Birgit Bomsdorf und Gerd Szwillus

# „UML und Aufgabenmodellierung: Softwaretechnik und HCI im Dialog“ auf der Konferenz Mensch & Computer 2001

## 1. Hintergrund und Zielsetzung

Bei der Entwicklung hochgradig interaktiver Systeme wird zunehmend – nicht zuletzt seitens der entsprechenden Ergonomienormen – die *Aufgabenangemessenheit* der entwickelten Systeme, insbesondere der Benutzungsschnittstelle, gefordert. Ein System kann nur dann aufgabenangemessen entwickelt werden, wenn dem Entwickler die am und mit dem System durchzuführenden Aufgaben bekannt sind, bzw. er selbst am Entwurf einer modifizierten Aufgabendurchführung kreativ beteiligt wird. Damit ist zunehmend die Modellierung von Arbeitsabläufen wesentlicher Bestandteil des Entwurfs der Mensch-Maschine-Schnittstelle, des *User Interface (UI).* Diese Modellierung kann nur aus der Sicht des Anwenders und seiner Aufgaben heraus erfolgreich sein, unabhängig von einer denkbaren softwaretechnischen Umsetzung der die Arbeitsabläufe unterstützenden Benutzungsschnittstelle. Andererseits muss das UI aber letztlich als ein Stück Software betrachtet, und somit auch als solches beschrieben, modelliert und implementiert werden. Es wird deutlich, dass hier zwei Modellierungswelten, zwei Gestaltungsräume aneinander grenzen, die einerseits sehr verschiedenartige Aspekte des zu entwickelnden Systems behandeln und andererseits zueinander „kompatibel“ bearbeitet werden müssen.

Auf der Seite der Softwareentwicklung (SE-Entwicklung) hat sich mittlerweile die UML *(Unified Modeling Language)* als De-facto-Standard auch in der Softwareindustrie durchgesetzt; auf Seiten der Mensch-Maschine-Kommunikation (HCI) wird zunehmend die Verwendung von Aufgabenmodellen propagiert. Derzeit wird typischerweise eines der beiden Konzepte eingesetzt, es zeigt sich aber immer deutlicher, dass Konzepte beider Modellierungsansätze notwendig sind. Dies hat bereits dazu geführt, dass beide Ansätze durch Konzepte erweitert werden, die bereits Bestandteil des jeweils anderen sind. So werden beispielsweise die sog. *Use Cases,* Bestandteil von UML, zunehmend um Komponenten erweitert, die zu den Kernkonzepten der Aufgabenmodellierung zählen. Inzwischen wird auch eine Integration der beiden Ansätze angestrebt.

Die Zielsetzung des Workshops bestand darin, dass die beiden Modellierungswelten (UML auf der einen und Aufgabenmodellierung auf der anderen Seite) einander kennen lernen, um voneinander zu lernen und idealerweise aufeinander zu zugehen. So ging es um die Verdeutlichung der Affinitäten und Unterschiede zwischen beiden Ansätzen sowie die Betrachtung existierender Integrationsideen, um auf dieser Basis Konzepte der Modellierungswelten zueinander in Bezug zu setzen und weitere Integrationsmöglichkeiten zu diskutieren.

## 2. Teilnehmer

Die 34 Teilnehmer des Workshops kamen sowohl aus dem universitären als auch aus dem industriellen Umfeld und beschäftigen sich theoretisch oder praxisorientiert mit der Entwicklung interaktiver Systeme. Der Tätigkeits- und Wissenshintergrund bezog sich dabei insgesamt vorrangig auf die Bereiche Softwaretechnik oder Mensch-Maschine-Kommunikation mit Kenntnissen und Erfahrungen auf dem Gebiet Modellierung von Benutzungsschnittstellen, Systemanalyse, Aufgabenanalyse und objektorientierte Modellierung. Einige der Teilnehmer nutzten den Workshop als Einstieg in die Aufgabenmodellierung oder UML während andere sich bereits mit dem einen oder anderen Aspekt aus der Thematik beschäftigt hatten und dies weiter diskutieren wollten.

