Eine Taxonomie für Aufgabenmodelle

Gerrit Meixner, Daniel Görlich

Zentrum für Mensch-Maschine-Interaktion (ZMMI)
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Trippstadter Str. 122, 67663 Kaiserslautern
{Gerrit.Meixner, Daniel.Goerlich}@dfki.de

Abstract: Dieser Beitrag beschreibt eine Taxonomie für Aufgabenmodelle, die ermöglicht, Aufgabenmodelle auf ihre Verwendbarkeit im Rahmen eines modellbasierten Entwicklungsprozesses hin zu analysieren und zu evaluieren. Dadurch können vorhandene Aufgabenmodelle geprüft und verbessert werden. Zudem erhalten Entwickler durch diese Taxonomie ein Hilfsmittel zur Auswahl eines passenden Aufgabenmodells für ihre eigenen Entwicklungsprozesse. Exemplarisch wird die Taxonomie auf die Useware Markup Language (useML) angewandt, wodurch Verbesserungspotenziale von useML aufgezeigt werden.

1 Einleitung

Schon seit langem wird versucht, die Interaktion zwischen Mensch und Maschine zu vereinfachen und zu verbessern. Heute erfolgt sie zumeist über grafische Benutzungsschnittstellen, doch Szenarien wie die Interaktion von Menschen mit Geräten mittels multimodaler Schnittstellen (z.B. visuell, haptisch, akustisch) werden in Zukunft eine bedeutendere Rolle spielen [Zü04]. Auch die stetig steigende Anzahl heterogener Plattformen (PC, Smartphone, PDA, etc.) führt dazu, dass Benutzungsschnittstellen auf einer Vielzahl von Zielplattformen konsistent gehalten werden müssen, damit ihre intuitive Handhabung und somit die Zufriedenheit der Nutzer gewährleistet wird [Lu04]. Aspekte wie Wiederverwendbarkeit, Flexibilität und Plattformunabhängigkeit gewinnen daher bei der Entwicklung von Benutzungsschnittstellen zunehmend an Bedeutung. Um den wiederkehrenden Entwicklungsaufwand für Einzellösungen jeweils für spezifische Plattformen, Modalitäten oder Nutzungskontexte zu mindern, bietet sich ein modellbasierter Ansatz an, der die Bedürfnisse und Anforderungen der Nutzer in den Fokus rückt. Im Zentrum eines solchen modellbasierten, nutzerzentrierten Entwicklungsprozesses muss ein Aufgabenmodell zur Repräsentation von Nutzeraufgaben stehen [PTV03]. Die Vielzahl verfügbarer Modellierungssprachen für Aufgabenmodelle erschwert allerdings die Auswahl derjenigen Sprache, die am besten zum jeweiligen Projekt passt. Im Rahmen dieses Beitrages wird daher eine Taxonomie eingeführt, die ermöglicht, Aufgabenmodellierungssprachen auf ihre Einsetzbarkeit in ganzheitlichen, modellbasierten Entwicklungsprozessen für Benutzungsschnittstellen hin zu evaluieren. Die Taxonomie wird zunächst allgemein vorgestellt (Kapitel 2) und anschließend exemplarisch auf die Useware Markup Language (useML) angewandt (Kapitel 3). Kapitel 4 gibt nach der Zusammenfassung einen Ausblick auf weiterführende Arbeiten.

2 Die Taxonomie und ihre Bewertungskriterien

Die Taxonomie setzt ihren Bewertungsfokus auf die Integration eines Aufgabenmodells in eine Architektur zur modellbasierten Entwicklung von Benutzungsschnittstellen, die möglichst auch konsistent mit verschiedenen Modalitäten und auf verschiedenen Plattformen bzw. Endgeräten einheitlich zu bedienen ist. Zur Evaluierung von Aufgabenmodellen werden Kriterien definiert, die auf den Arbeiten von [WVE98] und [BOP03] basieren, diese aber um zusätzliche Kriterien ergänzen. Sie alle werden nun näher beschrieben, wobei auch Korrelationen zwischen mehreren Kriterien aufgezeigt werden.

