Am Scheideweg? Plattformenabhängiges Design von Benutzungsscnittstellen

Peter Forbrig
Institut für Informatik
Albert-Einstein-Str. 21
D-18051 Rostock
pforbrig@informatik.uni-rostock.de
http://wwwswt.informatik.
uni-rostock.de

Michael Gellner
Institut für Informatik
Albert-Einstein-Str. 21
D-18051 Rostock
mgellner@informatik.uni-rostock.de
http://wwwswt.informatik.
uni-rostock.de

Abstract
Interaktive Anwendungen müssen heutzutage häufig auf einer Vielzahl unterschiedlicher Geräte bereitgestellt werden. Das Design der Benutzungsschnittstellen soll einerseits möglichst einheitlich und andererseits den Möglichkeiten der jeweiligen Plattform (Desktop, Organizer, Handy) aber auch optimal angepasst sein. Ansätze zur modellbasierten Softwareentwicklung werden im Papier vorgestellt, die die Basis für eine Diskussion über den Prozess des Designs bilden. Wie ist die aktuelle Arbeitsweise eines Designers? Auf welchem Abstraktionsniveau erfolgt das Design? Was sind die Visionen der zukünftigen Arbeit? Wie sind diese Visionen mit einem modellbasierten Ansatz vereinbar?

Keywords
User Interface Design, Plattform übergreifendes Design.

1.0 Einleitung

Am Lehrstuhl für Softwaretechnik der Universität Rostock wird seit mehreren Jahren an einem Konzept der modellbasierten Softwareentwicklung gearbeitet. Ausgehend von Modellen über Aufgaben, Objekte und Nutzer sowie von Umgebungsmodellen werden umfassende Modelle entwickelt, die sich schrittweise dem endgültigen Softwareprodukt nähern.

Die modellbasierte Entwicklung von Softwaresystemen wird immer populärer. Auf der einen Seite gibt es Ansätze, die Aufgabenmodelle als Ausgangspunkt benutzen. Beispiele sind [9, 5, 1]. Auf der anderen Seite gibt es Ansätze, die zunächst auf Objekte fokussieren. Ein Beispiel dafür ist die model-driven architecture von [7]. Die Entwicklung von UML mit seiner Einbeziehung von Anwendungsfällen und Interaktionsdiagrammen unterstützt die Auffassung, dass die Softwareentwicklung der Zukunft die Kombination von Objekt- und Aufgabenmodellen unterstützen sollte. Es gibt jedoch noch ein methodisches Defizit bei der Kombination der beiden Ansätze. Seit mehreren Jahren haben wir sowohl an Methoden, Technologien und Werkzeugen gearbeitet, um einen Entwicklungsprozess zu unterstützen, der aus einer Folge zu entwickelnder Modelle besteht.

Abbildung 2 fasst unsere Sicht auf die zu entwickelnden Modelle zusammen. Sie zeigt auch unsere Auffassung, dass es sehr wichtig ist, dass die Arbeit von Softwareentwicklern und HCI-Experten auf den gleichen Analysemodellen basiert.

Wir glauben an Softwareentwicklung als eine Folge von Transformationen. Solche Transformationen sind durch Softwareentwickler manuell oder besten falls interaktiv bzw. unterstützt durch Werkzeuge durchzuführen. Unsere Arbeit konzentriert sich besonders auf Methoden und Werkzeuge zur Unterstützung der Transformationen durch Muster.

Die werkzeugunterstützte Transformation von Klassendiagrammen durch Entwurfsmuster in Rational Rose ist in T 4 beschrieben. Die Unterstützung der Entwicklung von Aufgabenmodellen wird in [8] demonstriert. Dort wird auch die Animation von abstrakten Prototypen basierend auf einem Aufgabenmodell mit seinen temporalen Relationen und einem Dialoggraphen demonstriert.

Mit unserem Werkzeug DiaTask 8 (siehe Abbildung 2 sind wir in der Lage, einen Dialoggraphen zu entwickeln, der die Navigationsstruktur eines interaktiven Systems repräsentiert. Ein solcher Graph basiert hauptsächlich auf einem Aufgabenmodell. Unser Werkzeug

DiaTask erlaubt die Zuordnung mehrere Dialoggraphen zu einem Aufgabenmodell. Das erlaubt die Betrachtung unterschiedlicher Designentscheidungen für die Navigation.

Abbildung 3 demonstriert ein Beispiel eines Mailsystems mit einfachen Sichten (single views) »Start«, »Window« und »Mail List«, einer Mehrfachsicht (multiple view) »Window Mail« und der Endesicht (end view) »End«.

Zu Mehrfachsichten können verschiedene Instanzen während der Laufzeit erzeugt werden. Für das obige Beispiel bedeutet dies, dass der spätere Nutzer des Systems mehrere Mails gleichzeitig betrachten kann.