## 3. Ablauf

Innerhalb einer allgemeinen Einführungs- und Vorstellungsphase haben die Moderatoren zunächst in die Workshopthematik eingeführt. In sich anschließenden Kurzvorträgen wurden

- die beiden grundsätzlichen Blickwinkel bei der Erstellung interaktiver Systeme,
- grundsätzliche Konzepte der Aufgabenmodellierung,
- im Kontext des Workshops relevante Konzepte aus UML sowie
- die Bezüge zwischen diesen Modellierungsansätzen

dargestellt. Innerhalb des letzten Themenpunktes wurde von den Moderatoren basierend auf existierenden Beiträgen zur Integration der Modellierungswelten ein verallgemeinerter Ansatz vorgestellt. Daneben berichteten zwei Teilnehmer über ihre speziellen Lösungsansätze. So ging Peter Forbrig auf die Beziehungen zwischen Aufgabenmodellierung und objektorientierter Softwareentwicklung ein, während Mario Winter das System SCORES zur objektorientierten Anforderungsermittlung mit der UML berichtete.

An diese Vorstellungs- und Vortragsphase schloss sich als Kern des Workshops eine Diskussion an. Ausgangs-

punkt waren drei Themenkomplexe, die von den Moderatoren vorgeschlagen wurden. Aus diesen wurden von den Teilnehmern die folgenden Punkte ausgewählt, die dann in Untergruppen diskutiert wurden:

- **Integration der Aufgabenmodellierung in UML**
  Die Vorgabe seitens der Veranstalter für die Diskussion war die Betrachtung *notwendiger Spracherweiterungen zur Aufgabenmodellierung und Bezüge zu bisherigen UML-Elementen*. Somit ging es in dieser Untergruppe um die Betrachtung aufgabenmodell-orientierter Aspekte und ihrer Behandlung oder Nicht-Behandlung in UML.
- **Gesamtheitlicher Entwicklungsprozess**
  Innerhalb der Bereiche SE und HCI existieren verschiedenartige Entwicklungsprozesse. In dieser Untergruppe sollten die *Reihenfolgen und Abhängigkeiten* innerhalb der in den Bereichen SE und HCI jeweils existierenden Entwicklungsschritte und der eingesetzten Diagrammarten betrachtet werden, um so Ansatzpunkte eines einheitlichen Entwicklungsprozesses zu finden.

## 4. Integration der Aufgabenmodellierung in UML

Die Untergruppe hat die Diskussion notwendiger Spracherweiterungen zur Aufgabenmodellierung und Bezüge zu bisherigen UML-Elementen durch Betrachtung der folgenden zwei Fragen angegangen:

- Welche Informationen aus dem Umfeld „Aufgabenmodellierung" möchte man in der Praxis gerne festhalten, kann es aber nicht mit UML?
- Falls man diese Informationen modellieren könnte, wie würde man dann damit umgehen?

### 4.1 Defizite von UML in Bezug auf Informationen der Aufgabenmodellierung

Zunächst wurde deutlich bemerkt, dass es Defizite von UML gibt. Einige Teilnehmer erwähnten, dass neben der UML-Modellierung weitere Darstellungstechniken verwendet wurden, um in eigenen Strukturen, Tabellen und Dokumenten Informationen aus der Aufgabenanalyse festzuhalten, die in und mit UML nicht formulierbar sind.

**Benutzermodellierung:** So fehlt etwa die Möglichkeit, den Benutzer zu modellieren – seine Fähigkeiten, Vorkenntnisse, Voraussetzungen. Die andiskutierte Möglichkeit, den Benutzer als „Software"-Objekt in eine UML-Spezifikation einzubinden, wurde allgemein als unzureichend empfunden, insbesondere da echte Software-Objekte sich anders verhalten als Menschen (Nichtdeterminismus, Fehlerverhalten, Asynchronität).

