Electronic version of an article published as **Liggesmeyer P, Engels G, Münch J, Dörr J, Riegel N (Hrsg): Software Engineering 2009, 02-06. März 2009 in Kaiserslautern, LNI P-143, pp. 63-74**

# Kontextsensitive Dialogmodellierung

Jürgen Rückert, Barbara Paech
Institut für Informatik, Universität Heidelberg
Im Neuenheimer Feld 326
69120 Heidelberg
{rueckert,paech}@informatik.uni-heidelberg.de

**Abstract:** Die Anpassung von graphischen Benutzungsschnittstellen wird in vielen Anwendungsszenarien benötigt. Anwenderinnen möchten beispielsweise die Oberfläche personalisieren oder diese auf den Kontext des Einsatzes reagieren lassen. Wenn es gelingt die Anpassung zu modellieren, die entweder von Menschen definiert werden kann oder von der Maschine automatisch erlernt werden kann, dann kann das Software Engineering von Benutzungsschnittstellen verkürzt werden. Einerseits entsteht durch die Modellierung Mehraufwand. Andererseits verringert sich der Implementierungsaufwand durch modellgesteuerte, wiederverwertbare Komponenten oder durch Modell-zu-Quellcode-Transformationen.
Der Beitrag fokussiert auf die gleichzeitige Modellierung anpassbarer Strukturen, Eigenschaften und Verhalten von Benutzungsschnittstellen. Der Beitrag stellt ein technologieunabhängiges, kontextsensitives Dialogmodell vor. Das Dialogmodell wurde zur Modellierung eines Web Portals verwendet.

## 1 Einleitung

Die Anpassung von graphischen Benutzungsschnittstellen (BSS) wird in vielen Anwendungsszenarien benötigt. AnwenderInnen möchten beispielsweise die Oberfläche personalisieren oder diese auf den Kontext des Einsatzes reagieren lassen, wie auf den aktuellen Standort oder das verwendete mobile Gerät.
BSS sind bereits aus Aufgaben- [Pat99] und Domänendatenmodellen [Bal94] mit Hilfe von Modelltransformationen ableitbar. In diesen Transformationen müssen viele Entscheidungen und Annahmen festgelegt werden. Wenn die Entscheidungen und Annahmen nicht ausreichend wartbar und änderbar sind, besteht die Gefahr, dass BSS entstehen, die sehr schwer an die Bedürfnisse von AnwenderInnen anpassbar sind.
Um dies zu vermeiden wurden Modelle vorgeschlagen, die die BSS aus technischer Perspektive modellieren und Zwischenergebnisse der Transformationen darstellen, beispielsweise Präsentations- [LV04, Goo01, OAS04] und Dialogmodelle [Træ03, Bas04, W3Cb, Wei94]. Präsentationsmodelle modellieren den graphischen Teil der BSS. Dialogmodelle modellieren das Verhalten der BSS, welches für AnwenderInnen durch die Reaktion auf Ereignisse erfahrbar wird. Aufgrund seiner Komplexität, wird das Dialogmodell häufig bei der Transformation von Aufgaben- und Domänendatenmodellen zu Quellcode vernachlässigt. Es kann aber einen wichtigen Nutzen bringen – es kann entweder die Anleitung für einen Mechanismus sein, der die BSS zur Laufzeit steuert [HHS05, RP08] oder es kann die Grundlage einer Modell-zu-Quellcode-Transformation bilden [BMP04]. Soll

die BSS kontextabhängig sein, dann muss das Dialogmodell kontextsensitiv werden.
Dieser Beitrag stellt ein kontextsensitives Dialogmodell vor. Abschnitt 2 stellt wichtige Anforderungen an kontextsensitive Dialogmodelle auf. Abschnitt 3 stellt bestehende Ansätze zur Dialog- und Kontextmodellierung vor und beurteilt diese auf Erfüllung der in Abschnitt 2 definierten Anforderungen. Abschnitt 4 stellt kurz den bestehenden Dialogmodellierungsansatz *Guilets* [RP08] vor (4.1), der als Basis für das kontextsensitives Dialogmodell dient, erklärt das zur Modellierung der Anpassung verwendbare kontextsensitive *Guilet*-Dialogmodell (4.3) und veranschaulicht in Abschnitt 4.2 dessen Verwendung anhand des realen Web Portals *Heidelberg Mobil* [HMI08]. Abschnitt 5 fasst den Beitrag zusammen und gibt einen Ausblick auf zukünftige Vorhaben.