Kriterium 1: Modellierungsmächtigkeit. Ein Aufgabenmodell muss dem Entwickler die Konzentration auf Aufgaben, Aktivitäten und Handlungen erlauben [Pa99]. Dabei stellt die **Granularität der Aufgabenbeschreibung** einen vorrangigen Aspekt dar. Der Einsatz des Aufgabenmodells in einem modellbasierten Entwicklungsprozess erfordert verschiedene Abstraktionsstufen [LPV01]: Mit abstrakten Aufgaben wird die Modellierung übergeordneter Aufgaben beschrieben, während konkrete Aufgaben elementare bzw. atomare Handlungen sind. Allzu feingranulare, konkrete Handlungen wie z.B. Tastendrücke in GOMS [CMN83] sollten aber nicht modelliert werden [Pa03], weil dadurch bereits eine bestimmte Eingabemodalität (hier: eine Tastatur) vorausgesetzt würde.

Nach allgemeiner Auffassung besitzen Menschen mental zumeist individuelle, hierarchische Abbildungen von auszuführenden Aufgaben [Se88]. Die **hierarchische Struktur** entspricht somit der intuitiven Vorgehensweise von Menschen zur Problemlösung und Aufgabenbewältigung. Konsequent werden dabei komplexe Aufgaben in einfacher lösbare Unteraufgaben aufgeteilt [Di00]. Allerdings können Aufgaben aus unterschiedlichen Perspektiven modelliert werden [PTV03] – speziell sollten Aufgabenmodelle zwischen **interaktiven Benutzer-** und **reinen Systemaufgaben** unterscheiden [BS98].

Ein weiterer Aspekt der Modellierungsmächtigkeit betrifft den **Grad der Formalisierung**. Vielfach werden informelle Beschreibungen (Use Cases, Instructional Text) als Ausgangsbasis für die Modellierung genutzt, jedoch mangelt es diesen Beschreibungen an einer formalen Basis, was ihre nahtlose Integration in einen modellbasierten Entwicklungsprozess erschwert. Zur automatischen Prüfung von Aufgabenmodellen auf Korrektheit müssen eine ausdruckfähige Semantik und eine klare Syntax bereitgestellt werden. Als optimal gilt ein **semi-formaler Formalisierungsgrad** [OPB98].

Temporaloperatoren erlauben, zeitliche Abhängigkeiten und Ordnungen semantisch ausdrucksstark zu modellieren [Di04] [Pa99] [Di00]. Für die Generierung von Dialogmodellen aus Aufgabenmodellen werden Transformatoren benötigt, die zunächst Temporaloperationen des Aufgaben- auf Transitionen des Dialogmodells abbilden. Die zeitliche Abfolge einzelner Aufgaben ist ein wesentlicher Aspekt der Entwicklung von Benutzungsschnittstellen [Pa03] und öffnet erst den Weg zu vollständig modellbasierter Entwicklung [LPV01]. Auch die Kennzeichnung der **Optionalität** von Aufgaben ist essentiell [BOP03]. Mit ihr eng verbunden erlaubt die Angabe der **Mehrfachausführung** (Iterationen, Kardinalität), Ausführungshäufigkeiten für Einzelaufgaben zu formalisieren. Zuletzt können **Konditionen** festlegen, unter welchen logischen [Re03] oder zeitlichen Bedingungen Aufgaben überhaupt ausgeführt werden. Letztere werden, im

Gegensatz zu Temporaloperatoren, als quantitative zeitliche Aspekte der Aufgabenmodellierung bezeichnet. Kaum ein Aufgabenmodell ist jedoch in der Lage, quantitative und qualitative zeitliche Aspekte zugleich explizit zu modellieren.