**Benutzeraufgaben:** UML modelliert vorwiegend Systemabläufe und nicht das Verhalten, die Ziele und die Aufgaben des Benutzers. Betrachtet man etwa die Benutzung eines Fahrkartenautomaten, dann ist dies für den Benutzer eine Aufgabenerledigung, die zum Beispiel 10 Minuten dauern kann, weil der Benutzer nachdenken muss, die Bedienung verstehen muss und in der Anleitung nachlesen muss, was wie zu tun ist. Für den mit UML modellierten Automaten stellt sich der Verkauf als zum Beispiel 5-Sekunden-Prozess dar, weil sich dieses Modell ausschließlich um die Reaktionen des Automaten auf den dann endlich erfolgenden Knopfdruck des Benutzers bezieht.

**Benutzungskontext:** UML modelliert überhaupt nicht den Benutzungskontext. Die Gruppe diskutierte in diesem Zusammenhang das Szenario der „Oma", die Geld am Automaten abholen will: Die Bank fordert von ihren Kunden, mit dem Geldautomaten umgehen zu müssen; manche Benutzer – wie zum Beispiel die erwähnte „Oma" – haben jedoch Angst vor dieser Umstellung und werden möglicherweise daraufhin beschließen zu einer Bank zu wechseln, die diese Forderung nicht stellt. Ein bewusster Umgang mit derartigen Forderungen des Benutzerkontextes ist nötig, lässt sich aber mit UML in keiner Weise notieren. Ähnlich wie beim schon erwähnten Defizit beim Benutzermodell gibt es dafür keinerlei Modellierungsansätze. Die Gruppe war sich einig, dass auch Aufgabenmodellierungsansätze diese Aspekte teilweise vernachlässigen, indem sie sich auf die reine Ablaufbeschreibung konzentrieren bzw. beschränken. Im Contextual Design gibt es für derartige Informationen die sogenannten Cultural Diagrams. Diese werden zwar nicht direkt in Design umgesetzt, aber ihre bloße Existenz schaffen eine Betonung dieser Fragen, was zu ihrer expliziten Berücksichtigung führt.

**Umgang mit Hierarchie:** UML bietet teilweise hierarchische Strukturen, wird aber generell in dieser Hinsicht als etwas schwach empfunden. In Bezug auf die Aufgabenmodellierung, in der Hierarchien stark betont sind, wurden besonders die Aktivitätendiagramme erwähnt. Gefordert werden hier einerseits Zerlegungsmechanismen, um große Komponenten in hierarchisch gegliederte Komponenten aufzuspalten. Für die Use-Cases bietet das SCORES-Konzept hierfür eine Teilantwort. Zum anderen wurde aber betont, dass Hierarchiebildung auch zum Problem der Endekriterien führt: Auf welcher Ebene „endet" die Hierarchie, wann macht es keinen Sinn mehr in der bisherigen Denkweise weiter aufzuspalten, sondern einen Paradigmenwechsel einzuleiten. Dadurch müsste man dann Blattknoten der Hierarchie zu „Endknoten" erklären und einen „Diagrammwechsel" herbeiführen – die Endknoten entsprechen dann wieder einem anderen Diagramm. Dieses bei Hierarchisierung generelle Problem wird in UML kaum angegangen – die verschiedenen Diagramme sind teilweise zu wenig miteinander verknüpft, andere Techniken gelten als reicher was diese Hierarchisierungsfragen angeht. Ein anderes Problem in dieser Hinsicht scheint zu sein, dass etwa bei Use-Case-Diagrammen die Stufe „n" nicht die Summe der Teile von „n" ist, sondern mehr.

### 4.2 Verwendung von Informationen aus der Aufgabenmodellierung

Die zweite Fragestellung wurde deutlich losgelöst von UML und Aufgabenmodellierung diskutiert, stellt aber interessante Beziehungen zwischen Modellierung und Entwicklungstätigkeit her.

