieser Systeme möglichst optimal zu unterstützen.
Wir stellen einen Ansatz zur Modellierung multimodaler interaktiver Anwendungen in Form einer domänenspezifischen Modellierungssprache vor. UML-Zustandsautomaten werden in erweiterter Form verwendet. Benutzerschnittstellen verschiedener Modalitäten werden zu ausführbaren Modellen kombiniert, so dass eine interaktive Simulation der Anwendung möglich wird, die die multimodale Benutzerschnittstelle einschließt.

Summary. Handling complex interactive applications is a challenge both for the user and the developer. A typical example is the infotainment system in modern cars. Multimodality (haptic and speech input) supports the user by reducing distraction while driving. The combination of these different modalities increases the system's complexity. Hence, an appropriate development method has to be provided to support developers in an optimal way.
We present an approach for modelling multimodal interactive applications with a domain-specific modelling language. UML state diagrams are used and extended to enable integration of user interfaces for different modalities to one multimodal, executable application.

1. Einleitung

Interaktive Anwendungen sind Teil unseres Alltags geworden und werden zunehmend komplexer. Dies zeigt etwa der zunehmende Funktionsumfang eines Mobiltelefons. Multimodalität, die kombinierte Verwendung beispielsweise grafisch-haptischer Interaktion mit Sprachein- und -ausgabe, wird eingesetzt, um die Bedienung zu vereinfachen und intuitiver zu gestalten. Die Entwicklung einer multimodalen Anwendung erfordert die Spezifikation jeder Modalität sowie die Verknüpfung der Modalitäten. Dies erhöht den Entwicklungsaufwand der Anwendung.

Wir stellen eine Modellierungsmethode für multimodale interaktive Anwendungen vor, deren Tauglichkeit und Angemessenheit im Rahmen einer Expertenevaluierung (Dausend und Poguntke 2010) gezeigt wurde. In diesem Beitrag erläutern wir detailliert den Formalismus und die Methode. Die Verwendung des Formalismus Unified Modeling Language (UML) (OMG 2010) und die Simulierbarkeit der Modelle sowie der Benutzerschnittstelle (UI, engl. für User Interface) der Anwendung stellen die zentralen Aspekte unseres Konzepts der Entwicklung multimodaler interaktiver Anwendungen dar.

2. Multimodale Interaktion am Beispiel Fahrzeug

Multimodalität ist besonders in Umgebungen interessant, in denen der Nutzer einer wichtigen Primäraufgabe nachgeht, und die Anwendung daher nicht zu jedem Zeitpunkt die volle Aufmerksamkeit des Nutzers erhalten kann. Ein typisches Beispiel dafür ist das Infotainmentsystem im Fahrzeug. Die Ein- und Ausgabe über Sprache, alternativ zur grafisch-haptischen Interaktion ermöglicht dem Fahrer, eine Anwendung zu bedienen, ohne die Hände vom Lenkrad zu nehmen. Heutzutage beschränkt sich Multimodalität im Fahrzeug auf Sprachbedienung sowie die

DOI 10.1524/icom.2011.0030

grafisch-haptische Bedienung über ein Ausgabedisplay und Eingabemöglichkeiten wie Touchscreen, Touchpad oder zentrale Dreh- und Drücksteller. Gestensteuerung oder natürlichsprachliche Dialoge sind Beispiele aktueller Forschungsthemen, um die Interaktionsmöglichkeiten im Fahrzeug und damit die Modalitäten zu erweitern. Dadurch kann natürlichere Interaktion erreicht werden, die besser auf die Bedürfnisse und Präferenzen des Nutzers abgestimmt ist.

3. Formalismus und Methode

Die UML ist der De-facto-Standard zur Modellierung von Anforderungen und zum Entwurf von Software. Einige existierende Ansätze zur Modellierung interaktiver Systeme basieren auf UML (vgl. Abschnitt 4). Diese Ansätze erweitern UML, da sie selbst nicht ausreichend geeignet ist, um Konzepte der Mensch-Maschine-Interaktion zu modellieren (vgl. Meixner 2010).

Dieses Kapitel beschreibt unsere Erweiterungen der UML um Konzepte für multimodale Interaktion im Sinne einer Domain Specific Modelling Language (DSML). Es enthält eine kurze technische Einführung in UML-Zustandsautomaten (in UML: State Machines), sowie *Profile* und *Stereotypen*, eine detaillierte Vorstellung des vorgeschlagenen Formalismus sowie eine Erklärung der einzelnen Konzepte der Methodik.

3.1 UML-Zustandsautomaten

Die Zustandsautomaten der UML stellen eine Erweiterung endlicher Zustandsautomaten, wie sie von Harel (1987) eingeführt wurden, dar. Ein Zustand beschreibt eine Invariante des beschriebenen Systems, so dass sich ein Systemzustand als Menge aktiver Zustände beschreiben lässt. Über Transitionen, die mit Bedingungen oder Ereignissen annotiert sind, finden Zustandswechsel statt, sobald eine Bedingung erfüllt ist oder ein Ereignis auftritt.

Einige Konzepte von UML-Zustandsautomaten können direkt zur Modellierung von interaktiven Anwendungen verwendet werden.