Kriterium 2: Integrationsfähigkeit. Nur wenn ein Aufgabenmodell in eine modellbasierte Architektur zur Generierung von Benutzungsschnittstellen integrierbar ist, können nutzerzentrierte Benutzungsschnittstellen automatisch erzeugt werden [LPV01]. Daher sollten Aufgabenmodelle nicht isoliert, sondern ganzheitlich im Zusammenspiel bspw. mit Dialog- und Präsentationsmodellen betrachtet werden. Dabei ist speziell die eindeutige Identifikation aller modellierten Aufgaben zu beachten, um ihre konsistente Zuordnung zu Interaktionsobjekten der Benutzungsschnittstelle zu gewährleisten.

Kriterium 3: Kommunizierbarkeit. Obwohl nicht explizit für die Kommunikation zwischen Menschen gedacht, eignen sich Aufgabenmodelle jedoch zur Verbesserung der Kommunikation in Teams und zum Endnutzer [PTV03]. Aufgabenmodelle können u.a. zur Darstellung von Nutzeranforderungen [BOP03], zur Evaluation [Re03], zur Simulation und zur interaktiven Validierung genutzt werden. Der Fokus dieses Kriteriums liegt daher auf der Les- und Erlernbarkeit (Verständlichkeit) des Aufgabenmodells. Eine grafische, semi-formale Notation erweist sich dabei als überaus nützlich [OPB98].

Kriterium 4: Manipulationsfähigkeit. Dieses Kriterium beschreibt, wie einfach das Aufgabenmodell durch die Entwickler erstellt und manipuliert werden kann [BB95]. Generell unterscheidet man zwischen rein textuellen Notationen wie GOMS [CMN83] und grafischen Notationen wie CTT [Pa99] und GTA [WVE98]. Bei der Erstellung von Aufgabenmodellen sind grafische Notationen deutlich gebrauchstauglicher als textuelle [Di04], da sie u.a. die hierarchische Aufgabenstruktur deutlicher visualisieren können [Pa03]. Zur Manipulation bzw. Bearbeitung von Aufgabenmodellen sind dementsprechend grafische Notationen und Editoren notwendig [PTV03].

Kriterium 5: Anpassungsfähigkeit. Dieses Kriterium subsumiert die Möglichkeiten der flexiblen Anpassung eines Aufgabenmodells an neue Situationen und Domänen. Es korreliert mit den Kriterien 1 und 6. Umfangreiche Anpassungen können etwa bei der Entwicklung von Benutzungsschnittstellen für verschiedene Modalitäten und Plattformen notwendig werden. Des Weiteren sollte ein Aufgabenmodell Informationen bspw. über Nutzergruppen oder Bedienorte einbinden können.

Kriterium 6: Erweiterbarkeit. Dieses Kriterium, korreliert mit den Kriterien 1 und 5, bewertet, wie einfach eine Notation semantisch erweitert werden kann, da bisher kein Aufgabenmodell allgemeingültig für jeden Anwendungsfall nutzbar ist [BB95]. Generell sind semi-formale Notationen einfacher erweiterbar als formale [BOP03], die auf fundierten mathematischen Theorien basieren und einfache Erweiterungen kaum zulassen.

Kriterium 7: Verarbeitungsfähigkeit. Für den Grad der (automatisierten) Verarbeitung eines Aufgabenmodells ist u.a. relevant, in welcher Form es auf einem Datenträger gespeichert werden kann. Dabei ist insbesondere auf die Verwendung offener Standards wie XML zu achten, da proprietäre Formate Austausch und Verarbeitung von Aufgabenmodellen deutlich erschweren.