Die Gruppe hielt fest, dass es eine große Lücke gibt zwischen dem Sammeln und Hinschreiben von Information zur Aufgabenmodellierung und dem systematischen Verwenden dieser Information. Wie setzt man Information der Aufgabenanalyse konkret in Design um? Es ist allgemeine Meinung, dass man da-

für „modelltechnische Unterstützung" braucht – das reine Hinschreiben von Information reicht nicht aus. Vielmehr sollte es Notationen, Diagramme, Formalismen oder im weitesten Sinne Sprachen geben, die eine nachvollziehbare Umsetzung von Anforderungen etwa aus der Kontextanalyse in Designentscheidungen geben. Dafür müssten geeignete Modelle existieren.

Dies warf die Frage nach dem **Sinn von Modellierung** überhaupt auf. Allgemein wurde festgehalten, dass Modellierung viel Arbeit ist – und teilweise wenig Nutzen bringt. Modellierungsarbeit würde (auch vom Management) teilweise als Zeitverschwendung angesehen, ihr Nutzen für die späteren Phasen der Entwicklung bezweifelt. Die Gruppe hielt fest, dass Modellierung zumindest bei Informatikern suggeriert, dass anschließend aus dem Modell „irgendwas" **generiert** wird – das ist aber in vielen Fällen von Modellen nicht der Fall. Aber die Gruppe war sich einig, dass ein wesentlicher Nutzen von Modellen – auch wenn nichts generiert werden kann – in der **strukturierten Dokumentation** von Sachverhalten besteht. Diese mögen zwar nicht direkt in Code umsetzbar sein, sind aber ungeheuer hilfreich um bei Änderungen Argumentationen und Begründungen für eine Systemeigenschaft nachvollziehen zu können. Gezieltes Aufsuchen von Zusammenhängen wird dadurch stark erleichtert oder überhaupt erst möglich, was mit zunehmenden Umfang in informellen Textdokumenten nicht mehr möglich ist. Eine Teilantwort auf diese Forderung stellt das SHIRA-System von Rainer Wessler dar (vorgestellt im Workshop W12 der Mensch & Computer 2001): Dieses leistet eine nachvollziehbare Dokumentation von erwünschten Eigenschaften eines Systems bis hin zu konkreten Designentscheidungen. Dies ist ein Ansatz, der Kontext-Forderungen an ein System operationalisiert und diese nicht als hehre Forderungen ist „Vorwort" eines Lastenheftes verdrängen, wo diese durch ihre Informalität zuwenig Gewicht haben und bei der konkreten Entwicklungsarbeit nur nachrangig berücksichtigt werden.

Eine andere Sichtweise auf Systementwicklung ist die **„Design-For-Change"**-Idee. Ein System wird von vorneherein so entwickelt, dass es „vor Ort" angepasst werden kann, im Moment der Auslieferung und Installation quasi „zu Ende entwickelt" wird. In der Analyse- und Modellierungsphase werden diese „Lücken" im Wissen über Systemeigenschaft explizit gemacht und adäquat behandelt. Die Modellierung verträgt „unvollständiges Wissen", es fehlen ausdrücklich Informationen und das Design ist derart flexibel, dass es akzeptiert, dass das Verständnis der Aufgabe für ein endgültiges Design noch nicht ausreicht.

## 5. Gesamtheitlicher Entwicklungsprozess

Ziel dieser Thematik war die Diskussion eines gesamtheitlichen Entwicklungsprozesses, der die Entwicklungsschritte der Bereiche SE und HCI, insbesondere die dort eingesetzten Techniken und Modelle, zueinander in Bezug setzt. Hierzu wurden die „typischen" Entwicklungsphasen beider Bereiche betrachtet.

### 5.1 Entwicklungsphasen im Bereich SE

Die Phasen im Bereich SE ergaben sich sehr stringent; es wurde prinzipiell der klassische Softwareentwicklungsprozess beschrieben. Augenmerk lag hierbei jedoch auf den Schritten, innerhalb derer der zu implementierende Funktionsumfang der zu realisierenden Anwendung herauskristallisiert und definiert wird. Es wurden folgende Phasen genannt:

*Aufgabenstellung:* Bei der Beschreibung der Aufgabenstellung geht es um die Ermittlung der Anforderungen an das zu erstellende System. Zielsetzung ist dabei noch nicht die Fixierung der funktionalen Anforderungen, sondern vielmehr eine allgemeinere Beschreibung der Anwendung, also deren Zielsetzung und Leistungsvermögen.