Transitionen werden mit Hilfe von *Junctions* und *Decisions* verzweigt oder zusammengeführt. Dies ermöglicht, Alternativen innerhalb eines Dialogablaufs zu modellieren oder alternative Dialogpfade zu vereinigen. Unter Verwendung von *Histories* können unterbrochene Interaktionen wiederhergestellt werden. Unterzustände, die beim Verlassen eines Zustands aktiv waren, werden beim Wiederbetreten erneut aktiviert. *EntryPoints* und *ExitPoints* können zur Definition einer Schnittstelle eines Zustandsautomaten verwendet werden, z.B. um Dialogbausteine zu realisieren. Zum Betreten und Verlassen von Zuständen mit parallelen Regionen können die Knoten *Fork* und *Join* verwendet werden. So lassen sich unterschiedliche Varianten des Betretens und Verlassens eines parallelen Zustands modellieren.

3.2 Profile und Stereotypen der UML

Die UML kann auf zwei Arten erweitert werden: Über *Metamodellerweiterung* oder *Profile und Stereotypen*. Profile werden von der UML angeboten: „A profile defines limited extensions to a reference metamodel with the purpose of adapting the metamodel to a specific platform or domain." (OMG 2010). Profile umfassen eine Menge von Stereotypen, die einzelne Aspekte einer Erweiterung beschreiben. Im Unterschied dazu ließe sich das UML-Metamodell entsprechend erweitern, z.B. durch die Verwendung von Vererbung, und Hinzufügen neuer Klassen.

Eine Metamodellerweiterung muss nicht garantieren, dass die Semantik für die Erweiterung zugrundeliegender UML-Elemente komplett erhalten bleibt. Dem gegenüber müssen Erweiterungen durch Profile dies garantieren. Zudem darf die Semantik eines Profils nie im Widerspruch zur UML-Semantik stehen. Ein weiteres Argument für die Nutzung von Profilen bietet die existierende Werkzeugunterstützung, die in der Regel nur Profile einschließt. Metamodellerweiterungen werden hingegen von keinem uns bekannten Werkzeug unterstützt.

Die Erweiterung durch *Profile* basiert auf folgenden Konstrukten: *Stereotypen* definieren die Erweiterung, z.B. durch Attribute oder Operationen. *Extensions* beschreiben die Beziehungen zwischen verschiedenen Elementen des Metamodells und/oder den Stereotypen. Beispielsweise beschreiben sie die Beziehung zwischen einem Stereotyp und einem Element des Metamodells, das durch diesen erweitert wird. *Constraints* definieren formale Einschränkungen auf Profilen. *Comments* dienen der Beschreibung der Semantik des Profils und können für Bemerkungen genutzt werden. Damit Modelle, die ein Profil verwenden, automatisch ausgeführt werden können, muss die Semantik formal beschrieben werden.

Selic (2010) beschreibt detailliert, unter welchen Umständen die Entwicklung einer DSML sinnvoll ist und gibt einen Leitfaden zur Erstellung von DSMLs unter Verwendung von UML-Profilen. Eine Anleitung zur Profilerstellung liefern auch (Fuentes-Fernandez und Vallecillo-Moreno 2004).

3.3 Formalismus

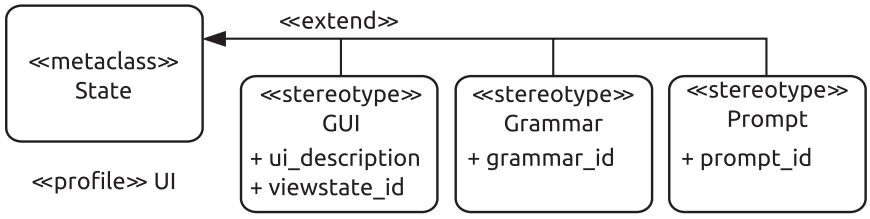
Als Grundlage zur Modellierung von multimodalen interaktiven Systemen verwenden wir UML-Zustandsautomaten (OMG 2010). Wir definieren ein Profil mit Stereotypen für grafisch-haptische Dialoge, diese beinhalten eine grafische Repräsentation als Ausgabe und die Eingabe über haptische Eingabegeräte, und Sprachdialoge mit Spracheingabe und -ausgabe.

Statische Aspekte beziehen sich auf die UI-Beschreibung, z.B. die Repräsentation des UIs zu einem bestimmten Zeitpunkt, eine bestimmte Phrase zur Ausgabe durch Sprachsynthese oder eine Grammatik zur Spracherkennung.

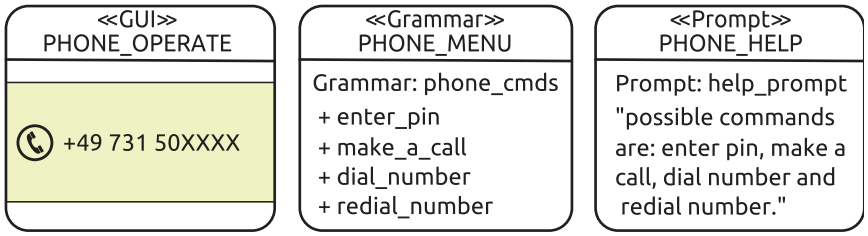
Eine Erweiterung dieses Profils (vgl. Abb. 1(a)) ist in zwei Dimensionen möglich: 1) Unterstützung weiterer UML-Diagramme durch Einbeziehung weiterer UML-Elemente des Metamodells, z.B. *Activities* und *Actions*, oder 2) Einführung weiterer Modalitäten, z.B. Gestik, durch die Definition neuer Stereotypen.

