*Beitr. für: Mensch und Computer 2007: Interaktion im Plural*

# Visuelle Spezifikation interaktiver Softwaresysteme

Thomas Memmel, Mathias Heilig, Tobias Schwarz, Harald Reiterer

Arbeitsgruppe Mensch-Computer Interaktion, Universität Konstanz

**Zusammenfassung**

Wir stellen eine Methode zur Modell-getriebenen Erzeugung interaktiver Prototypen und Systemspezifikationen vor. Durch eine visuelle Spezifikationen werden bei gleichzeitiger Steigerung der Softwarequalität die Entwicklungszeiten verkürzt und die Zusammenarbeit mit Zulieferern optimiert.

## 1 Einleitung

In diesem Beitrag stellen wir eine Werkzeugkette zur Modell-getriebenen Erzeugung interaktiver Prototypen vor. Durch die konsequente Verwendung von visuellen Simulationen für funktionale und nicht-funktionale Anforderungen können Prototypen als Vehikel für eine neue Art der Systemspezifikation fungieren. Der Prozess der Anforderungsermittlung soll damit so verändert werden, dass Prototypen als treibende Kraft bis hin zur Systemimplementation beim IT-Zulieferer eingesetzt werden können. Ein Modell-getriebener Ansatz erlaubt dabei die Kreation von Prototypen durch abstrakte und leicht verständliche Methoden. Damit soll der Einsatz von Text-basierten Ausdrucksmitteln reduziert werden.

## 2 Werkzeugkette zur visuellen Spezifikation

Wir trennen den Entwicklungsprozess in die drei Dimensionen Design, Inhalt und Verhalten, wodurch sich einzelne Komponenten für die Gestaltung der Werkzeugkette ergeben. Die Dimension Inhalt beinhaltet die Aufbereitung und Darstellung der Informationsobjekte. Mit der Dimension Design wird die Gestaltung von Blenden und Bildschirmen abgedeckt. Bedienabläufe und Menüstrukturen sind der Dimension Verhalten zuzuordnen. Wie in Abbildung 1 dargestellt, sind mehrere Stakeholder am Entwicklungsprozess beteiligt. Zu den Aufgabenbereichen eines Programmierers gehört die Bereitstellung der benötigten Komponenten

für die Werkzeugkette. Zur Erstellung der domänenspezifischen Modelle (DSM) und der Modellierungssprache (DSL) für die Dimensionen Inhalt und Verhalten ist das Wissen aller Stakeholder über die Anwendungsdomäne nötig. Da sie mit der Domäne hinreichend vertraut sind, müssen die Entwickler keine neue Sprache lernen. Die DSL wird auf Basis einer Metamodellierungssprache in mehreren Zyklen (Luoma 2004). Dazu wird das Meta-Case Tool „MetaEdit" verwendet, mit dem die DSMs visuell erstellt werden können. Für die Erstellung der DSL werden die Konzepte der Domäne durchleuchtet und greifbare Objekte identifiziert und klassifiziert. In unserem Beispiel gibt es eine Reihe von Bildschirmklassen und Eingabeelementen. Für diese Objekte werden eindeutige Bezeichnungen bestimmt, aussagekräftige Symbole erstellt und die Beziehungen der Objekte beschrieben. Durch die Verknüpfung von statischen Bildschirmen und möglicher Interaktion werden die Transitionen in andere Systemzustände beschrieben. Durch das zu Grunde liegende formale Modell können bestimmte Dialogabläufe mit Hilfe von Regeln erlaubt und andere ausgeschlossen werden. So wird vermieden, dass inkonsistente Bedienabläufe modelliert werden.