## 2 Anforderungen an kontextsensitive Dialogmodelle

Kontextabhängige BSS berücksichtigen beim Übergang von einem Zustand A in einen Zustand B den Nutzungskontext der AnwenderInnen. Kontextabhängige BSS können in vielen Anwendungsszenarien benötigt werden [CK00]. Der Nutzungskontext besteht aus einem Satz von Kontextdaten, die die konkrete Situation beschreiben, in der die Zustandsänderung der BSS gewünscht wird. Kategorisierungen für diese Kontextdaten wurden vielfach vorgeschlagen (z.B. in [Bol07, Kap03]). Die folgende Auflistung zeigt exemplarisch Kontextdaten und Beispiele.

- **Rechte oder Rolle der AnwenderInnen**. Abhängig von der Rolle sind unterschiedliche Aktionen möglich.
- **Persönliche Daten der AnwenderInnen**. Abhängig von individuellen Einstellungen (Profil) werden unterschiedliche Sichten einer Seite angezeigt. Sprache, Nationalität, Geschlecht bestimmen das Layout und die Datendarstellung. Abhängig vom Geburtsdatum und Altersgruppe werden interessierende Produkte angezeigt.
- **Eingegebene oder ausgewählte Daten**. Abhängig vom Einkaufsbetrag wird eine Kreditprüfung veranlasst.
- **Hardware**. Abhängig von der verwendeten Hardware werden Seiten in mehrere kleinere Seiten zerlegt und die Abfolge der Seiten angepasst.
- **Umgebung**. Abhängig vom Wetter werden die angezeigten Daten sortiert (bei Regen zuerst die Museen, bei Sonne die Schwimmbäder).
- **Zeitpunkt**. Abhängig von der Tageszeit (Tag oder Nacht) wird das Layout (Farben) angepasst.
- **Ort**. Abhängig vom Standort werden interessante Orte in der Nähe angezeigt.

Üblicherweise dienen Kontextregeln dazu die Kontextdaten, die die konkrete Situation beschreiben, in eine Liste von Anpassungen zu überführen. Die Kontextregel besitzt eine Wenn-Seite (left hand side, LHS) mit bool'schen Bedingungen und eine Dann-Seite (right hand side, RHS) mit Anpassungen für die BSS. Es ist wahrscheinlich, dass die Regeln während der Entwicklungszeit der BSS noch nicht alle bekannt sind oder, dass sich diese während des Betriebs der BSS ändern. Die BSS kann sich auf mehrere Arten und Weisen

anpassen. Im Folgenden sind die möglichen Anpassungen dargestellt. Wir nutzen diese Anpassungen zur Definition von Anforderungen an kontextsensitive Dialogmodelle.

