Ausführbare UML-Modelle multimodaler Interaktionsanwendungen

Marcel Dausend, Mark Poguntke

marcel.dausend@uni-ulm.de, mark.poguntke@daimler.com

Abstract: Komplexe interaktive Anwendungen stellen sowohl für die Nutzer als auch für die Entwickler eine große Herausforderung dar. Ein Beispiel für eine solche Anwendung ist das Infotainmentsystem in modernen Fahrzeugen. Der Nutzer wird bei der Bedienung durch Multimodalität (Sprachbedienung neben der grafisch-haptischen Bedienung) unterstützt, um die Ablenkung während des Fahrens zu reduzieren. Da die Kombination der verschiedenen Interaktionsmöglichkeiten die Komplexität der Anwendung deutlich erhöht, ist es notwendig, 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, so dass sie die Integration von Benutzerschnittstellen verschiedener Modalitäten zu ausführbaren Modellen einer multimodalen Anwendung ermöglichen. Die Funktionalität einer Anwendung in Kombination mit den unterschiedlichen Ein- und Ausgabemodalitäten kann in unserem erstellten Werkzeug umgesetzt und durch Simulation validiert werden.

1 Einleitung

Interaktive Anwendungen, wie z.B. die Bedienung eines Mobiltelefons, sind Teil unseres Alltags geworden und werden zunehmend komplexer. Multimodalität, die gemeinsame 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. Die Vielfalt der Kombinationsmöglichkeiten der Interaktionen erhöht jedoch den Entwicklungsaufwand der Anwendung.

Wir stellen eine Modellierungsmethode für multimodale interaktive Anwendungen vor, deren Tauglichkeit und Angemessenheit im Rahmen einer Expertenevaluierung [DP10] gezeigt wurde. In diesem Beitrag erläutern wir detailliert den Formalismus, die Methode und ihre Werkzeugunterstützung. Insbesondere gehen wir auf die Ausführbarkeit der Modelle und die Simulation des User Interfaces (UIs) ein. Die Verwendung des Formalismus UML und die Simulierbarkeit der Modelle stellen den zentralen Aspekt unseres Konzepts zur Unterstützung der Entwicklung multimodaler interaktiver Anwendungen dar.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

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 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 Unified Modeling Language (UML) [OMG10] 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 5). Diese Ansätze erweitern UML, da UML selbst nicht ausreichend geeignet ist, um Konzepte der Mensch-Maschine-Interaktion zu modellieren (vgl. [Mei10]).

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-STATE MACHINES, sowie PROFILE und STEREOTYPEN, eine detaillierte Vorstellung des vorgeschlagenen Formalismus sowie eine Erklärung der einzelnen Konzepte der Methodik.

3.1 UML-Zustandsdiagramme

Im Folgenden werden ausgewählte Konzepte von UML-Zustandsautomaten genannt und erläutert, inwiefern diese zur Modellierung von interaktiven Anwendungen verwendet werden können.

Transitionen (Zustandsübergänge) werden mit Hilfe von JUNCTIONS und DECISIONS verzweigt oder zusammengeführt. Dies ermöglicht Alternativen innerhalb eines Dialogablaufs zu modellieren oder alternative Dialogpfade zu vereinigen. Ein Pfad von Transition zwischen zwei Zuständen bildet eine sogenannte COMPOUND TRANSITION. 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 Zustandsdiagramms verwendet werden, z.B. um Dialogbausteine zu

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

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.

Folgende Gründe sprechen dafür Zustandsdiagramme 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 [Köl02] [GMB06]. Unser Ansatz [DP10] führt einige Erweiterungen ein.

3.2 Profile und Stereotypen der UML

Die UML kann auf zwei Arten erweitert werden: Über *Metamodellerweiterung* sowie *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."[OMG10, p. 679]. Profile umfassen eine Menge von Stereotypen, die einzelne Aspekte einer Erweiterung beschreiben. Im Gegensatz 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 nicht 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 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 (*Textual annotations*) 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.