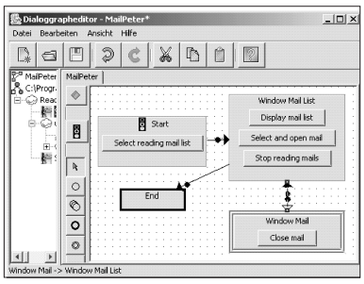

Abbildung 2: Dialoggraph für ein Mailsystem

Abbildung 3 gibt einen Eindruck des abstrakten Prototyps, der aus dem Dialoggraphen von Abbildung REF Abbildung 2 generiert wurde. Dabei ist die Situation festgehalten, die entstanden ist, nachdem die Mails m1 und m2 geöffnet wurden. Aktiv ist die Sicht »Window Mails List«.

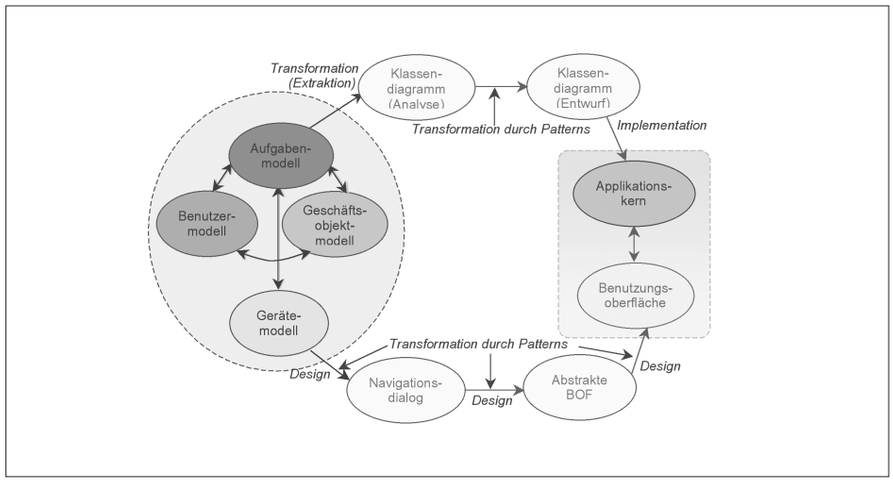

Abbildung1: modellbasierte Softwareentwicklung

2.0 Abstraktversus konkret

Die grundsätzliche Entwicklung der Modelle ist von abstrakt zu konkret. Die Frage ist aber, ob dies auch für alle Teilprozesse gilt. Dies soll am Beispiel der Dialoggraphen diskutiert werden: Erfolgt für jede Dialogsicht zunächst ein abstraktes Design, wie es durch das Projekt CanonSketch [2] vorgeschlagen wird oder entwickelt man zunächst ein konkretes Design, von dem später abstrahiert wird? Mit dieser Frage berührt man eine grundsätzliche Frage über die Arbeitsweise beim Design. Erfolgt das Design zunächst auf einem sehr abstrakten Niveau oder ist Design immer an ganz konkrete Ausprägung der Designelemente gebunden?

Eine Antwort auf diese Fragestellung kann in diesem Beitrag nicht gegeben werden. Er ist als Grundlage für nachfolgende Diskussionen gedacht. In diesem Papier wollen wir an einem konkreten Problem der Gestaltung von Benutzungsoberflächen die angeschnittene Problematik veranschaulichen und zwei mögliche Wege mit ihrer Werkzeugunterstützung vorstellen.

Abbildung 3: Animierter Dialoggraph

Der erste Weg ist die Unterstützung eines abstrakten Designs mit extra dafür geschaffenen abstrakten Elementen, wie dies durch das nachfolgend beschriebene Projekt CanonSketch verfolgt wird.

Eine Alternative ist das ganz konkrete Entwerfen von Benutzungsoberflächen mit einer nachfolgenden Abstraktion. Durch die Abstraktion wird es ermöglicht, alternative Designentscheidungen zu erkennen und vorzuschlagen.
Bevor die Diskussion vertieft wird, seien die Ansätze kurz durch die unterstützenden Werkzeuge charakterisiert.

2.1 CanonSketch

Ziel dieses Projektes ist die Unterstützung der Anforderungsanalyse. Dabei wird ein neuartiger Editor entwickelt, der die Möglichkeit der Entwicklung abstrakter Benutzungsoberflächen eröffnet.
Der Editor sieht aus wie ein herkömmlicher GUI-Editor, präsentiert aber die Elemente der Benutzungsoberfläche nur in einer sehr abstrakten Form.
Es handelt sich um ein Werkzeug zur Manipulation von »kanonischen abstrakten Prototypen«. Die verschiedenen abstrakten Elemente haben eine grafische Repräsentation und können zweidimensional angeordnet werden [6].

2.2 Eclipse-Plugin GUI-Editor

Basierend auf einem existierenden Open-Source-Projekt v4All [10] wurde ein GUI-Editor für die Eclipse-Umgebung entwickelt, der Spezifikationen von Benutzungsoberflächen in der Notation von XUL [3] abspeichert. Bei XUL (XML User Interface Language) handelt es sich um eine Sprache zur Beschreibung von Oberflächen für Softwareanwendungen auf der Basis der XML-Technologie.
Der Editor ermöglicht die Entwicklung neuer Benutzungsoberflächen. Er kann aber auch bestehende XUL-Spezfikationen als graphische Benutzungsoberfläche anzeigen und erlaubt es, Veränderung an diesen vorzunehmen.

