Modellbasierte Oberflächenentwicklung ohne Oberflächen- und Verhaltensmodellierung

Olaf Böde

Freiberuflicher Ingenieur
Marner Straße 43a
22047 Hamburg
olaf.boede@gmx.de

Abstract: Der Beitrag beschreibt einen Ansatz der nicht den Anspruch erhebt eine umfassende modellbasierte Entwicklung von Oberflächen umzusetzen. Er beschreibt einen Weg der den Entwicklungsaufwand durch Kombination von manueller und modellbasierter Entwicklung minimieren soll.

Grundlage des Ansatzes ist die Generierung von Oberflächen aus Definitionen der darzustellenden Datenstrukturen. Es wird gezeigt, dass durch die Einführung von Übergängen zwischen manueller und modellbasierter Programmierung effektive Möglichkeiten existieren, die benötigte Flexibilität der Oberflächengestaltung zu gewährleisten.

1 Einführung

Die modellbasierte Entwicklung hat im Bereich der Serverprogrammierung eine schon beachtliche Akzeptanz gefunden. Hier werden insbesondere Datenstrukturen und Schnittstellensignaturen modelliert. Aus diesen lassen sich mit Hilfe relativ einfacher Abbildungsvorschriften Persistenzcode, Datenklassen, Geschäftslogikinfrastrukturcode etc. generieren.

Für die sehr spezifische Geschäftslogik wird jedoch meist manuell geschriebener Code, der unter Verwendung des Generation Gap Entwurfsmuster [Vli96] eingebunden wird, genutzt.

Im Bereich der Oberflächenprogrammierung wird die modellbasierte Entwicklung derzeit nicht so häufig genutzt. Ansätze die die Struktur der Oberflächenelemente, ihr Verhalten und ihre Zustände detailliert modellieren sind oft komplex. Sie werden wohl deshalb noch nicht sehr häufig angewandt.

Der im Folgenden beschriebene Ansatz verzichtet auf Layout- und Verhaltensmodellierung und zeigt eine Möglichkeit der Kombination von modellbasiert und manuell hergestelltem Oberflächencode.

2 Beschreibung des Ansatzes

In dem Ansatz werden die in der Oberfläche abzubildenden Datenstrukturen modelliert. Dieses Modell wird zur Generierung von Präsentationsmodellen für die Umsetzung des in [Fow04] beschriebenen Presentation Model Patterns genutzt.

Die generierten Präsentationsdatenmodelle verfügen über reichhaltige Informationen für die Oberflächendarstellung wie: Werte, internationalisierte Titel, Wertoptionen, verfügbare Kommandos etc. Zusätzlich beinhalten die hier genutzten Präsentationsmodelle auch Metainformationen über die Struktur der Modellobjekte selbst.

Auf dieser Basis können View Klassen zur Abbildung von Standardlayouts (z.B. Baum, Formular, Menü) realisiert werden. Diese können gegebene Präsentationsmodelle ohne weitere Layout Codierung darstellen.

Welche Art von View zur Darstellung der Geschäftsklassen einer Anwendung genutzt wird ist mit View Mapping Definitionen konfigurierbar.