- Sichten können aus anderen Sichten zusammengesetzt werden. Beispiel: Die Startseite eines Web Portals zum eLearning ist zusammengesetzt aus Wochenübersicht-, Newsforum- und Dateiablage-Sichten. 1. Diese Kombination kann allerdings von den AnwenderInnen in ihrem Profil geändert werden. 2. Für ein mobiles Gerät darf die Startseite nur aus einer einzigen Sicht bestehen, da das Display des mobilen Geräts nicht ausreichend groß ist.
  **Anforderung 1.** Das kontextsensitive Dialogmodell soll die Beschreibung der Anpassung der Komposition von Seiten und Sichten erlauben. Ziel dieser Anforderung ist, ein Dialogmodell zu schaffen, welches in der Lage ist deutlich zu machen, bei welchem Ereignis Seiten und Sichten neu zusammengestellt werden.
- Sichten und ihre Bestandteile können ihre Eigenschaften ändern. Beispiel: Neues Layout der Sichten, dies umfasst Farbe und Position.
  **Anforderung 2.** Das kontextsensitive Dialogmodell soll die Beschreibung der Anpassung von Eigenschaften von Sichten, Seiten und Widgets erlauben. Ziel dieser Anforderung ist, ein Dialogmodell zu schaffen, welches in der Lage ist deutlich zu machen, bei welchem Ereignis die Eigenschaften neu gesetzt werden.
- Sichten können mit anderen Daten aus anderen Anwendungsdiensten versehen werden. Beispiel für die Änderung der Datenquellen: Im Kontext *Freizeit* werden die Veranstaltungen von anderen externen Diensten geladen wie im Kontext *Geschäftsbetrieb*. Beispiel für die Änderung der Datenabfragen: Eine Seite zur Buchung eines Mietwagens wird bei angemeldeten AnwenderInnen mit maßgeschneiderten Auswahlen für Fahrzeugtyp, -farbe und Abholort belegt [LKZ06].
  **Anforderung 3.** Das kontextsensitive Dialogmodell soll die Beschreibung der Anpassung der Datenweitergabe an Sichten und Widgets erlauben und damit deren kontextabhängige Datenverwendung. Ziel dieser Anforderung ist, ein Dialogmodell zu schaffen, welches in der Lage ist deutlich zu machen, dass die angesprochenen Dienste zur Datenabfrage und deren Ergebnis kontextabhängig variieren.
- Sichten können andere Anwendungsdienste verwenden, um die Daten der Widgets zu verarbeiten. Beispiel: Die Prüfung der Kreditkarte erfolgt bei unterschiedlichen Banken (Web Services) – je nach Land in dem sich die AnwenderInnen befinden.
  **Anforderung 4.** Das kontextsensitive Dialogmodell soll die Beschreibung der Anpassung der Verwendung von Daten ermöglichen. Die Daten werden verwendet von Widgets. Ziel dieser Anforderung ist, ein Dialogmodell zu schaffen, welches in der Lage ist auszudrücken, dass die angesprochenen semantischen Dienste (zur permanenten Datenänderung) variieren.
- Sichten können unterschiedliche Aktionen für AnwenderInnen anbieten. Beispiel: Menüeinträge können kontextabhängig verfügbar sein.
  **Anforderung 5.** Das kontextsensitive Dialogmodell soll die Beschreibung der Anpassung der zur Verfügung stehenden AnwenderInnen-Aktionen erlauben. Ziel dieser Anforderung ist, ein Dialogmodell zu schaffen, welches in der Lage ist zu beschreiben, welche Ereignisse abhängig vom Kontext ausgelöst werden können.

## 3 Verwandte Arbeiten

Man unterscheidet Ansätze zur Modellierung des Kontextes (z.B. UsiXML [LV04]), zur Modellierung der Anpassung der BSS (z.B. Catwalk [LKZ06], UWE [KKK07] und aufgabenbasierte Ansätze wie Contextual CTT [BC04] und [WFTS07]) und zur Modellierung und Auswertung von Kontextregeln (z.B. [Che04]). Letztere sind für uns nicht von Interesse, da die Auswertung von Regeln (Context Reasoning) nicht Bestandteil der Dialogmodellierung ist.
Stellvertretend wird der populäre Ansatz UsiXML und der auf semantischen Technologien basierende Ansatz Catwalk in den Abschnitten 3.1 und 3.2 vorgestellt. Die Ansätze erfüllen die in Abschnitt 2 aufgestellten Anforderungen in unterschiedlich hohem Maße. Dies liegt daran, dass die Ansätze nicht primär die Dialogmodellierung fokussieren. Tabelle 1 zeigt die Erfüllung der Anforderungen. Die Begründung der Beurteilung wird in nachfolgenden Abschnitten 3.1 und 3.2 gegeben.

| Anforderung | Kontextabhängige... | UsiXML | Catwalk |
|---|---|---|---|
| 1 | ...Komposition der Präsentation | modellierbar | modellierbar |
| 2 | ...Eigenschaften der Präsentation | modellierbar | modellierbar |
| 3 | ...Datenanpassung | - | modellierbar |
| 4 | ...Datenverwendung | - | - |
| 5 | ...Aktionen | modellierbar | - |

Tabelle 1: Erfüllung der Anforderungen durch bestehende Ansätze für kontextabhängige BSS