Statische Aspekte

Das UI-Profil aus Abb. 1 (a) definiert die Stereotypen «GUI», «Grammar», «Prompt» und deren statische Aspekte zur Modellierung von graphisch-haptischen und sprachgesteuerten Dialogen.



(a) UML-Metamodell: Ausschnitt des UI-Profils.



(b) Stereotypisierte Zustände mit multimodalen Aspekten: Grafische Elemente («GUI»), Grammatiken zur Spracherkennung («Grammar») und Systemausgaben via Prompt («Prompt»).

Abbildung 1: Die Profildefinition 1(a) und ein Anwendungsbeispiel der Stereotypen in 1(b).

Jeder Stereotyp enthält Attribute zur Referenzierung modalitätsspezifischer Inhalte und erweitert die Klasse State des UML-Metamodells.

Der Stereotyp «GUI» ermöglicht die Integration eines Grafischen User Interface (GUI), indem er die grafische Repräsentation eines UI-Elements (Widget) definiert. Dazu wird das Attribut **ui_description** verwendet. Sein Wert ist eine Referenz zu einer formalen Beschreibung einer UI-Spezifikation. Das Attribut **viewState_id** verweist auf ein bestimmtes Widget aus dieser Spezifikation (vgl. Abb. 1 (a) und (b)).

Zur formalen Beschreibung der UI-Spezifikation kommen eine Reihe existierender Formalismen in Frage, z.B. das abstrakte AIUML (Merrick, Wood und Krebs 2004) oder MXML (Tapper, Labriola, Boles und Talbot 2008).

Der Stereotyp «Grammar» ermöglicht die Modellierung von Spracheingaben. Er referenziert über das Attribut **grammar_id** eine Grammatik, z.B. in Form der Speech Recognition Grammar Specification (W3C 2004).

Eine Grammatik kann andere Grammatiken verwenden, so dass auch komplexe Grammatiken definiert werden können und Grammatiken sich einfach wiederverwenden lassen. Der Zustand **PHONE_MENU** in Abb. 1 (b) enthält die komplexe Grammatik **phone_help_commands**, die andere Grammatiken, u.a. **enter_pin**, **make_a_call**, …, enthält.

Prompts realisieren die Sprachausgabe des Systems. Wir definieren den Stereotyp «Prompt» mit dem Attribut **prompt_id**. Abbildung 1 (b) (rechts) zeigt einen Zustand, der einen Prompt darstellt und der dazu dient, dem Nutzer mögliche Sprachkommandos mitzuteilen.

Dynamische Aspekte

Ein wichtiger Schritt zur Definition einer ausführbaren Modellierungssprache ist die Definition ihrer Semantik. Bezüglich eines Profils ist es notwendig, verschiedene Aspekte zu klären: Temporales Verhalten jedes Stereotyps, die Beziehung der Verhalten verschiedener Stereotypen untereinander und die Beziehung zur Semantik der UML – in unserem Fall Zustandsautomaten. Eine formale Semantik für UML-Zustandsautomaten definieren (Kohlmeyer 2009).

Während eines Zustandsübergangs werden folgende Verhalten ausgeführt: *Exit*, *Effect*, *Entry* und *Do* referenzieren in der UML definierte *Behaviors* (OMG 2010).

Das Verhalten der Stereotypen wird durch neue Attribute vom Typ *Behavior* definiert.

LoadViewState dient der Anzeige des referenzierten Widgets, *LoadGrammar* führt zur Aktualisierung der aktiven Grammatik des Spracherkenners und *PlayPrompt* gibt den hinterlegten Prompt in Form einer akustischen Systemausgabe wieder. Die Verhalten der Stereotypen verwenden die Informationen der Attribute, die als Teil der statischen Beschreibung des Profils definiert wurden (vgl. Abschnitt Statische Aspekte). *LoadViewState* und *LoadGrammar* starten, nachdem ein Zustand betreten wurde, parallel zu *Entry*. Sobald diese drei Verhalten abgeschlossen sind, startet *PlayPrompt* parallel zu *Do*. Falls ein Zustand eine ausgehende *Transition* ohne *Trigger* und *Guard* besitzt, kann der Zustand verlassen werden, sobald beide Verhalten, *Do* und *PlayPrompt*, beendet sind.

3.4 Methode

Folgende Gründe sprechen dafür, Zustandsautomaten als Basis für unsere DSML zu verwenden: Die Charakteristik des Verhaltens von UIs wird sehr gut von Zustandsautomaten reflektiert. Die Interaktion zwischen Benutzer und System findet in Form eines Dialogs statt. Dieser besteht in der Regel aus mehreren Schritten zwischen Dialogzuständen. In Zustandsautomaten beschreibt eine Menge von aktiven Zuständen einen Gesamtzustand. Erst wenn Ereignisse auftreten, wird ein Zustandswechsel durchgeführt.