*Bestandsaufnahme:* Innerhalb der Bestandsaufnahme werden Komponenten identifiziert, die bei der Realisierung der neuen Anwendung wieder verwendet werden können. Hierzu werden beispielsweise Pattern und Programmbibliotheken untersucht.

*Funktionale Anforderungen:* Auf Basis der Aufgabenbeschreibung und der Bestandsaufnahme werden die funktionalen Anforderungen beschrieben. Zur Modellierung eignet sich hier insbesondere der Einsatz von Use Cases.

*Nichtfunktionale Anforderungen:* Neben den funktionalen sind auch nichtfunktionale Anforderungen zu ermitteln und zu beschreiben. Diese umfassen Aspekte, die den Kontext einer Anwendung und die Umgebung ihres Einsatzes betreffen.

*Design:* Ausgehend von den Use Cases kommen innerhalb des Systemdesigns vor allem Sequenzdiagramme aber auch weitere Beschreibungstechniken der UML zum Einsatz.

*Implementierung und Test:* Die Inhalte dieser Phasen wurden als wichtig hinsichtlich einer gesamtheitlichen Entwicklung gesehen, jedoch in dieser Untergruppendiskussion nicht detailliert betrachtet.

### 5.2 Entwicklungsphasen im Bereich HCI

Hinsichtlich der Entwicklungsphasen im Bereich HCI wurden verschiedene Vorgehensweisen und Zusammenhänge gesehen. Es wurde ein möglicher Prozess mit folgenden Phasen skizziert:

*Aufgabenanalyse:* Die Aufgabenanalyse ist der Phase der Ist-Analyse des SE vergleichbar. Zentral in der Aufgabenanalyse sind die Benutzer und deren Aufgaben (u. a. die Art der Aufgabendurchführung und die dabei involvierten Gegenstände). Hierbei ist es wichtig, auch den Benutzungskontext eines Systems zur Aufgabendurchführung festzuhalten. Die Aufgabenanalyse wurde von einigen Teilnehmern nicht als Teil der Aufgabenmodellierung gesehen. Übereinstimmung bestand hingegen hinsichtlich der einzusetzenden Techniken wie Interviews und Fragebogen.

*Zielsetzungen für ein System:* Im Hinblick auf das zu entwickelnde System sind dessen Funktionen (Beitrag bzw. Unterstützung des Systems bei der Aufgabendurchführung) festzuhalten. Dies erfolgt noch auf abstrakter Ebene. Das resultierende Dokument entspricht damit noch keiner funktionalen Anforderungsbeschreibung, sondern liefert Vorgaben für diese.

*Aufgabensynthese:* Innerhalb der Aufgabensynthese (teilweise von den Teilnehmern auch als Aufgabenanalyse bezeichnet) geht es nun um den Entwurf des zukünftigen Systems aus Benutzersicht. Die hier entstehenden Do-

kumente beschreiben, wie der Benutzer seine Aufgaben in Zukunft mithilfe der neuen Anwendung durchführen wird.

*Identifizierung der Aufgabenobjekte:* Mit der Modellierung der Aufgabenobjekte werden sowohl die Gegenstände und Mittel beschrieben, die für die Aufgabenerledigung relevant sind, als auch Aspekte der Umgebung insgesamt, in der die Aufgaben durchgeführt werden. Innerhalb dieser Phase wird von einigen Teilnehmern bei ihrer Entwicklungsarbeit bereits eine objektorientierte Denkweise eingenommen, was sich einerseits in der Wahl der verwendeten Techniken (wie Objekt- und Klassendiagramme) und andererseits in der Vorgehensweise zeigt, diese Phase nicht von der nachfolgend beschriebenen (Identifizierung der (UI)-Objekte) zu differenzieren.