### 3.1 UsiXML

Der zur Entwicklung von modellbasierten BSS populäre Ansatz UsiXML [LV04] bietet neben einem Präsentationsmodell ein Kontextmodell. Ein explizites Dialogmodell ist nicht Bestandteil von UsiXML. Das Kontextmodell von UsiXml modelliert die Umgebungen, AnwenderInnen und Plattformen. Das Plattformmodell basiert auf dem *User Agent Profile* [WAP01], welches eine Verfeinerung des *Composite Capabilities / Preferences Profile* [W3Ca] des W3C darstellt. Entsprechend ist das Plattformmodell sehr technologisch orientiert. Das Nutzermodell besteht aus einer einzigen Klasse *userStereotype* mit fünf Eigenschaften. Das Umgebungsmodell besteht aus einer einzigen Klasse *environment* mit drei Eigenschaften. Zur Modellierung der Anpassung bietet UsiXML keine speziellen Möglichkeiten. Es besteht nur die allgemeine Möglichkeit das Präsentationsmodell durch XML-Transformation unter Verwendung des Kontextmodells in ein neues UsiXML-Präsentationsmodell zu transformieren [Sta05]. Die Entwicklung und Wartung der XML-Transformation ist sehr aufwendig, wenig transparent und erlaubt keine ereignisbasierte Anpassung der BSS und, durch Fehlen des Dialogmodells, keine Modellierung der Verhaltensanpassung. UsiXML erlaubt durch die XML-Transformation unter Verwendung des Werkzeugs *TransformiXML* [Sta05] (unter Verwendung der *Attributed Graph Grammar*) die Anpassung der Komposition (Anforderung 1) und der Eigenschaften (Anforderung 2) der Präsentation. Da Aktionen von AnwenderInnen durch graphische Komponenten ausgelöst werden, kann durch Ein- und Ausblenden die Anforderung 5 rudimentär erfüllt werden.

### 3.2 Catwalk

Catwalk [LKZ06] ist ein komponentenorientiertes Rahmenwerk für Web-Anwendungen. Die BSS wird durch verschiedene Ontologien beschrieben: eine *domain ontology*, mehrere *context ontologies* und ein *context relations model*, welches die beiden vorherigen Ontologien in Beziehung setzt. Das *navigation*, *view* und *presentation model* enthalten Regeln zur Anpassung des Systems auf Basis der Ontologien. Catwalk unterscheidet die *selection*, *parameterization* und *presentation* der Präsentation, was der Anpassung der Komposition (Anforderung 1), der Eigenschaften (Anforderung 2) und Datenpräsentation (Anforderung 3) entspricht. Catwalk erlaubt die Anpassung der Komposition und der Eigenschaften der Präsentation, sowie der Datenabfragen – berücksichtigt aber nicht die Anpassung der Datenauswertung und der zur Verfügung stehenden Aktionen.

Die derzeitigen Ansätze zur Modellierung von BSS, von denen hier nur eine sehr kleine Anzahl von repräsentativen Vertretern vorgestellt werden konnte (mächtigere Ansätze sind uns nicht bekannt) bieten zwar viele Möglichkeiten zur Beschreibung des Kontexts, zur Modellierung der Präsentation oder bieten BSS-Referenz-Architekturen an, die klar separierte Komponenten (*Context Reasoner*) zur Ermittlung der notwendigen Anpassungen enthalten. Die Beschreibung der Änderung der BSS in einem Dialogmodell wurde bisher allerdings noch nicht vollständig addressiert. Dies ermöglicht die Dialogmodellierung mit *Guilets*, die im nächsten Kapitel erläutert wird.

## 4 Guilets

### 4.1 Der Lösungsansatz

Der Lösungsansatz *Guilets* [RP08] stellt zum Erstellen der anwendungsspezifischen *Guilet*-Dialogmodelle in XML während des Entwurfs bereit: **1.** Das *Guilet*-Metamodell in XML Schema. **2.** Den *Guilet*-Editor, als Eclipse-Feature auf Basis des *Eclipse Graphical Modeling Framework* in Java (und auf Basis des *Guilet*-Metamodells).
Der Lösungsansatz *Guilets* stellt (auf Basis des *Guilet*-Metamodells) zur schnellen Entwicklung eines BSS-Prototyps und zur Simulation bereit: **3.** Die *Guilet*-Referenzarchitektur für BSS. **4.** Ein Werkzeug in Java zur Generierung von Quellcode von Verhaltenskomponenten (so genannte Exekutoren, eine Default-Implementierung) und deren Schnittstellen. **5.** Ein Werkzeug in Java zur Generierung von Quellcode der Präsentationen (eine Default-Implementierung) und deren Schnittstellen. **6.** Ein *Guilet*-Dialogkern in Java zur Koordination der BSS auf Basis aller oben genannten Einheiten.
Das Charakteristische am *Guilet*-Dialogmodell ist, und das unterscheidet es von anderen Ansätzen zur Dialogmodellierung [Træ03, Bas04, W3Cb, Wei94], die integrierte Modellierung des hierarchischen Aufbaus von Sichten, die Abstraktion von graphischen Komponenten (*Pisa Interactor Abstraction* [Mar97]), die Modellierung von Sichtenabfolgen (UML Statecharts [Har87]) und die Verwendung von Modellelementen für Anwendungs- und BSS-Dienste innerhalb von Sichten.
Im Rahmen, des im Abschnitt 4.2 vorgestellten Beispiels, wurde ein Dialogmodell erstellt (1. und 2.), aus diesem Modell Default-Verhaltenskomponenten und Default-Präsentationen generiert (4. und 5.), Präsentationen und kontextabhängige Verhaltenskomponen-