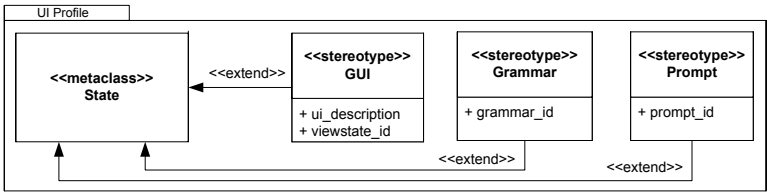
Selic beschreibt detailliert, unter welchen Umständen die Entwicklung einer DSML sinnvoll ist und gibt einen Leitfaden zur Erstellung von DSMLs unter Verwendung von UML-

ISBN 978-3-88579-286-4

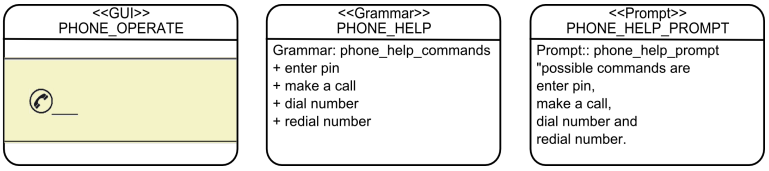
PROFILEN [Sel11]. Eine Anleitung zur Profilerstellung liefern auch [FFVM04].

3.3 Formalismus

Als Grundlage zur Modellierung von multimodalen interaktiven Systemen verwenden wir UML-STATE MACHINES (Zustandsautomaten) [OMG10]. 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. Dieses PROFIL berücksichtigt sowohl statische als auch dynamische Aspekte der Benutzerschnittstelle.



(a) UML-Metamodel: Ausschnitt des UI-Profils UI_PROFILE



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

Abbildung 1: Die Profildefinition Abb. 1(a) und ein Anwendungsbeispiel Abb. 1(b).

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.

Dynamische (bzw. Verhaltens-) Aspekte betreffen den Dialog zwischen Benutzer und System. Dynamische Aspekte sind definiert als Änderungen statischer Aspekte als Folge der Auswertung von Aktionen des Benutzers (oder des Systems) im zeitlichen und situativen Kontext.

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.

ISBN 978-3-88579-286-4

3.3.1 Statische Aspekte

Das UI-PROFIL aus Abb. 1 definiert die Stereotypen ≪GUI≫, ≪Grammar≫, ≪Prompt≫ und deren statische Aspekte zur Modellierung von graphisch-haptischen und sprachgesteuerten Dialogen. 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)).

Zur formalen Beschreibung der UI-Spezifikation kommen eine Reihe existierender Formalismen in Frage, z.B. das abstrakte AIUML [MWK04], MXML [TLBT08].

Der STEREOTYP ≪Grammar≫ ermöglicht die Modellierung von Spracherkennung. Er referenziert über das Attribut *grammar_id* eine Grammatik, z.B. in Form der Speech Recognition Grammar Specification [W3C04].

Eine Grammatik kann andere Grammatiken verwenden, so dass auch komplexe Grammatiken definiert werden können und Grammatiken sich einfach wiederverwenden lassen. Der Zustand PHONE_HELP 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 und werden analog zu Grammatiken eingeführt. Wir definieren den STEREOTYP ≪Prompt≫ mit dem Attribut *prompt_id*. Abbildung 1(b) zeigt (rechts) einen Zustand der einen Prompt darstellt und der dazu dient, dem Nutzer mögliche Sprachkommandos mitzuteilen.

3.3.2 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 [Dau07] und [Koh09].

Abbildung 2 zeigt Verhalten, die während eines Zustandsübergangs ausgeführt werden. EXIT, EFFECT, ENTRY und DO referenzieren in der UML definierte BEHAVIORS [OMG10].

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 (nicht in Abb. 1(a) enthalten). Diese verwenden die Informationen der Attribute, die als Teil der statischen Beschreibung des PROFILS definiert wurden (vgl. Abschnitt 3.3.1).

