

HCI braucht MDA

Argumentationsleitfaden für die modellgetriebene Entwicklung von Benutzeroberflächen.

Reengineering ist teuer und ineffizient

Der Druck einer Umstellung von Client-Server-Software auf netzzentrierte Anwendungen mit Web-Oberflächen wächst. Dabei zeigt sich, wie schnell Entwicklungsplattformen veralten. Der mit einer Technologieumstellung verbundene Reengineering-Aufwand, vor allem bei großen und dialogintensiven Geschäftsanwendungen, ist daher in der Regel hoch. Dies ist auch darauf zurückzuführen, dass der plattformspezifische Code oft eng mit den in der Software eigentlich abgebildeten Geschäftsvorfällen verzahnt ist und nur mit hohem Aufwand - wenn überhaupt - auf die neue Plattform portiert werden kann.

MDA kann helfen, Reengineering zu vermeiden

Durch die zunehmender Komplexität von Softwaresystemen und der gleichzeitigen Forderung nach kürzeren Entwicklungszyklen ist die modellgetriebene Architektur (model driven architecture, MDA) für die Softwareentwicklung von größter Bedeutung. Die rapide technologische Entwicklung im Webumfeld bringt konkurrierende Entwicklungsumgebungen (J2EE, .NET, LAMP) hervor - der Industriestandard der kommenden Jahre ist dabei jedoch nicht absehbar.

Die modellgetriebene Entwicklung beruht auf der Idee, möglichst große Teile der Anwendung in Form von plattformunabhängigen, wieder verwendbaren Modellen zu formulieren und aus diesen Modellen den plattformspezifischen Code zu generieren. Bei einem Wechsel der Entwicklungsplattform können die Modelle in der neuen Entwicklungs- und Laufzeitumgebung weiterverwendet werden, soweit die modellierten Anwendungsteile nicht technologiespezifisch sind. Dazu gehören zum Beispiel Prozesse, Klassen-, Daten- oder Workflowmodelle.

Die Modelle haben dabei einen doppelten Nutzen, da sie sowohl bei fachlich als auch bei technisch bedingten Änderungen der Anwendung stark kostensenkend wirken.

Durch die konsequente Anwendung von Modellen ist der geschäftsspezifische Teil der Anwendung vom technologiespezifischen Teil getrennt. Bei Änderungen an den Geschäftsprozessen werden die Modelle verändert, während das technische Basissystem unverändert bleibt. Bei einem Technologiewechsel wird das technische Basissystem ausgetauscht, während die Modelle stabil bleiben. Bei einer modellgetriebene Anwendung ist die Wartung und Qualitätssicherung also leichter, flexibler und leichter portierbar. Und das bei gleichzeitiger Senkung der Entwicklungskosten.

Standardisierung der Sprache

In den letzten Jahren wurde viel Aufwand in die Entwicklung einer Vielzahl von Notationen zur Darstellung von algorithmischen Problemen und deren modellhafte Lösung investiert. Nach SADT und IDEF entstand Mitte der 90er Jahre mit UML (unified modeling language) erstmals eine international standardisierte Notation. Mit UML lässt sich sowohl ein großer Teil statischer als auch dynamischer Gesichtspunkte eines Softwaresystems modellieren. Ein UML-Modell beschreibt die Objekthierarchie, Beziehungen, Anwendungsfälle und funktionelle Abläufe und geht auf Architektur und Laufzeitumgebung ein.

Zentrales Ergebnis der objektorientierten Analyse (OOA) in UML ist ein Klassenmodell. Das Klassenmodell beschreibt die logische Struktur der Anwendung und gibt die Struktur für das spätere Softwaresystem vor. Die in der Analyse definierten Klassen werden während der weiteren Entwicklung in zunehmender Detaillierung ausgearbeitet und schließlich zu vollständig ausprogrammierten Klassen einer Programmiersprache.

Dialogintensive Geschäftsanwendungen (z.B. Verwaltungssoftware, Kredit-sachbearbeitungssoftware) bestehen in der Regel zu mehr als $\frac{2}{3}$ aus Dialogmasken und Datenmapping zwischen Frontend und Persistenzschicht. Bei solchen Anwendungen entfällt weniger als $\frac{1}{3}$ des Gesamtumfangs auf die Geschäftsprozesslogik. Für diesen Teil bietet UML keine Hilfestellung an, denn Oberflächenabläufe, Anwendungsnavigation und Präsentationsinhalte können in UML nicht modelliert werden.