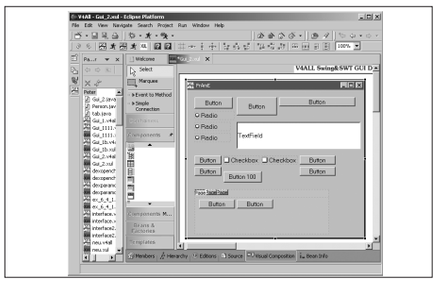

Abbildung 4: GUI-Editor für XUL

Abbildung 4 gibt einen Eindruck vom Aussehen des GUI-Editors in der Eclipse-Umgebung wieder. In seiner ersten Version war der Editor ein eigenständiges System. In der Zwischenzeit erfolgte eine Integration zu unserem System DiaTask, so dass die generierten Prototypen für Dialogsichten manipuliert werden können. Die Verbindung zu den zugehörigen Aufgaben bleibt dabei für alle Oberflächenelemente erhalten. Die Animation der Dialoggraphen erfolgt danach mit der verbesserten Gestaltung der Benutzungsoberflächen.

2.3 Vorgehensweise

Die beiden vorgestellten Vorgehensweisen sind in Abbildung 5 kurz grafisch zusammengefasst. Dabei ist auch der Scheideweg zu erkennen, der in unserer Überschrift angesprochen wird. Erfolgt das Design einer Benutzungsoberfläche sofort auf abstrakte Art und Weise oder wird zunächst immer konkret gestaltet und nachfolgend abstrahiert?

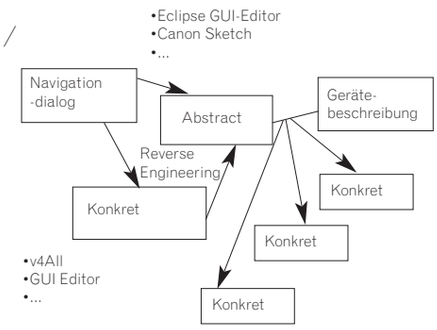

Abbildung 5: Entwicklungsprozess der Benutzungsoberfläche

Die Entscheidung darüber müssen letztendlich Designer treffen. Von den abstrakten Elementen können dann Alternativvorschläge für konkrete Elemente erfolgen. Abbildung 6 gibt ein kleines Beispiel dafür, dass sowohl für die Abstraktion als auch für die Konkretisierung genutzt werden kann.

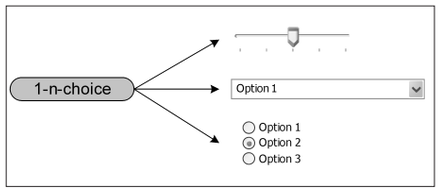

Abbildung 6: Abstrakte Objektrepräsentation für GUI- Elemente

3.0 Zusammenfassung

Der Beitrag diskutiert einen modellbasierten Ansatz zur Entwicklung von Interaktiven Systemen. Er stellt kurz Werkzeuge vor, die einen Entwicklungsprozess basierend auf Transformationen unterstützen. Als Hauptfragestellung wird die Fragestellung »abstrakt versus konkret« bei der Entwicklung von Benutzungsoberflächen herauskristallisiert. Zwei verschiedene Vorgehensweisen werden vorgestellt.
Sind die Sprachelemente von CanonSketch geeignet, um die Arbeit von Designern zu unterstützen? Ist der Weg des abstrakten Designs wirklich sinnvoll? Wird der Weg über ein konkretes Design bevorzugt? Welche Werkzeugunterstützung wünscht sich ein Designer? Diese Fragestellungen sollten durch Diskussionen mit den Betroffenen geklärt werden.

4.0 References

1 Cameleon: http://giove.cnuce.cnr.it/cameleon.html

2 CanonSketch: http://dme2.uma.pt/canonsketch/

3 Deakin, N.: XUL Tutorial. XUL Planet. 2000.

4 Forbrig, P.; Lämmel, R.; Mannhaupt, D.: Patterns-oriented develpment with Rational Rose, Rational Edge, Vol. 1, No. 1, 2001.

5 Paterno, F., Mancini, C., Meniconi, S., ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. Proc. Interact 97, Sydney, Chapman & Hall, 1977, S. 362-369.

6 Constantine, L. Canonical Abstract Prototypes for Abstract Visual and Interaction Design. DSV-IS, Madeira, June 2003, Portugal. OMG, http://www.omg.org/mda/

8 Sinnig, D., Gaffar, A., Reichart, D., Forbrig, P., Seffah, A., Patterns in Model-Based Engineering, Proc. CADUI 2004, Madeira.

9 Wilson, S.; Johnson, P.: Bridging the generation gap: From work tasks to user interface design, In Vanderdonckt, J. (Ed.), Proc. of CADUI 96, Presses Universitaires de Namur, 1966, S. 77-94. v4All: http://v4all.sourceforge.net/index_start.html