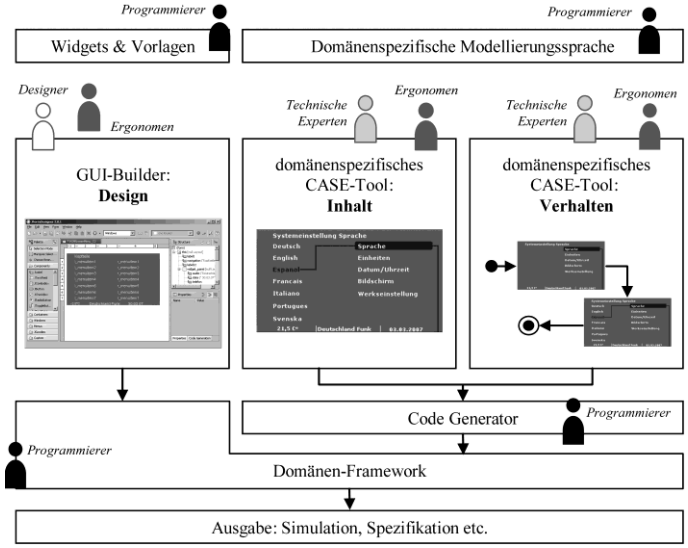


*Abbildung 1: Aufbau der Werkzeugkette; ín Anlehnung an (Memmel et al. 2007)*

Für das Design wird der GUI-Builder JFormDesigner verwendet, wohingegen bei den Dimensionen Inhalt und Verhalten auf die Prinzipien der Modell-getriebenen Softwareentwicklung zurückgegriffen wird. Zum einen existiert derzeit keine Metamodellierungssprache, die genug Möglichkeiten bietet eine ausreichende Modellierungssprache für Benutzungsoberflächen zu erstellen. Zum anderen bieten GUI-Builder mehr Potential für Designer kreative Ideen umzusetzen, als eine formale Modellierungssprache. Der GUI-Builder ermöglicht dem Designer eine Gestaltung ohne Programmierkenntnisse. Selbst definierte Widgets können integriert und ungenutzte Funktionen und Symbole ausgeblendet werden. Der JFormDesigner erzeugt neben Java-Code auch XML zur Beschreibung der Benutzeroberfläche. Die Layouts der Bildschirme werden bei der Modelltransformation in die Simulation integriert.

Im Gegensatz zu State Chart Diagrammen werden in der Werkzeugkette anstelle von abstrakten Symbolen visuelle Objekte aus der Domäne verwendet. Diese Objekte unterstützen die Kommunikation im Entwicklungsprozess (Hamberger et al. 2003). Im Modellierungseditor können Objekte auf der Zeichenfläche platziert und anschließend editiert werden. Die Objekte auf der Zeichenfläche können nun mit den verschiedenen Eingabeelementen verbunden werden. Dadurch wird ein Navigationspfad des Systems dargestellt. Der Benutzer kann die Komplexität der Modelle reduzieren, indem diese in mehrere kleinere Einheiten aufgeteilt werden (vgl. Abbildung 2, 1-2). Zudem können mit dem Modellierungswerkzeug unterschiedliche Abstraktionsebenen implementiert werden. Der Benutzer kann damit steuern, ob er zu einem Bildschirms weitere Details zu Logik und Verhalten betrachten und diese editieren möchte (vgl. Abbildung 2, 2-3). Dadurch wird die Sicht auf die Problemstellung an den jeweiligen Fokus angepasst. Mit der entworfenen DSL kann die Anwendungsdomäne so bestmöglich abgebildet werden. Die Transformation der Modelle in verschiedene Ausgabeformate geschieht durch einen Code Generator mit zwei Aufgaben: (i) die Interpretation der formalen Sprache und (ii) die Konkretisierung der Modelle auf Basis der formalen Sprache durch eine Transformation mittels Transformationsregeln oder einer Makroexpansion mittels Templates. Der Einsatz generativer Methoden erhöht im Gegensatz zu manueller Programmierung die Code-Qualität und reduziert die Fehleranfälligkeit. Für die Transformation der Modelle wird der Generator des MetaCase Tools verwendet. Die Transformationsregeln und Templates werden einmalig erstellt.