Die generelle Eignung von Zustandsautomaten als Grundkonzept zur Modellierung von Sprachdialogen wurde bereits gezeigt (Kölzer 2002; Goronzy, Mochales und Beringer 2006). Unser Ansatz (Dausend und Poguntke 2010) führt einige Erweiterungen ein. Die Modellierung abstrakter Dialoge unter Verwendung von Zustandsdiagrammen wird in (De Melo 2010) beschrieben. Die folgende Beschreibung unserer Methode umfasst ausgewählte Konzepte der Stereotypen des UI-Profils. Ein durchgängiges Beispiel einer konkreten Anwendung des UI-Profils ist in (Dausend und Poguntke 2010) beschrieben.

Grafische Benutzerschnittstelle

Zur Beschreibung grafisch-haptischer Interaktion mit Zustandsautomaten und dem Stereotyp «GUI» werden drei Grundkonzepte beschrieben: 1) Eindeutige Zuordnung einer grafischen Repräsentation zu einem Zustand, 2) Hierarchisierung von Zustandsautomaten, sowie 3) Paralle-

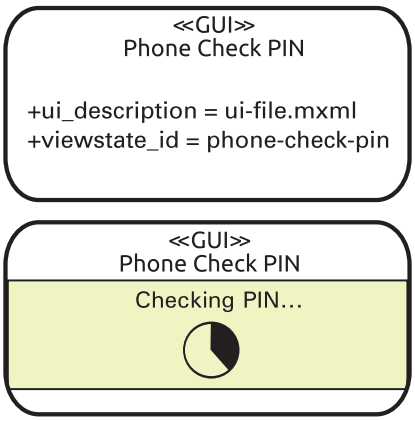


Abbildung 2: Alternative Darstellung von Zuständen mit dem Stereotyp «GUI».

lität aktiver Zustände. Unter Verwendung dieser Grundkonzepte ist es möglich, den Bildschirminhalt zur Ausführungszeit dynamisch zu komponieren.

Die Verwendung des Stereotyps «GUI» erlaubt es, Zuständen einen Bildschirminhalt als Widget (vgl. Abschnitt Statische Aspekte) zuzuordnen. Dazu wird der *Stereotyp* «GUI» zu einem Zustand hinzugefügt und den Attributen des Stereotyps werden Werte zugewiesen. Dabei referenziert der Wert des Attributs **ui_description** eine formale Beschreibung eines UIs, das eine Menge von Widgets definiert und **viewstate_id** ein darin enthaltenes Widget.

Zur Erleichterung der Lesbarkeit der Dialoge werden statt textueller Notation die grafischen Repräsentationen der formal beschriebenen Widgets eines Stereotyps im Zustand angezeigt (vgl. Abb. 2, unten).

Ein grundlegendes Konzept zur Strukturierung in UML-Zustandsautomaten ist die Hierarchisierung, also die Verfeinerung eines Zustands durch weitere Unterzustände. Hinsichtlich der Modellierung von Dialogen interaktiver Anwendungen ermöglicht dies beispielsweise die Strukturierung von Aufgaben in Teilaufgaben oder die Kapselung von Dialogen zur Wiederverwendung, z.B. Bestätigungsdialoge.

Zur Unterstützung des Stereotyps «GUI» von grafischen Repräsentationen unterschiedlicher Hierarchisierung wurde ein Kompositionskonzept passend zu Zustandsautomaten definiert. Eine einfache, aber effektive Möglichkeit ist die Komposition von Repräsentationen über Ebenen. Der Bildschirminhalt einer Anwendung wird dabei durch Übereinanderlegen verschiedener Widgets erzeugt, z.B. ein sich öffnendes Menü über dem aktuellen Fenster einer Applikation.

Überträgt man das Kompositionskonzept grafischer Repräsentationen auf Zustandsautomaten, indem die Tiefe eines Zustands die Ebene der Repräsentation eines Widgets bestimmt und der Wurzelzustand die unterste Ebene einer grafischen Repräsentation definiert, folgt daraus, dass grafische Repräsentationen hierarchisch tiefer liegender Zustände die Repräsentationen hierarchisch höher liegender Zustände ganz oder teilweise überlagern.

Abbildung 3 verdeutlicht, wie sich eine gesamte Repräsentation unter Verwendung der Kombination der aktiven Zustände (links – graue Zustände) in verschiedenen Ebenen der Zustands hierarchie realisieren lässt. Der Wurzel zustand definiert den Hintergrund der Anwendung. Alle weiteren aktiven Zustände sind ebenfalls durch den Stereotyp «GUI» erweitert (mitte) und sind Unterzustände des Wurzelzustands. Durch Überlagerung der Widgets der aktiven Zustände aufsteigend vom Wurzelzustand wird die gesamte Repräsentation beschrieben (rechts).

Neben der Möglichkeit der Hierarchisierung von Zustandsdiagrammen kann Parallelität verwendet werden, um Nebenläufigkeit zu modellieren. Grafische Benutzerschnittstellen bestehen oft aus verschiedenen Komponenten, die parallel verwendet werden. Abbildung 4 zeigt die Modularisierung einer grafischen Komponente am Beispiel eines File Browsers unter Verwendung von Parallelität. Dabei wird das beschriebene GUI in sinnvolle Teileinheiten zerlegt, z.B. einen Verzeichnisbaum und eine Dateiansicht. Beide Einheiten können so unabhängig voneinander beschrieben werden. Gemeinsames Verhalten wird auf der Ebene des gemeinsamen Vaterzustands definiert.