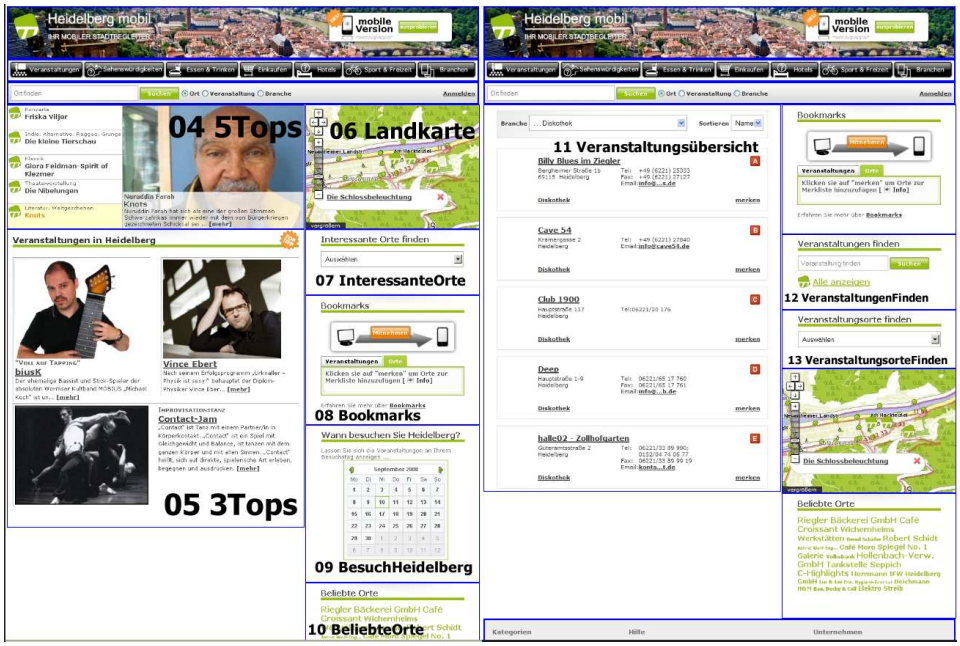


Abbildung 1: Ausschnitt des Web Portals *Heidelberg Mobil* (dicke Linien zeigen die Abgrenzung der Sichten)

ten an die konkreten Bedürfnisse angepasst (implementiert) und alle Komponenten durch den Dialogkern zur Laufzeit koordiniert (6.).

### 4.2 Das Anwendungsbeispiel

Zur Veranschaulichung einer kontextabhängigen, graphischen BSS verwenden wir das Web Portal *Heidelberg Mobil* [HMI08]. Abbildung 1 zeigt die Webseite des Portals vor (links) und nach (rechts) der Anpassung. Die Webseite ist eine Komposition einzelner Sichten, die eindeutig identifizierbar sind (`06`) und einen Namen haben (`Landkarte`). Die mobile Webseite [HMI08] von *Heidelberg Mobil* wird ebenfalls aus Sichten kombiniert. Durch die größenbeschränkten Geräte (PDA, iPhone, Handy) können nicht alle der in Abbildung 1 dargestellten Sichten verwendet werden. Zudem entfallen die Sichten zur Suche und Auswahl der aktuellen Lokation durch die automatische Ermittlung des Standorts, welche bei mobilen Geräten ohne Zutun der AnwenderInnen geschieht. Der in diesem Abschnitt 4 vorgestellte Lösungsansatz kann für beliebige Kontextänderungen angewandt werden (nicht nur für Lokationsänderungen) und ist damit für beliebige mobile und nicht-mobile Web Portale verwendbar. Die Anpassung erfolgt unter Berücksichtigung der Kontextregeln aus Tabelle 2. Die Kontextregeln werten Stamm- und Bewegungsdaten der AnwenderInnen aus. Es wird angenommen, dass die AnwenderInnen dem System bekannt sind (durch Einloggen). Bewegungsdaten sind beispielsweise das von Anwende-