3 Anwendung der Taxonomie auf die Useware Markup Language

Zur nutzer- und aufgabenorientierten Bediensystementwicklung wird am Zentrum für Mensch-Maschine-Interaktion der Useware-Entwicklungsprozess verfolgt, der Nutzeraufgaben als zentrale Elemente in so genannten Benutzungsmodellen strukturiert. Ihre Struktur und Semantik sind in der Useware Markup Language (useML 1.0 nach [Re03]) verankert, deren XML-Schema jedoch mittels beliebig definierbarer Attribute (Nutzergruppen, Bedienorte, etc.) an spezielle, projektspezifische Gegebenheiten angepasst werden kann. Die in Analysen ermittelten Aufgabenstrukturen und mentalen Modelle der Nutzer werden dabei in einer hard- und softwareunabhängigen Struktur integriert, die komplexe Aufgaben in Tätigkeiten, Handlungen und zuletzt elementare Tätigkeiten bzw. Operationen unterteilt. Jegliche komplexe Aufgaben aller Nutzergruppen an allen Geräten werden in einer Hierarchie von Benutzungsobjekten (BO's) hinterlegt, die letztlich in elementare Benutzungsobjekte (eBO's) zergliedert werden. useML sieht fünf eBO-Typen vor: ändern, auslösen, auswählen, eingeben sowie informieren [Re03] [Zü04].

Die Aufteilung des Benutzungsmodells in BO's und fünf eBO-Typen wird hier als ausreichend feingranular interpretiert. Anhand der eBO's kann direkt auf korrespondierende, abstrakte Interaktionsobjekte geschlossen werden [Re03], was die gröbere Aufgabentypisierung des de-facto-Standards CTT nicht zulässt [Lu04]. useML erfüllt zudem das Hierarchiekriterium dieser Taxonomie und kann als semi-formal angesehen werden: Zwar basiert useML nicht auf formalen mathematischen Grundsätzen wie etwa die Petri-Netze, jedoch wird der Aufbau der Sprache durch ihr XML-Schema klar definiert, was Syntax- und Konsistenzprüfungen erlaubt.

useML 1.0 rückt die Benutzer in den Vordergrund und kennt daher keine reinen Systemaufgaben. Für eine spätere Anbindung der Applikationslogik an die Benutzungsschnittstelle ist dieser Aufgabentyp aber ebenfalls nötig, da Systemaufgaben durchaus Bestandteil größerer, interaktiver Bedienhandlung sein können. Weiterhin sind in useML 1.0 keine Temporaloperatoren vorgesehen, Mehrfachausführungen von Aufgaben nicht modellierbar und BO's und eBO's können nicht als optional oder erforderlich markiert werden (Das Attribut „gewichtung", das projektspezifische, relative Werte etwa von 1 bis 10 annehmen kann, genügt nicht zur formalen Repräsentation der Optionalität). Zuletzt können zwar logische Vor- und Nachbedingungen spezifiziert werden, nicht jedoch zeitliche Bedingungen und Invarianten. Die Abwägung all jener Unterkriterien lässt daher nur zu, die **Modellierungsmächtigkeit** von useML als **niedrig** zu bewerten.

Die Integrationsfähigkeit des Benutzungsmodells in eine vollständige, modellbasierte Architektur für die Entwicklung von Benutzungsschnittstellen ist schwierig zu beurteilen, da für den Useware-Entwicklungsprozess bislang keine weiteren Modelle definiert wurden. [Lu04] bemängelt insbesondere das Fehlen eines Dialog- und Präsentationsmodells. Für die Integration in eine Gesamtarchitektur müssen das Benutzungsmodell semantisch erweitert und weitere Modelle angebunden werden. Auch eindeutige Kennungen (ID's) von Aufgaben sieht useML nicht vor: BO's und eBO's werden aktuell nur über deren Namen identifiziert, also bloße Zeichenketten, die nicht zur eindeutigen Zuordnung ausreichen. Insgesamt muss die **Integrationsfähigkeit** des Benutzungsmodells in useML daher bisher als **niedrig** eingestuft werden.