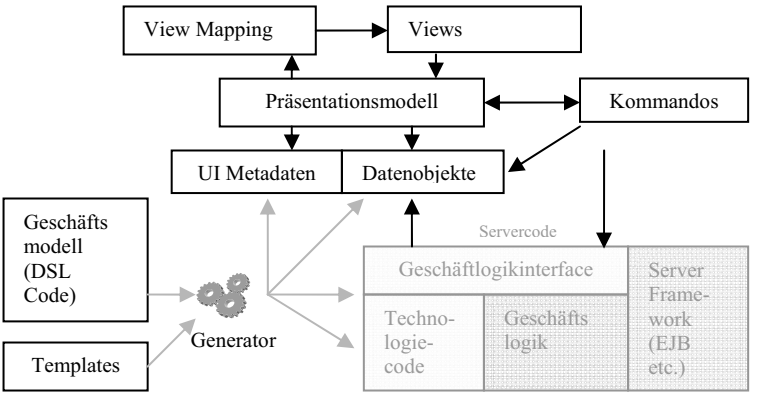

Abbildung 1: Modellbasiert hergestelltes Presentation Model Pattern

3 Erzeugung von uniformen Oberflächen ohne manuelle Codierung

Die von der oben beschriebenen Architektur bereitgestellten Standardviews (z.B. Formulare und Bäume) können genutzt werden um modellierte Geschäftsklassen nahezu ohne manuelle Programmierung zu visualisieren und zu bearbeiten.

Diese Oberflächen können natürlich nur eine uniforme Gestaltung anbieten. Dies kann für manche Stammdatenverwaltung schon ausreichend sein.

Das folgende Bild zeigt eine modellbasiert erzeugte uniforme Nutzerverwaltung die Standardviews (Bäume und Formulare) benutzt:

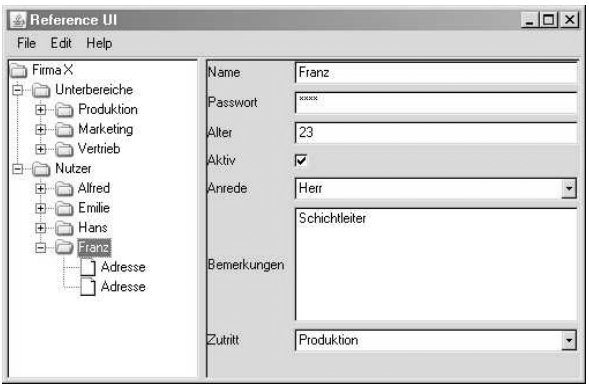

Abbildung 2: Eine ohne manuelle View Codierung erzeugte Oberflächenanwendung

In den meisten Fällen sind solche uniformen Oberflächen jedoch nicht ausreichend. Der folgende Abschnitt beschreibt einen Weg zur weiteren Gestaltung der Oberfläche.

4 Integration von manuellem Code für spezifische Oberflächen

Zur Gestaltung spezifischer Oberflächen können anwendungsspezifische Views manuell (bzw. per GUI Builder) codiert werden. Diese werden mit Hilfe des oben erwähnten View Mappings in die Architektur integriert.

Weitere manuell codierte Funktionalitäten (z.B. anwendungsspezifische Kommandos und Validatoren) werden technisch aus dem Modell durch Verweise auf die Namen der einzubindenden Klassen integriert.

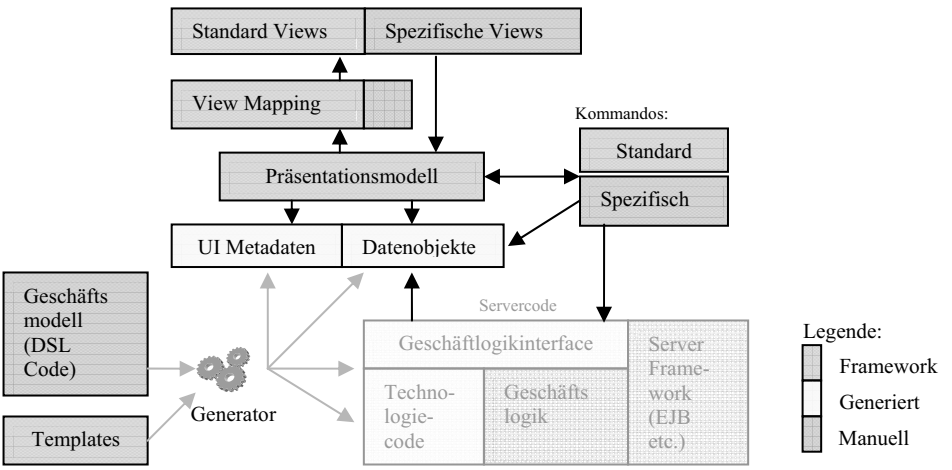

Abbildung 3: Kombination von generiertem, manuellem und Framework- Code

Auf diese Weise wurde die modellbasierte Beispielanwendung um eine manuell geschriebene Darstellung für Nutzerobjekte ergänzt:

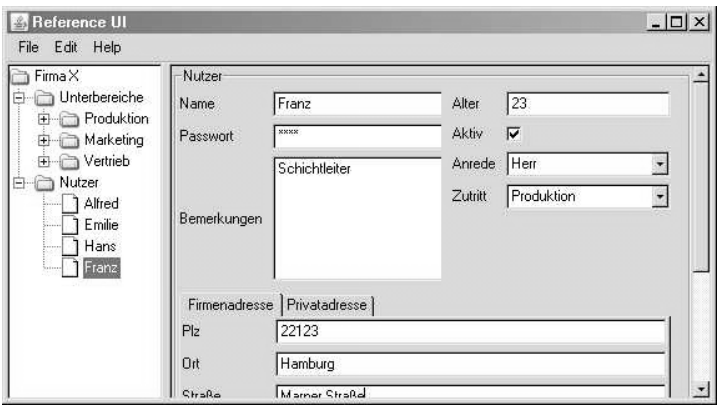

Abbildung 4: Mit manuellem View Code weiterentwickelte Oberfläche

Die manuelle View Codierung kann auch von der Reichhaltigkeit der modellbasiert hergestellten Präsentationsmodelle profitieren. So konnte der dargestellte Nutzerview mit 30 Zeilen Code (netto) realisiert werden.

Es ist also möglich eine effiziente Kombination von Standardlayouts die ohne UI-Codierung bereitstehen mit manuell hergestelltem Oberflächen herzustellen.

5 Anwendbarkeit des Ansatzes für verschiedene Endgeräte

Durch die Nutzung des Presentation Model Entwurfsmusters in Kombination mit Standardviews, ist ein großer Teil des Oberflächencodes vollkommen unabhängig von der konkret genutzten View Technologie (z.B. JSF, SWT, SmartPhone …).

Wenn die üblichen Standardviews für verschiedene Endgerätetechnologien bereitgestellt werden, wird es möglich, eine Anwendung mit relativ geringem Aufwand für verschiedene Endgeräte bereitzustellen. Wenn im günstigsten Fall nur Standardviews genutzt werden ist sogar ein minimaler Aufwand (keine Codierung) erreichbar.

Ein sehr sinnvolles ‚Endgerät' für das Presentation Model ist der automatische Test. Auf diese Weise ist eine sehr effiziente, von der View Technologie unabhängige Testung von großen Teilen des Oberflächencodes möglich.

5 Zusammenfassung

Der beschriebene Ansatz zeigt eine Möglichkeit, die modellbasierte und manuelle Oberflächenentwicklung zu kombinieren. Er wurde im Rahmen eines Industrieprojektes zur Rationalisierung des Oberflächenentwicklungsaufwandes entwickelt.

Gegenwärtig ist damit die Herstellung von Java-Swing Oberflächen möglich. Zur Realisierung der modellbasierten Codegenerierung wird das MDSD Werkzeug openArchitectureWare [oAW] genutzt.

Da sich die Architektur bislang als sehr hilfreich erwiesen hat, ist eine Veröffentlichung als Open Source Projekt unter dem Namen „MyUI“ vorgesehen.

Literaturverzeichnis

[Fow04] Fowler, Martin: Development of Further Patterns of Enterprise Application Architecture; 2004; URL: http://www.martinfowler.com/eaaDev/PresentationModel

[oAW] openArchitectureWare, ein Open Source MDSD Werkzeug; URL: http://www.openarchitectureware.org

[Vli96] Vlissides, John: Generation Gap; C++ Report, November/December 1996; URL: http://www.research.ibm.com/designpatterns/pubs/gg.html