LoadViewState und LoadGrammar starten, nachdem ein Zustand betreten wurde, paral-

lel 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.

Eine genaue Beschreibung der Semantik der STEREOTYPEN wird im nächsten Kapitel im Kontext ihrer Anwendungskonzepte und der Methodik erläutert.

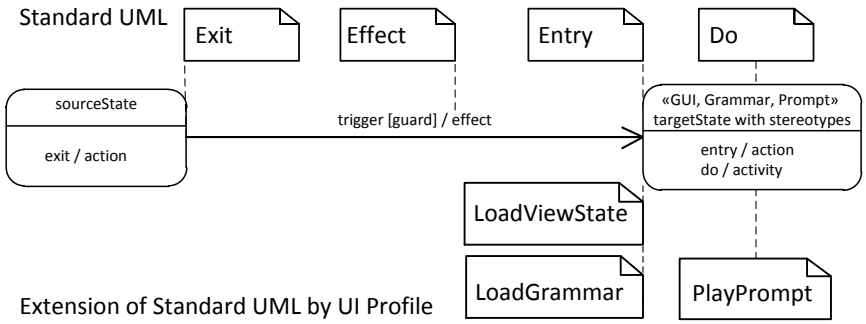


Abbildung 2: Ausführung einer Transition zwischen zwei Zuständen in zeitlicher Abfolge (von links nach rechts) gemäß der UML-Semantik (obere Hälfte) und die Ausführungssequenz des UI-Profils (untere Hälfte).

3.4 Methode

Die Beschreibung der Methode beschränkt sich auf einzelne Konzepte der Anwendung der STEREOTYPEN des UI-PROFILS. Die Modellierung abstrakter Dialoge unter Verwendung von Zustandsdiagrammen wird in [Mel10] beschrieben. Ein durchgängiges Beispiel einer konkreten Anwendung des UI-PROFILS ist in [DP10] beschrieben.

3.4.1 Grafische Benutzerschnittstelle

Zur Beschreibung grafisch-haptischer Interaktion mit Zustandsdiagrammen und dem STEREOTYP ≪GUI≫ werden drei Grundkonzepte beschrieben: 1) Die Zuordnung einer grafischen Repräsentation zu einem Zustand, 2) die Hierarchisierung von Zustandsautomaten, sowie 3) Parallelität 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 3.3.1) zuzuordnen. Die Verwendung erfolgt über das Hinzufügen des STEREOTYPS ≪GUI≫ zu einem Zustand und die Zuweisung von Werten zu den Attributen des Stereotyps. Dabei referenziert das Attribut *ui_description* eine formale Beschreibung eines UIs, das eine Menge von Widgets definiert. Das Attribut *viewstate_id* referenziert eindeutig 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

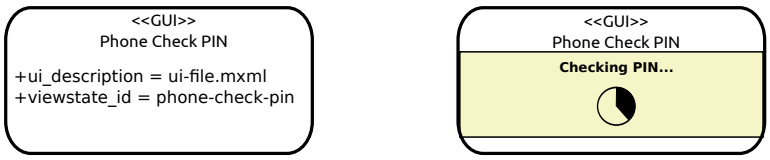


Abbildung 3: Unterschiedliche Darstellung von Zuständen mit dem STEREOTYP ≪GUI≫. Links: Anzeige der Attribute und ihrer Werte in textueller Form innerhalb des Zustands; Rechts: Darstellung des referenzierten Widgets im Zustand.

angezeigt (vgl. Abb. 3, rechts).