Parallelität spielt vor allem bei der Modellierung multimodaler Konzepte eine wichtige Rolle, da sie eine getrennte Beschreibung des UIs für die einzelnen Modalitäten sowie Multimodalität ermöglicht (vgl. Abschnitt Multimodalität sowie (Dausend und Poguntke 2010)).

Sprachbedienung

Die Sprachbedienung ist die zweite wichtige Modalität, die wir bei der Modellierung interaktiver Anwendungen berücksichtigen. Sie setzt sich aus Spracherkennung und Sprachsynthese zusammen.

Dialogsysteme verwenden zumeist sprecherunabhängige Spracherkennung, die den Wortschatz des Spracherkenners entsprechend des aktuellen Anwendungskontexts einschränkt, um eine optimale Erkennungsrate zu gewährleisten. Das heißt, dass die Spracherkennung in Abhängigkeit vom Systemzustand nur bestimmte Wörter und Sätze verstehen kann.

Der Stereotyp «Grammar» wird einem

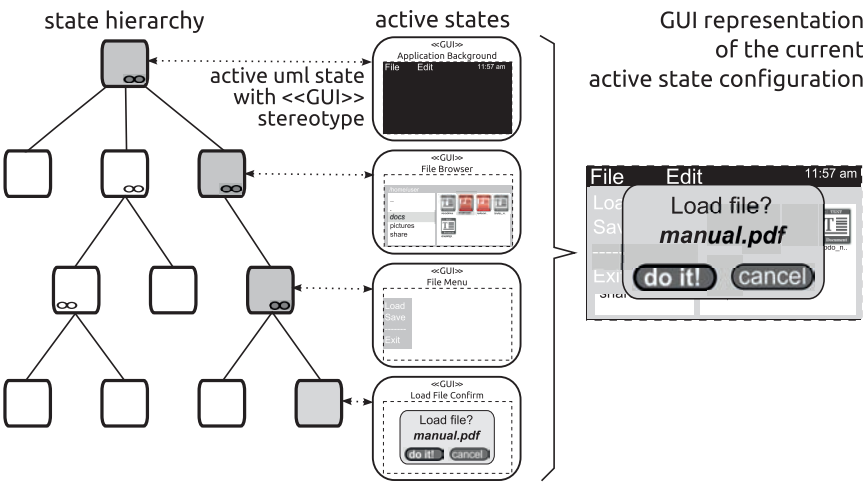


Abbildung 3: Zustandshierarchie einer Beispielanwendung (links); Darstellung der Zustände inkl. ihres <<GUI>> Stereotypen (Mitte); Resultierende GUI Repräsentation entsprechend der aktuell aktiven Zustände (rechts).

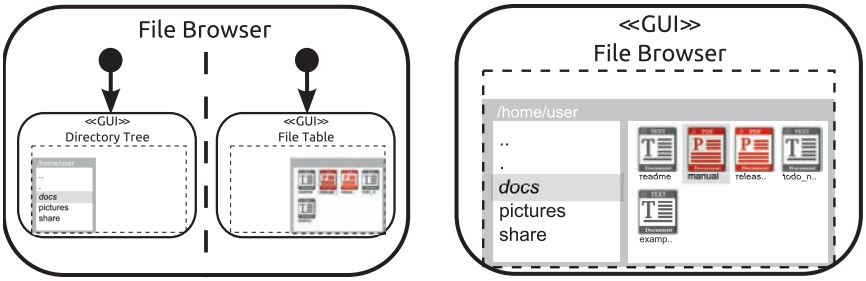


Abbildung 4: Modularisierung grafischer Benutzerschnittstellen unter Verwendung von Parallelität. Realisierung einer GUI-Schnittstelle eines Dateibrowsers (links) durch Verwendung eines orthogonalen Zustands mit zwei Unterzuständen. Im Zustand rechts wird die GUI-Schnittstelle als eine Gesamtkomponente aufgefasst.

Zustand hinzugefügt und ermöglicht die Referenzierung einer Grammatik, die als Eingabe für einen Spracherkenner dient. Die Konzepte zur Modellierung lehnen sich eng an die Konzepte zur Beschreibung der grafischen Repräsentation an.

Die aktuelle Grammatik einer interaktiven Anwendung wird analog der GUI durch die Komposition der Grammatiken aller aktiven Zustände gebildet, d.h. alle aktiven Zustände mit einem Stereotyp «Grammar».

Der Stereotyp «Prompt» dient der Wiedergabe von Systemausgaben in Form von Sprachsynthese. Dazu wird das *Do*-Verhalten eines Zustands erweitert.

Multimodalität

Multimodalität, die grafisch-haptische und Sprachbedienung einschließt, kann unter Verwendung der eingeführten Stereotypen modelliert werden (Dausend und Poguntke 2010). Im Folgenden gehen wir auf Eigenschaften von Stereotypen ein, die bezüglich der Modellierung multimodaler Anwendungen von Interesse sind.