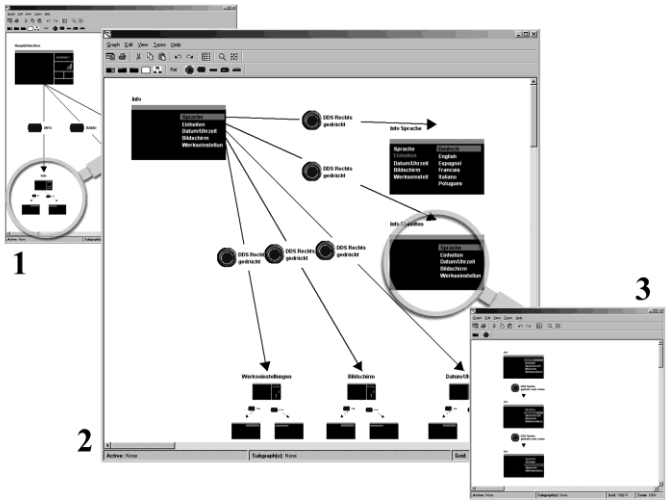


*Abbildung 2: Modelle der domänenspezifischen Sprache in verschiedenen Abstraktionsebenen*

Funktionen, die sich von Modell zu Modell nicht ändern, werden daher in einem Domänenframework bereitgestellt. Eine Konfigurationsdatei verbindet einerseits die Bildschirme mit dem Domänen-Framework, andererseits ordnet sie jedem Bildschirm ein Design zu. Somit wird die Design-Komponente mit den zwei Dimensionen Inhalt und Verhalten verbunden. Schließlich wird auf Knopfdruck eine Simulation des Modells generiert (vgl. Abbildung 3),

die interaktiv erlebt und evaluiert werden kann. Die jeweiligen Zustände, in der sich die Simulation befindet, werden dazu parallel in der Modellierungsumgebung hervorgehoben.

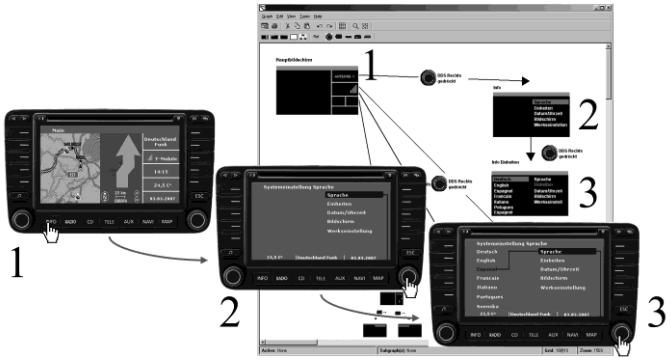


*Abbildung 3: Interaktive Simulation auf einem Entwickler PC*

# 3 Resümee und Ausblick

Die vorgestellte Werkzeugkette bietet einige Mehrwerte, die durch eine Anwendung in der Praxis bestätigt worden sind (Bock 2007). Sie kann in beliebigen Anwendungsdomänen angewandt werden, zu denen eine DSL erzeugt werden kann. Obwohl zunächst ein initialer Aufwand bei der Erzeugung der domänenspezifischen Sprache geleistet werden muss, kann der Entwicklungsprozess jedoch durch den Einsatz Modell-getriebener Methoden verbessert werden. Durch die Aufteilung in unterschiedliche Modellierungsbereiche und Abstraktionsebenen wird die Modellierung des Systems erleichtert. Die Erzeugung interaktiver Simulationen erlaubt frühe Evaluationen und erhöht so die Qualität des UI. Gleichzeitig werden kürzere Entwicklungs- und Reaktionszeiten erreicht. Den Entwicklungspartnern können jeder Zeit formale Spezifikationen zur Verfügung gestellt werden (z.B. in XML).

# 4 Literaturverzeichnis

Bock, C. (2007): Model-Driven HMI Development: Can Meta-CASE Tools do the Job?, In Proceedings of the 40th Annual Hawaii International Conference, Waikoloa, HI, USA, 2007

Hamberger W., Deutler P., Bouaziz T. Audi Multi Media Interface (MMI)--von der Idee zum Produkt, Interdisziplinär-Prozessorientiert-Modellreihenübergreifend; Elektronik im Kraftfahrzeug. Tagung Baden-Baden. 2003, September 25-26. VDI-Bericht 1789.

Luoma, J.; Kelly, S.; Tolvanen, J.-P. (2004): Defining Domain-Specific Modeling Languages: Collected Experiences, Finnland, Jyväskylä 2004

Memmel, T., Bock, C., Reiterer, H. (2007): Model-driven prototyping for corporate software specification, In Proceedings of the Engineering Interactive Systems (EIS) 2007, Salamanca, Spain, 2007