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 STEREOTYP ≪GUI≫ von grafischen Repräsentationen unterschiedlicher Hierarchisierung, wurde ein Kompositionskonzept passend zu Zustandsautomaten definiert. Eine einfache, aber effektive Möglichkeit stellt die Komposition von Repräsentationen über Ebenen dar. Ein 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 4 verdeutlicht, wie sich eine gesamte Repräsentation unter Verwendung der Kombination der aktiven Zustände in verschiedenen Ebenen der Zustandshierarchie realisieren lässt. Der Wurzelzustand definiert den Hintergrund der Anwendung. Alle weiteren grauen Zustände (rechts) sind ebenfalls durch den STEREOTYP ≪GUI≫ erweitert und sind aktive Unterzustände des Wurzelzustands. Durch Überlagerung der aktiven Zustände aufsteigend vom Wurzelzustand wird die gesamte Repräsentation beschrieben. Zur Überlagerung von Repräsentationen können auch Ebenen mit Transparenz verwendet werden, z.B. zum Ausgrauen inaktiver Bedienelemente.

Neben der Möglichkeit der Hierarchisierung von Zustandsdiagrammen kann Parallelität verwendet werden, um Nebenläufigkeit zu modellieren. Grafische Benutzerschnittstellen bestehen oft aus verschiedenen Komponenten, die häufig auch parallel verwendet werden. Abbildung 5 zeigt die Modularisierung einer grafischen Komponente am Beispiel eines File Browsers, unter Verwendung von Parallelität. Dabei wird die 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.

ISBN 978-3-88579-286-4

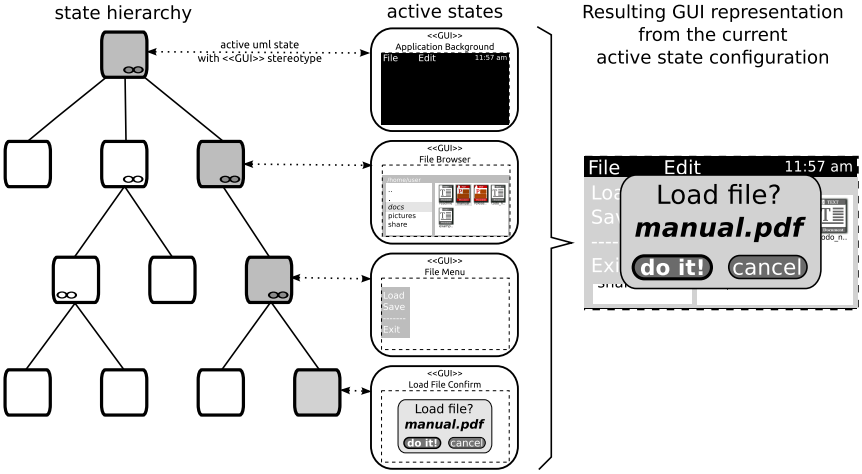


Abbildung 4: Zustandshierarchie einer Beispielanwendung (links); Darstellung der Zustände inkl. ihres ≪GUI≫ STEREOTYPEN (Mitte); Resultierende GUI Repräsentation entsprechend der aktuell aktiven Zustände (rechts).

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 3.4.3 sowie [DP10]).

3.4.2 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 Anwendungskontext 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 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≫.

Eine Erweiterung dieses Konzepts ermöglicht den Ausschluss von Grammatiken. Grammatiken können hierarchisch höher definierte Grammatiken ausschließen. Dadurch können Zustandsübergänge übergeordneter Zustände in Abhängigkeit des aktuellen Dialogkontexts verhindert werden.

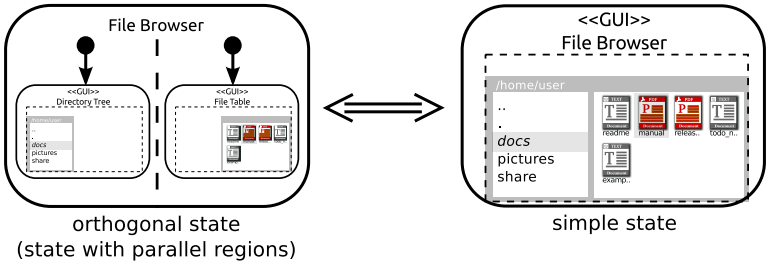


Abbildung 5: Modularisierung grafischer Benutzerschnittstellen unter Verwendung von Parallelität. Realisierung einer GUI-Schnittstelle eines Dateibrowsers (links) durch Verwendung eines orthogonalen Zustand mit zwei Unterzuständen. Im Zustand rechts wird die GUI-Schnittstelle als eine Gesamtkomponente aufgefasst.