Da die Useware-Entwicklung ein interdisziplinäres Vorgehen verlangt, sollten Benutzungsmodelle einfach lesbar und useML leicht erlernbar sein. Für die Betrachtung von useML als XML-Dialekt wird prinzipiell nur ein Text- oder XML-Editor benötigt, was jedoch XML-Kenntnisse erfordert. Besser verständlich ist die webbrowser-ähnlichen Darstellung des useML-Viewers [Re03], der aber weder zur Bearbeitung von Benutzungsmodellen noch für interaktive Simulationen zur Validierung oder Evaluation entwickelt wurde. Insgesamt wird die **Kommunizierbarkeit** von useML lediglich als **mittelmäßig** bewertet. Ein vollwertiger grafischer useML-Editor zur Bearbeitung von Benutzungsmodellen steht aber kurz vor der Fertigstellung [MGS08]. Bis dahin muss useML zwar eine **niedrige Manipulationsfähigkeit** bescheinigt werden; der useML-Editor wird zukünftig aber eine **hohe Manipulationsfähigkeit** gewährleisten und die Effizienz der Erstellung von Benutzungsmodellen mit useML deutlich erhöhen.

useML wurde entwickelt, um die systematische Entwicklung von Bedienschnittstellen zu unterstützen [Re03]. Die Modellbildung ist bereits möglich, bevor Wissen über die Zielplattform(en) vorhanden ist. Zudem kennt useML explizit aus dem Hauptschema ausgelagerte Attribute (Benutzergruppen, Gerätefunktionsmodelle, etc.), die problemlos projektspezifisch angepasst werden können, ohne das Hauptschema anzutasten, wodurch useML für eine Vielzahl von Modalitäten und Plattformen anpassbar ist. So können für zahlreiche Anwendungsdomänen geeignete Schemaattribute vordefiniert werden, weshalb **Anpassungsfähigkeit** und **Erweiterbarkeit** von useML als **hoch** bewertet werden. Zuletzt ist durch die Umsetzung des Benutzungsmodells in useML als XML-Dialekt seine automatische Weiterverarbeitung möglich. Mittels spezieller Transformatoren können Prototypen direkt aus useML erzeugt werden [Re03]. Die automatische **Verarbeitungsfähigkeit** (Transformation) von useML ist somit ebenfalls **hoch** zu bewerten.

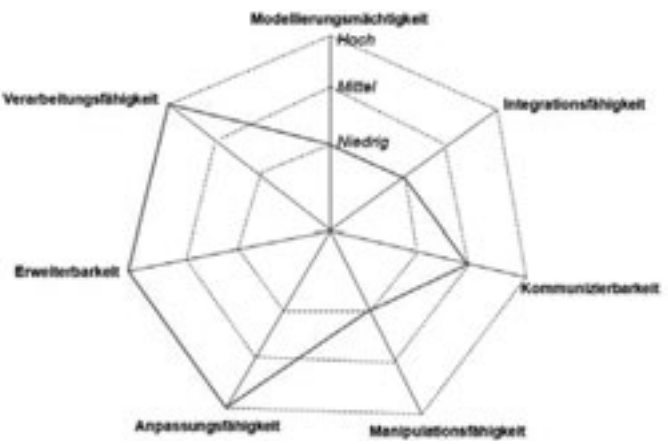

Abbildung 1: Visualisierung der Bewertung des Benutzungsmodells

Die hier angewandte Taxonomie auf useML wurde zum Zwecke der Selbstevaluierung entwickelt und um nachzuprüfen, ob useML den praktischen Anforderungen noch genügt. Zusammenfassend visualisiert Abbildung 1 das Ergebnis der Anwendung als Netzdiagramm und zeigt deutlich die aktuellen Defizite, welche wiederum Ansatzpunkte für anstehende und zukünftige Verbesserungen der Useware Markup Language darstellen.