Die Semantik der definierten Stereotypen wurde widerspruchsfrei konstruiert, so dass die Stereotypen in beliebiger Kombination verwendet werden können.

Ein Zustand kann beispielsweise durch «GUI» als auch «Grammar» erweitert werden. Diese Kombination der Stereotypen realisiert ein Sprachdialogkonzept, wie es z.B. von Mercedes Benz eingesetzt wird: Benutzer werden während der Spracheingabe durch die Anzeige einer Auswahl aktuell möglicher Sprachkommandos über den sogenannten Teleprompter unterstützt.

Ein weiteres Konzept, das durch die Kombination von Stereotypen realisiert werden kann, ist das sogenannte Barge-In. Es ermöglicht dem Benutzer, die Sprachausgabe des Sprachdialogsystems jederzeit zu unterbrechen. Dazu werden «Grammar» und «Prompt» in einem Zustand kombiniert.

4. Verwandte Arbeiten

De Melo (2010) gibt einen Überblick über Ansätze zur Modellierung von Benutzerschnittstellen und liefert eine detaillierte Analyse und Bewertung. Hinsichtlich der Eigenschaften Angemessenheit, Ausdrucksmächtigkeit, Erweiterbarkeit um Multimodalität, Universalität, Verbreitung, Verständlichkeit und Werkzeugunterstützung bietet UML insgesamt die meisten Vorzüge. Neben UML (OMG 2010) werden beachtete Ansätze wie CTT (Paterno, Mancini und Meniconi 1997), UsiXML (Vanderdonckt 2005) und andere in den Vergleich einbezogen.

Die Stärke von CTT liegt in der hierarchischen Modellierung der Aufgabendomäne. Dialoge werden abstrakt unter Angabe der Modalitäten definiert. Eine konkrete UI-Beschreibung der einzelnen Modalitäten ist allerdings nicht vorgesehen.

Die Grundlage von UsiXML bilden abstrakte Modelle, die mit Hilfe der Definition von Kontext und Transformationen sowie deren Anwendung zu konkreten Modellen führen. Benutzerschnittstellen werden in drei Transformationsschritten ausgehend von abstrakten Interaktionselementen realisiert. Die Anwendung von UsiXML in einem modellbasierten Entwicklungsprozess wird durch eine heterogene breite Werkzeuglandschaft unterstützt. Leider können UML-Modelle nicht direkt integriert werden. Eine Standardisierung von UsiXML ist in Arbeit.

Bisherige Ansätze zur Modellierung von interaktiven Systemen mit UML behandeln schwerpunktartig statische Aspekte, wie bei der Strukturierung der Präsentationselemente beispielsweise durch die Verwendung von Stereotypen für Klassendiagramme (WISDOM) (Nunes und Cunha 2000), oder konzentrieren sich auf Aspekte des konzeptuellen Designs, der Navigation oder der Präsentation (UWE) (Hennicker und Koch 2001). Diese Ansätze verwenden ebenfalls Profile zur Aufgabenmodellierung und zur Beschreibung der grafischen Elemente, berücksichtigen jedoch keine Multimodalität.

Abschnitt 3.1 motiviert die Wahl der Zustandsautomaten zur Modellierung multimodaler interaktiver Systeme anhand der Arbeiten (Kölzer 2002) und (Goronzy, Mochales und Beringer 2006). Beide Arbeiten unterscheiden sich aber, wie im Folgenden näher beschrieben, von unserem Ansatz.

Dialog Statecharts (Kölzer 2002) adaptieren Harel Statecharts (Harel 1987), um Dialoge mit wechselnder Initiative zu modellieren. Die Notation und die Semantik der Statecharts wurden umfangreich verändert und an die Bedürfnisse der Sprachdialogmodellierung angepasst, so dass das Verhalten letztlich vom verwendeten Dialogsystem abhängt.

Gornonzy und andere (Goronzy und Beringer 2005; Goronzy, Mochales und Beringer 2006) schlagen ein Konzept zur Modellierung multimodaler Interaktion mit UML-Zustandsautomaten vor. Die Umsetzung der Konzepte ist jedoch nicht durchgängig UML-konform. In der Folge gehen einige Vorteile durch die Verwendung von UML verloren, z.B. müssen UML-Experten die Semantik des Werkzeugs lernen und der Austausch der Modelle wird erschwert. Die Integration der eingeführten Konzepte in UML wurde nicht formal, z.B. in Form einer UML-Erweiterung, definiert, sondern ist durch die Implementierung gegeben. Außerdem ist die Modellierung multimodaler Dialoge in der Art beschränkt, dass sprachgesteuerte und grafisch-haptische

Dialoge strikt voneinander getrennt in parallelen Zustandsdiagrammen strukturiert werden müssen. Weiterhin ist es nicht möglich, die verwendeten Formalismen für die einzelnen Modalitäten auf das Zielsystem abzustimmen, da die Widgets auf Java-Implementierungen basieren.

Grammatiken können ähnlich unserem Ansatz durch Vererbung von Zuständen zu Unterzuständen dynamisch konstruiert werden, für die Definition von grafischen UIs ist eine Komposition aufgrund des Zustandsmodells jedoch nicht vorgesehen.