Der STEREOTYP ≪Prompt≫ dient der Wiedergabe von Systemausgaben in Form von Sprachsynthese. Dazu wird das DO Verhalten eines Zustands erweitert. Prompts werden normalerweise in Textform definiert und zur Ausführung über eine Graphem-zu-Phonem Umwandlung an eine Sprachsynthese weitergeleitet.

3.4.3 Multimodalität

Multimodalität, die grafisch-haptische als auch Sprachbedienung einschließt, kann unter Verwendung der eingeführten Stereotypen modelliert werden [DP10]. Im Folgenden gehen wir auf Eigenschaften von Stereotypen ein, die bezüglich der Modellierung multimodaler Anwendungen von Interesse sind.

Stereotypen können beliebig miteinander kombiniert werden solange ihre Semantik keine Widersprüche zur UML-Spezifikation aufweist. 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.

Nachdem der Zustand betreten und die Grammatik geladen wurde, wird der referenzierte Prompt wiedergegeben (vgl. Abb. 2). Dabei kann ein auftretendes Erkennungsergebnis aufgrund der geladenen Grammatik zum Verlassen des Zustands führen, bevor die Sprachausgabe beendet wurde.

4 Werkzeugunterstützung

Die Modellierung multimodaler Systeme stellt trotz Verwendung der UML eine komplexe Aufgabe dar. Insbesondere, wenn die Dialoge optimal auf den situativen Kontext abgestimmt werden sollen und es sich um große interaktive Anwendungen, wie z.B. Fahrzeugdialoge, handelt.

Parallel zur Konzepterstellung haben wir ein Werkzeug entwickelt [Zil09], das die Modellierung multimodaler interaktiver Systeme durch ausführbare Modelle ermöglicht. Das Werkzeug unterstützt UML-konforme Zustandsautomaten sowie alle definierten statischen und dynamischen Konzepte des UI-Profils. Es bietet eine ansprechende Visualisierung der Ausführung der Modelle sowie die Simulation des UIs inklusive der Möglichkeit die Interaktion zu beeinflussen.

Die technische Grundlage des Werkzeugs bilden zwei Komponenten: Ein selbst entwickelter Editor für UML-Zustandsdiagramme auf Basis der Eclipse IDE[1] und FlexBuilder 3 [TLBT08] zur formalen Beschreibung grafischer Elemente.

Zur Modellierung von Multimodalität bietet das Werkzeug weitere Komponenten: Einen Editor für Grammatiken, einen für Prompts und einen für Stereotypen. Die Widgets des grafischen UIs werden über das FlexBuilder3-Plugin für Eclipse erstellt und über sogenannte *ViewStates* identifiziert.

Alle Daten eines Modells und der Widgets werden in einem gemeinsamen Projekt verwaltet. Die Spezifikation des UIs wird in MXML-Dateien gespeichert. Diese Spezifikationen können im Editor mit einem Zustand durch Verwendung des Stereotyps ≪GUI≫ assoziiert werden.

Das Werkzeug (vgl. Abb. 6) bietet neben den verschiedenen Editoren (s.o.) im wesentlichen fünf Komponenten: 1) Einen Dateibaum (nicht in Abb. 6 enthalten), 2) einen Editor für Zustandsautomaten (oben, links und rechts), 3) ein *Properties-Tab*, das aktuelle Eigenschaften eines im Editor selektierten Elements anzeigt und dessen Bearbeitung ermöglicht, 4) eine Fehlerübersicht (*Problems-Tab*) und 5) das *Simulations-Tab* (in Abb. unten).