4 Zusammenfassung und Ausblick

Die hier vorgestellte Taxonomie dient der Bewertung und Evaluation von Aufgabenmodellen, die in modellbasierten Entwicklungsprozessen für Benutzungsschnittstellen genutzt werden. Demnächst soll diese Taxonomie auf weitere Aufgabenmodelle angewandt und ein direkter Vergleich zwischen etablierten Methoden zur Aufgabenmodellierung geschaffen werden. In useML identifizierte Defizite wurden zum Teil bereits in der semantisch erweiterten Version 2.0 und im grafischen useML-Editor integriert [MGS08].

Literaturverzeichnis

[BB95] Brun, P.; Beaudouin-Lafon, M.: A taxonomy and evaluation of formalism for the specification of interactive systems. In: Proc. of the HCI'95 Conference on People and Computers X, 1995.

[BOP03] Balbo, S.; Ozkan, N.; Paris, C.: Choosing the right task modelling notation: A Taxonomy. In: Diaper, D.; Stanton, N. (Hrsg.): The Handbook of Task Analysis for Human-Computer Interaction. Lawrence Erlbaum Associates, 2003; S. 445-466.

[BS98] Bomsdorf, B.; Szwillus, G.: From task to dialogue: Task based user interface design. In: SIGCHI Bulletin, Band 30, Heft 4, S. 40-42.

[CMN83] Card, S. K.; Moran, T. P.; Newell, A.: The psychology of human-computer interaction. Lawrence Erlbaum Associates, 1983.

[Di00] Dittmar, A.: More precise descriptions of temporal relations within task models. In: Proc. of the 7th International Workshop on Interactive Systems: Design, Specification and Verification, 2000; S. 151-168.

[Di04] Dix, A. et al.: Human-Computer Interaction. Prentice Hall, 2004.

[LPV01] Limbourg, Q.; Pribeanu, C.; Vanderdonckt, J.: Towards Uniformed Task Models in a Model-Based Approach. In: Proc. of the 8th International Workshop on Interactive Systems: Design, Specification and Verification, 2001; S. 164-182.

[Lu04] Luyten, K.: Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development. Transnationale Universiteit Limburg, 2004.

[MGS08] Meixner, G; Görlich, D.; Schäfer, R.: Unterstützung des Useware-Engineering Prozesses durch den Einsatz einer modellbasierten Werkzeugkette. USEWARE 2008, Baden-Baden, VDI-Bericht 2041, VDI/VDE-GMA, Düsseldorf, VDI-Verlag, 2008, S. 219-232.

[OPB98] Ozkan, N.; Paris, C.; Balbo, S.: Understanding a Task Model: An Experiment. In: Proc. of HCI on People and Computers, 1998; S. 123-137.

[Pa99] Paternò, F.: Model-based design and evaluation of interactive applications, Springer, 1999.

[Pa03] Paternò, F.: ConcurTaskTrees: An Engineered Notation for Task Models. In: Diaper, D.; Stanton, N. (Hrsg.): The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, 2003; S. 483-501.

[PTV03] Paris, C.; Lu, S.; Vander Linden, K.: Environments for the Construction and Use of Task Models. In: Diaper, D.; Stanton, N. (Hrsg.): The Handbook of Task Analysis for Human-Computer Interaction, Lawrence Erlbaum Associates, 2003; S. 467-482.

[Re03] Reuther, A.: useML – systematische Entwicklung von Maschinenbediensystemen mit XML. Fortschritt-Berichte pak, Band 8. Technische Universität Kaiserslautern, 2003.

[Se88] Sebillotte, S.: Hierarchical planning as a method for task analysis: The example of office task analysis. In: Behavior and Information Technology, Band 7, Heft 3, S. 275-293.

[WVE98] van Welie, M.; van der Veer, G.; Eliens, A.: An ontology for task world models. In: Proc. of the 5th International Workshop on Interactive Systems: Design, Specification and Verification, 1998, S. 57-70.

[Zü04] Zühlke, D.: Useware-Engineering für technische Systeme, Springer, 2004.