Die Aufgabenmodellierung geht als wichtiger Teil der Modellierung der multimodalen interaktiven Anwendungen voraus und kann laut (De Melo 2010) ebenfalls mit UML modelliert werden. Eine hierarchische Aufgabenanalyse wie bei CTT ist weiterhin möglich. Zudem sind Aufgaben- und Dialogmodell durch die Kopplung der Diagramme (*Use Case*, *Activity* und *State Machines*) in UML stärker als bisher miteinander verbunden.

Einige Ansätze verwenden Profile, deren Stereotypen konkrete Widgets definieren, um eine Beschreibung der grafischen Repräsentation auf Ebene der UML vorzunehmen, u.a. (Hennicker und Koch 2001; Martins und Silva 2007). Daraus resultieren allerdings einige Nachteile. So können nur Elemente verwendet werden, die vom Profil vorgesehen sind, z.B. Combo-Boxen oder Textfelder. Da die vorgestellten Konzepte unabhängig vom verwendeten Formalismus sind, ist unser Ansatz nicht auf einen konkreten Formalismus beschränkt. Somit können bei geeigneter Wahl des Formalismus beliebige Widgets verschiedenster Abstraktionsstufen mit unterschiedlichen Schnittstellen verwendet werden, z.B. ein Widget mit einer Uhr, das bereits das Verhalten der Uhr kapselt. Eine Beschreibung der Widgets durch Stereotypen, wie in anderen Ansätzen, ließe sich ebenfalls mit unserem Konzept realisieren.

Zur Unterstützung der Anwendung unseres Ansatzes und zur Evaluierung unserer Methode wurde ein Werkzeug entwickelt (Zilles 2009). Im Vergleich zu den genannten Arbeiten (s.o.) wurde besonderer Wert auf UML-Konformität hinsichtlich Notation und Ausführungssemantik der Modelle gelegt. Zudem wurden alle definierten statischen und dynamischen Konzepte unseres UI-Profils integriert. Die Entwicklung komplexer multimodaler Systeme wird durch eine interaktive Simulation der Zustandsdiagramme unterstützt. Ergänzend dazu wird auf Basis der Konfiguration aktiver Zustände und ihrer Stereotypen stets das resultierende UI angezeigt. Mögliche Interaktionen, z.B. Sprachkommandos, werden zur Interaktion mit der simulierten Anwendung angeboten. Die Eigenschaften des Modells, z.B. Werte von Variablen, können ebenfalls während der Simulation eingesehen und verändert werden.

5. Zusammenfassung und Ausblick

Der vorgeschlagene Ansatz zur Modellierung multimodaler interaktiver Anwendungen stellt Tauglichkeit und Anwendbarkeit in sein Zentrum. Die Hauptschwierigkeit ist, geeignete Beschreibungskonzepte zu definieren, die alle an der Entwicklung beteiligten Personen und Rollen adressieren und gleichzeitig die Beschreibung komplexer, multimodaler Anwendungen ermöglichen.

Eine Expertenevaluierung unseres Ansatzes (Dausend und Poguntke 2010) hat gezeigt, dass der Ansatz sowohl hinsichtlich Tauglichkeit als auch Benutzerfreundlichkeit als geeignet angesehen wird, um interaktive multimodale Anwendungen zu modellieren. Im Rahmen einiger Fallstudien mit unserem Ansatz wurden komplexe Modelle von existierenden interaktiven Anwendungen, z.B. die Bedienung des Telefons im Fahrzeug, erstellt und unter Verwendung unseres Werkzeugs validiert.

Im Gegensatz zu Ansätzen vergleichbarer Zielsetzung definieren wir eine DSML unter Verwendung eines UML-Profils für multimodale interaktive Anwendungen. Statische und dynamische Aspekte des Formalismus werden als UML-Stereotypen des Profils definiert. Die Methode beschreibt, wie verschiedene Konzepte unimodaler und multimodaler Interaktion durch das Profil und die Verknüpfung mit UML-Zustandsautomaten realisiert werden. Dabei bleibt die Semantik der UML stets erhalten, so dass der Ansatz werkzeugunabhängig ist und existierende Verfahren wie Codegenerierung für Zustandsautomaten, die auf der UML-Semantik basieren, verwendet werden können. Zur Unterstützung der Entwicklung wurde basierend auf einer formalen Beschreibung beider Formalismen ein Werkzeug implementiert, das die Validierung und Simulation der Zustandsautomaten und des UIs unterstützt.

Im Rahmen einiger Fallstudien wurde das Werkzeug bereits erfolgreich eingesetzt. Es hat sich gezeigt, dass die multimodalen Modelle interaktiver Anwendungen durch die Simulation deutlich einfacher erstellt werden können. Gerade Synchronisationsprobleme zwischen Modalitäten ließen sich so besser identifizieren. Die Simulation des UIs, insbesondere die Anzeige der aktuellen Gesamtrepräsentation des Systems, liefert einen guten Eindruck des Zusammenspiels der Modalitäten.

Zukünftige Arbeiten können der Erweiterung der Methode und ihrer Werkzeugunterstützung dienen, z.B. durch Einbeziehung weiterer Diagrammarten der UML oder Erweiterung des Profils um weitere Modalitäten. Im Rahmen kleinerer Werkzeugerweiterungen wurden bereits weitere Anwendungen erprobt: Sowohl Codegenerierung der modellierten grafischen UIs und ihres Verhaltens, als auch eine GOMS-Analyse der spezifizierten UIs innerhalb der Simulation basierend auf einem neuen Stereotypen, wurden prototypisch erfolgreich angewendet und können weiter verfolgt werden.