*Identifizierung der (UI)-Objekte:* In dieser Phase werden sowohl UI- als auch Software-Objekte mit ihren Attributen und Methoden festgelegt. Da diese vielfach aus den Aufgabenobjekten resultieren, wird eine entsprechende Unterscheidung häufig nicht vorgenommen, so dass gerade dies auch dazu führt, die Schritte dieser und der vorherigen Phase als identisch zu betrachten und keine explizite Modellierung der Aufgabenobjekte durchzuführen. Dies kann insbesondere dann sinnvoll sein, wenn eine direkte Abbildung zwischen den Aufgaben- und den Systemobjekten möglich ist. Existierende Aufgabenobjekte werden dann als Systemobjekte abgebildet und analoge Objektmanipulationen erlaubt.

*Evaluierung und Prototyping:* Analog den Testphasen im SE-Entwicklungsprozess sind Evaluierung und Prototyping wichtiger Bestandteil der Anwendungsentwicklung im Bereich HCI.

In der Diskussion, insbesondere bezüglich des HCI-Entwicklungsprozesses, zeigte sich eine unterschiedliche Verwendung der Begriffe und Vorstellungen der durchzuführenden Schritte. So erfolgt beispielsweise in einigen Ansätzen der Teilnehmer keine Modellierung eines Dialogmodells zur expliziten Spezifikation der Mensch-Maschine-Interaktionen, da das diesbezügliche Systemverhalten (teilweise) bereits mit der Beschreibung der Objekte und ihrer Methoden – diese repräsentieren „die Schnittstelle"– erfolgt.

Insgesamt werden von den Teilnehmern die auf der Aufgabenmodellierung aufsetzenden Schritte und damit auch die einzusetzenden Modelle verschiedenartig gehandhabt. So werden zum Teil auf der Basis der Aufgabenbeschreibung die Modelle der Systemsicht entwickelt (z. B. Objekt- und Klassenmodell, Interaktionsdiagramme) oder aber zunächst die Oberfläche und deren Benutzung aus der Sicht der Benutzer entworfen. Der Wechsel zwischen den beiden Gestaltungsräumen, d. h. der benutzungs- und aufgabenzentrierten Sicht und der systemzentrierten Sicht, wird vielfach implizit vorgenommen.

## 6. Abschließende Betrachtung

Die oben dargestellten Diskussionsergebnisse der beiden Untergruppen wurden gegen Ende des Workshops der Gesamtgruppe vorgestellt. In beiden Gruppen zeigte sich, dass viele Fragestellungen, nicht nur hinsichtlich eines gemeinsamen Dialogs zwischen den Bereichen HCI und SE, zu klären sind. So sind einerseits Aufgabenmodellierung und UML in sich unzureichend in ihrer Darstellungskraft, und andererseits bedarf es noch weiterer Möglichkeiten zur Modellierung ihrer Zusammenhänge bzw. Abhängigkeiten. Die genannten Defizite spielen auch bei der Betrachtung eines einheitlichen Entwicklungsprozesses eine Rolle. Dieser wird aber auch durch die Tatsache erschwert, dass im HCI Bereich derzeit noch keine „Vereinheitlichung" der Vorgehensweise sowie der Verwendung von Modellkonzepten und Begriffen existiert. Weiter fortgeschritten ist dies im SE Bereich, aber selbst hier bestehen unterschiedliche Ansätze. Beide Untergruppen des Workshops verdeutlichen, dass bezüglich der Zusammenführung beider Modellierungswelten, aber auch insgesamt im Dialog zwischen HCI und SE, wir alle sicher noch am Anfang der Entwicklung stehen.

**Dr. Birgit Bomsdorf,** Fachbereich Informatik, FernUniversität Hagen. E-Mail: birgit.bomsdorf@fernuni-hagen.de

**Prof. Dr. Gerd Szwillus,** Fachbereich Mathematik/Informatik, Universität Paderborn. E-Mail: szwillus@upb.de, Internet: www.upb.de/cs/ag-szwillus