Zur Prüfung der Korrektheit der Modelle wird eine statische Modellprüfung durchgeführt. Dazu werden die in der UML definierten CONSTRAINTS berücksichtigt [OMG10]. Um sicherzustellen, dass die Modelle ausgeführt werden können, wurden weitere Bedingungen identifiziert, dem Profil hinzugefügt und in die Modellprüfung integriert.

Identifizierte Probleme werden in einer Fehlerübersicht (*Problems-Tab*) angezeigt. Die Fehlerquellen in Diagrammen können direkt angesprungen und Fehler aufgrund möglichst eindeutiger Fehlerbeschreibungen gezielt behoben werden. Dies erleichtert vor allem in großen Modellen die Fehlerkorrektur. Zur Realisierung wurde eine direkte Verlinkung von Fehlern mit den zugehörigen Quelldiagrammen in Kombination mit Hervorhebung der Fehlerquellen im Diagramm implementiert.

Die wichtigste Komponente des Werkzeugs ist der Interpreter (vgl. [Zil09]). Er ermöglicht die Validierung des dynamischen Verhaltens durch Ausführung der Zustandsdiagramme.

[1] Eclipse IDE - http://www.eclipse.org/

Die Implementierung des Interpreter basiert auf der formalen Definition des Verhaltens von UML-Zustandsautomaten durch Dausend [Dau07] und Kohlmeyer [Koh09]. Im Zuge der Implementierung wurde die Semantik auf UML-Version 2.2 angepasst und in eine objektorientierte Implementierung überführt [Zil09] (vgl. Abb. 6).

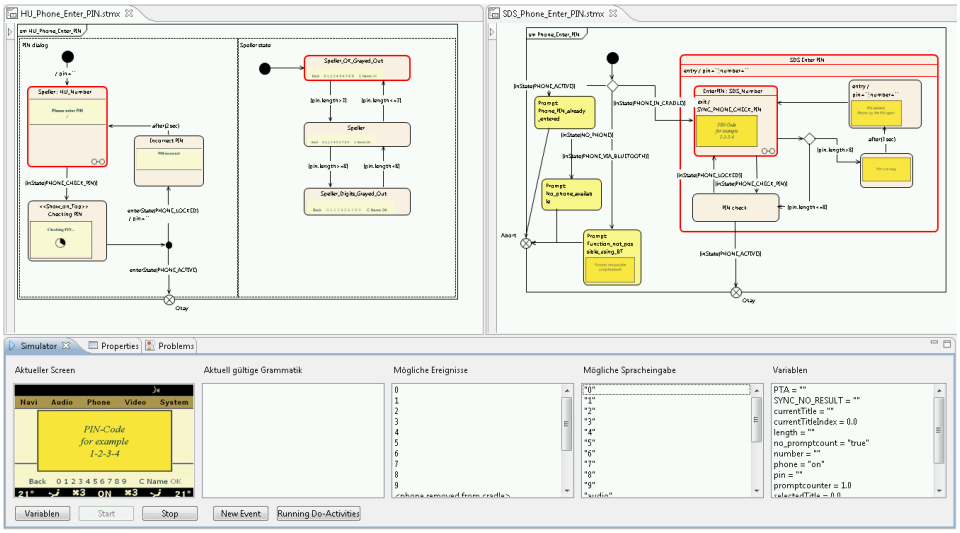


Abbildung 6: Zwei parallel aktive Zustandsautomaten eines multimodalen interaktiven Systems zur PIN-Eingabe im Fahrzeug: Den Zustandsautomat der graphisch-haptischen Modalität zur PIN-Eingabe (links) und den entsprechenden Sprachdialog unter Verwendung eines Teleprompters (rechts). Aktive Zustände während der Simulation sind rot hervorgehoben. Das Simulationsfenster (unten) dient der Anzeige des aktuellen Systemzustands und der Steuerung der Simulation.

Das in Abschnitt 3.3.2 beschriebene dynamische Verhalten wurde in der Implementierung umgesetzt. Die aktuell gültige Grammatik einer Zustandskonfiguration zur Laufzeit wird berechnet und angezeigt. Außerdem wird überprüft, ob Zustandsübergänge aufgrund von Spracheingaben der aktuell gültigen Grammatik erlaubt sind.