Außer Aspekten der Modellierung von UIs können UML-Konzepte zur Erstellung von DSML weiter untersucht werden. Beispielsweise existiert bislang keine Methodik zur Verknüpfung der Semantik eines UML-Profils mit der UML-Semantik die zu einer Integrierten Semantik führt. Außerdem wäre es wünschenswert, Verfahren zu haben, die eine Konsistenzprüfung zwischen der Semantik von UML und Profilen ermöglicht.

Literatur

Dausend, M. ; Poguntke, M.: Spezifikation multimodaler interaktiver Anwendungen mit UML. In: Mensch & Computer : Oldenbourg Verlag, 2010

De Melo, G.: Modellbasierte Entwicklung von Interaktionsanwendungen, Universität Ulm, 2010

Fuentes-Fernández, L. ; Vallecillo-Moreno, A.: An Introduction to UML Profiles. In: The European Journal for the Informatics Professional vol. 5, (2004), 6–13

Goronzy, S. ; Beringer, N.: Integrated Development and on-the-Fly Simulation of Multimodal Dialogs. In: Ninth European Conference on Speech Communication and Technology, 2005

Goronzy, S. ; Mochales, R. ; Beringer, N.: Developing Speech Dialogs for Multimodal HMIs Using Finite State Machines. In: Ninth International Conference on Spoken Language Processing. Pittsburgh, Pennsylvania : ISCA, 2006

Harel, D.: Statecharts: A Visual Formalism for Complex Systems. In: Science of Computer Programming vol. 8 (1987), Nr. 3, 231–274

Hennicker, R. ; Koch, N.: Modeling the User Interface of Web Applications with UML. In: Practical UML-Based Rigorous Development Methods-Countering or Integrating the eXtremists, Workshop of the pUML-Group at the UML. vol. 200, 2001

Kohlmeyer, J.: Eine formale Semantik für die Verknüpfung von Verhaltensbeschreibungen in der UML 2, Universität Ulm, 2009

Kölzer, A.: Diamod: Ein Werkzeugsystem zur Modellierung natürlichsprachlicher Dialoge, Mensch & Buch Verlag, 2002

Martins, C. ; Silva, A.: Modeling User Interfaces with the XIS UML Profile. In: Proc. 9th Int. Conf. Enterprise Information Systems. Funchal, Portugal : Springer, 2007.

Meixner, G.: Entwicklung einer modellbasierten Architektur für multimodale Benutzungsschnittstellen, Technische Universität Kaiserslautern, Kaiserslautern, 2010.

Merrick, R. A. ; Wood, B. ; Krebs, W.: Abstract User Interface Markup Language. In: Developing User Interfaces with XML: Advances on User Interface Description Languages (2004), 39–46

Nunes, N. J. ; Cunha, J. F.: Towards a UML Profile for Interaction Design: The Wisdom Approach. In: Lecture Notes in Computer Science vol. 1939, Springer (2000), 101–116

OMG: Unified Modeling Language Superstructure v2.3. In: Superstructure Specification vol. 2 (2010)

Paternò, F. ; Mancini, C. ; Meniconi, S.: ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction, 1997.

Selic, B.: The Theory and Practice of Modelling Language Design for Model-Based Software Engineering – A Personal Perspective. In: Lecture Notes in Computer Science vol. 6491 (2011), 290–321

Tapper, J. ; Labriola, M. ; Boles, M. ; Talbot, J.: Adobe Flex 3 : Pearson Education, 2008.

Vanderdonckt, J.: A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In: Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE. vol. 5, 2005, 16–31

W3C: Speech Recognition Grammar Specification 1.0, 2004.

Zilles, J.: Entwicklung eines Interpreters für Zustandsautomaten multimodaler Systeme, Universität Ulm, 2009

1

2

1 Marcel Dausend (marcel.dausend@uni-ulm.de) studierte Medieninformatik an der Universität Ulm. Nach Beendigung des Studiums arbeitete er ab Juli 2007 als Doktorand in der Daimler Forschung und Vorentwicklung, wo er im Bereich der Spezifikation von Sprachdialogsystemen und grafisch-haptischen Mensch-Maschine Schnittstellen tätig war. Seit April 2010 ist er wissenschaftlicher Mitarbeiter am Institut für Programmiermethodik und Compilerbau an der Universität Ulm und arbeitet am Thema modellbasierter Spezifikation.

2 Mark Poguntke (mark.poguntke@daimler.com) studierte Medieninformatik an den Universitäten Ulm und Monash, Melbourne. Nach Beendigung des Studiums arbeitet er seit Januar 2009 als Doktorand in der Daimler Forschung und Vorentwicklung, wo er im Bereich der Modellierung und Spezifikation von Mensch-Maschine Schnittstellen im Fahrzeug tätig ist. Sein Schwerpunkt liegt dabei auf abstrakten, flexiblen Modellierungen als Vorbereitung auf spätere Erweiterbarkeit nach Abschluss der Entwicklung.