Warum werden Oberflächen nicht auch modelliert?

Eine Modellierungsmöglichkeit der Präsentationsschicht - d.h. der Mensch-Maschine-Schnittstelle also des „View“ und des „Controllers“ im MVC-Entwurfsmuster (Model-View-Controller) - ist aus Sicht einer dialogintensiven Geschäftsanwendung nicht minder wünschenswert als die Objektmodellierung. Denn die geschäftsspezifischen Dialogabfolgen, Dialoginhalte und Navigationsstrukturen sind der eigentlich beständigste Teil der Anwendung und über Jahrzehnte und Plattformwechsel hinweg wiedererkennbar. Trotz MVC und UML musste die in Ausdehnung und fachlicher Komplexität stets unterschätzte Präsentationsschicht bisher bei jedem Technologiewechsel unter hohem Aufwand neu erdacht und wieder codiert werden, da es dafür lange keine Modellierungsmöglichkeiten gab.

Ziel des schon erwähnten MVC-Entwurfsmusters ist die Aufteilung von interaktiven Applikationen in Teilsysteme mit abgegrenzter Funktionalität und hohem Wiederverwendungsgrad. Das „Model“ ist ein Objekt aus dem Anwendungsbereich, dessen Zustand von den Benutzern beobachtet und evtl. gesteuert werden soll. Der „View“ ist ein sichtbares Element der Bedienungsfläche, an dem man den Zustand des Modells ablesen kann. Der „Controller“ ist ebenfalls ein Element der Bedienungsfläche, wird jedoch von Ereignissen aus Interaktionen der Anwendung mit dem Benutzer beeinflusst bzw. manipuliert (z.B. per Maus oder Tastatur) und steuert den Zustand des Modells.

Nach geeigneten Modellnotationen für HCI wird noch gesucht

UML deckt den „Model“-Aspekt und die technischen Teile des „Controller“-Aspekts des MVC-Entwurfsmusters ab. Der „View“ und die Benutzerschnittstellen-spezifischen Teile des „Controller“ werden hingegen von Spezialmodellierungssprachen abgedeckt oder nach wie vor codiert. Die XML-gestützte Modellierung von interaktiven Oberflächen schließt ein Informationsmodell für Anforderungsmanagement, ein iteratives Modellierungsverfahren, die fachlich-orientierte Erstellung von formalen Spezifikationen und die technisch-orientierte, plattformunabhängige Bereitstellung der Modelle als XML-Baum ein.

Mit Modellierungswerkzeugen, die auf XML und XSLT aufbauen und UML- sowie IDEF-Artefakte mitverwenden, kann die Anwendungsoberfläche für beliebige Zielsysteme einschließlich des Mappings auf die Persistenzschicht nahezu vollständig aus dem Modell der Dialogoberflächen und des Workflows erzeugt werden. Als Nebeneffekt kann man die Zieloberfläche der Anwendung bereits während der Modellierungsphase erproben, somit den Abstimmungsgrad im Entwicklungsteam erhöhen und die tatsächliche Deckung des Produkts mit den Anforderungen sicherstellen.

Der Wert einer weitgehend modellgetriebenen Anwendung liegt in ihrer Robustheit, Flexibilität und Wiederverwendbarkeit und damit in den niedrigen Kosten. Das Modell bleibt stabil, nur die Art der Modellumsetzung in der Zielplattform verändert sich. Je größer der modellierte Anteil der Anwendung ist, umso weniger Code muss bei fachlich oder technisch bedingten Veränderungen der Anwendung neu programmiert werden.

Das Potenzial zur Kosteneinsparung, Aufwandsenkung und Qualitätssteigerung der modellgetriebenen Entwicklung liegt somit auf der Hand.

pch, 08/2004
last revised 01/2006

Paul Chlebek
HCI • GUI • UML • OOD • MDA • XSLT

mailto: pc@benutzeroberflaeche.de
phone: +49 173 106 1830

<http://www.projectpeople.net/chlebek/>
<http://www.benutzeroberflaeche.de>

Links:

UML

<http://www.jeckle.de/unified.htm>