Zur Wiedergabe von Prompts wurde bislang auf die Einbindung einer Sprachsynthese verzichtet. Statt dessen wird die Wiedergabe simuliert indem in Abhängigkeit von der Promptlänge die Generierung eines COMPLETION EVENTS für den Zustand verzögert wird. DO-Verhalten werden nach dem Betreten eines Zustands gestartet (vgl. Abb. 2) und können danach jederzeit beendet werden, um ihren Einfluss auf das Modell zu überprüfen.

Abbildung 6 zeigt das Werkzeug während der Ausführung eines multimodalen Modells, das die vorgestellten Konzepte verwendet. Konkret handelt es sich um die Eingabe einer PIN zum Endsperren eines Mobiltelefons unter Verwendung eines Fahrzeug UIs (vgl. [DP10]). Während der Interpretation werden aktive Zustände in geöffneten Editoren rot hervorgehoben. In dieser Situation wurde der Sprachzustand aktiviert, so dass die PIN-Eingabe nun sprachlich und per Nummernblock (vgl. [DP10]) gleichzeitig möglich ist. Zur Modellierung der Zifferneingabe werden in beiden Modalitäten Dialogbausteine verwendet, die als Unterzustände eingebunden wurden und derzeit ebenfalls aktiv sind.

Die aktuelle Bildschirmrepräsentation wird entsprechend des in Abschnitt 3.4.1 vorgestellten Konzepts bestimmt (vgl. Abb. 4) und im *Simulations-Tab* (vgl. Abb. 6, unten links) dargestellt. Außer den Widgets der sichtbaren aktiven Zustände wurden weitere Widgets einbezogen, z.B. die Navigations- und Statusleiste.

Das *Simulations-Tab* dient der Überwachung und Steuerung des Interpreters. Es bietet die Möglichkeit im Modell verwendete Variablen zentral mit initialen Werten zu belegen und die Interpretation zu starten bzw. zu stoppen. Zur effektiven Steuerung des Interpreters werden EVENTS und mögliche *Spracheingaben*, die zu einem Zustandsübergang führen können, angezeigt. Sie werden durch Selektieren gefeuert und es können jederzeit zusätzliche Events definiert und gefeuert werden. Außerdem ist die Steuerung des Interpreters über eine TCP-Schnittstelle möglich.

5 Verwandte Arbeiten

De Melo [Mel10] 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 [OMG10] werden beachtete Ansätze wie CTT [PMM97], UsiXML [Van05] 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 nicht vorgesehen.

Die Grundlage von UsiXML bilden abstrakte Modelle, die mit Hilfe der Definition von Kontexten 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. Jedoch können UML-Modelle nicht direkt integriert werden. Eine Standardisierung von UsiXML ist in Arbeit.

Bisherige Ansätze zur Modellierung von interaktiven Systemen mit UML setzen Schwerpunkte bei statischen Aspekten, wie bei der Strukturierung der Präsentationselemente beispielsweise durch die Verwendung von Stereotypen für Klassendiagramme (WISDOM) [NC00], oder konzentrieren sich auf Aspekte des konzeptuellen Designs, der Navigation oder der Präsentation (UWE) [HK01]. 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 [Köl02] und [GMB06]. Beide Arbeiten unterscheiden sich aber, wie im folgenden näher beschrieben, von unserem Ansatz.

Dialog Statecharts [Köl02] adaptieren Harel Statecharts um Dialoge mit wechselnder In-

ISBN 978-3-88579-286-4

itiative zu modellieren. Die Notation und die Semantik der Statecharts wurde dabei umfangreich verändert und an die Bedürfnisse der Sprachdialogmodellierung angepasst, so dass das Verhalten letztlich vom verwendeten Dialogsystem abhängt.

Gornonzy und andere [GB05, GMB06] schlagen ein Konzept zur Modellierung multimodaler Interaktion mit UML-Zustandsautomaten vor. Die Umsetzung der Konzepte ist jedoch nicht durchgängig UML-konform. In 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 [Mel10] 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. [HK01, MS07]. 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.

6 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 [DP10] hat gezeigt, dass der Ansatz sowohl hinsichtlich Tauglichkeit als auch Benutzerfreundlichkeit als geeignet angesehen wird, um

ISBN 978-3-88579-286-4

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

[Dau07] M. Dausend. Entwicklung einer ASM-Spezifikation der Semantik der Zustandsautomaten der UML 2.0. Diplomarbeit, Universität Ulm, 2007.

[DP10] M. Dausend und M. Poguntke. Spezifikation multimodaler interaktiver Anwendungen mit UML. In *Mensch & Computer*, Seiten 215–224. Oldenbourg Verlag, 2010.

[FFVM04] L. Fuentes-Fernández und A. Vallecillo-Moreno. An Introduction to UML Profiles. *The European Journal for the Informatics Professional*, 5(2):6–13, 2004.

[GB05] S. Goronzy und N. Beringer. Integrated Development and on-the-Fly Simulation of Multimodal Dialogs. In *Ninth European Conference on Speech Communication and Technology*. ISCA, 2005.

[GMB06] S. Goronzy, R. Mochales und N. Beringer. Developing Speech Dialogs for Multimodal HMIs Using Finite State Machines. In *Ninth International Conference on Spoken Language Processing*, Seiten 1774–1777, Pittsburgh, Pennsylvania, 2006. ICSLP, ISCA.

[HK01] R. Hennicker und N. Koch. 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*, Jgg. 200, 2001.

[Koh09] Jens Kohlmeyer. *Eine formale Semantik für die Verknüpfung von Verhaltensbeschreibungen in der UML 2*. Dissertation, Universität Ulm, 2009.

[Köl02] Anke Kölzer. *Diamod: Ein Werkzeugsystem zur Modellierung natürlichsprachlicher Dialoge*. Dissertation, Universität Koblenz, 2002.

[Mei10] Gerrit Meixner. *Entwicklung einer modellbasierten Architektur für multimodale Benutzungsschnittstellen*. Dissertation, TU Kaiserslautern, 2010.

[Mel10] Guido de Melo. *Modellbasierte Entwicklung von Interaktionsanwendungen*. Dissertation, Universität Ulm, 2010.

[MS07] C. Martins und A. Silva. Modeling User Interfaces with the XIS UML Profile. In *Proc. 9th Int. Conf. Enterprise Information Systems*, number 9, Funchal, Portugal, 2007. ICEIS, Springer.

[MWK04] R.A. Merrick, B. Wood und W. Krebs. Abstract User Interface Markup Language. *Developing User Interfaces with XML: Advances on User Interface Description Languages*, Seiten 39–46, 2004.

[NC00] N.J. Nunes und J.F. Cunha. Towards a UML Profile for Interaction Design: The Wisdom Approach. *Lecture Notes in Computer Science*, 1939:101–116, 2000.

[OMG10] OMG. Unified Modeling Language Superstructure v2.3, 2010.

[PMM97] F. Paternò, C. Mancini und S. Meniconi. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, Seiten 362–369, 1997.

[Sel11] B. Selic. The Theory and Practice of Modelling Language Design for Model-Based Software Engineering — A Personal Perspective. *Lecture Notes in Computer Science*, 6491:290–321, 2011.

[TLBT08] J. Tapper, M. Labriola, M. Boles und J. Talbot. *Adobe Flex 3*. Pearson Education, 2008.

[Van05] J. Vanderdonckt. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In *Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE*, Jgg. 5, Seiten 16–31. Springer, 2005.

[W3C04] W3C. Speech Recognition Grammar Specification 1.0, 2004.

[Zil09] Jonas Zilles. Entwicklung eines Interpreters für Zustandsautomaten multimodaler Systeme. Diplomarbeit, Universität Ulm, 2009.