| Bedingung (LHS) | | Aktion (RHS) | | |
|---|---|---|---|---|
| **Ausdruck** | **Kategorie** | **ID** | **Ausdruck** | **Anf.** |
| Nutzer.Ereignis = SucheVeranstaltung | Ereignis | 1 | Entferne Sicht 5Tops | 1 |
| | | 2 | Entferne Sicht 3Tops | 1 |
| | | 3 | Zeige Sicht Veranstaltungübersicht | 1 |
| | | 4 | Entferne Sicht Interessante Orte | 1 |
| | | 5 | Zeige Sicht VeranstaltungenFinden | 1 |
| | | 6 | Zeige Sicht VeranstaltungsorteFinden | 1 |
| Nutzer.Name = Mayer | Stammdaten | 7 | Lade die persönlichen Bookmarks in Sicht Bookmarks | 3 |
| | | 8 | Setze Hintergrundfarbe weiß im Fenster | 2 |
| Nutzer.Typ = Heidelberger | Stammdaten | 9 | Verschiebe die Sicht Landkarte nach unten | 2 |
| | | 10 | Verschiebe die Sicht Bookmarks nach oben | 2 |
| | | 11 | Entferne die Sicht BesuchHeidelberg | 1 |
| Lokation = Unbestimmt | Lokation | 12 | Positioniere die Sicht Landkarte auf den Bismarckplatz | 3 |
| Uhrzeit > 22:00 und < 23:00 | Uhrzeit | 13 | Zeige in der Sicht Veranstaltungsorte zuerst die Diskotheken an | 3 |
| Gerät = Desktop | Plattform | 14 | Zeige in Sicht Bookmarks die Aktion Mitnehmen an | 5 |

Tabelle 2: Kontextregeln am Beispiel eines kontextabhängigen Web Portals (Anf. referenziert die Anforderungen aus Abschnitt 2)

rInnen ausgelöste Ereignis (Zeile 1), die zur Laufzeit ermittelten Werte Lokation (Zeile 4) und Uhrzeit (Zeile 5) und das Gerät, an dessen Hardware die BSS angepasst werden soll (Zeile 6). Zu den Stammdaten zählen die Personalisierungseinstellungen, die am Namen (im Sinne ID) festgemacht werden (Zeile 2), verschiedene Eigenschaften (ob ortskundig oder nicht, Zeile 3). Die Spalte *Anforderung* zeigt, dass die Sichten des Portals häufig neu komponiert werden (Anforderung 1), dass Eigenschaften von Sichten und Widgets geändert werden (Anforderung 2), dass die Widgets mit kontextabhängigen Daten gefüllt werden (Anforderung 3) und dass AnwenderInnen kontextabhängig Ereignisse auslösen können (Anforderung 5). Das Portal besitzt allerdings keine kontextabhängige Auswahl von Anwendungsdiensten (Anforderung 4). Bei der Dialogmodellierung des Portals gehen wir davon aus, dass die BSS aus Seiten besteht, die in einem Rahmen angezeigt werden, beispielsweise eine Webseite in einem Browser oder eine Seite zum Einloggen in einem neu erscheinenden modalen Fenster. Die Seiten sind aus geschachtelten Sichten aufgebaut. Die Sichten beinhalten elementare, graphische Komponenten (Widgets), beispielsweise Textfelder, Tabellen oder Landkarten. In den Seiten, Sichten und Widgets können AnwenderInnen Ereignisse auslösen. Wie in Abschnitt 2 gezeigt, kann der Kontext die Präsentation und das Verhalten der BSS umfangreich ändern – prinzipiell bei jedem von AnwenderInnen ausgelösten Ereignis. Dialogmodelle beschreiben das Verhalten bei Eintreten eines Ereignisses, wobei üblicherweise das Ereignis einer Seite, Sicht oder Widget zugeordnet wird. Das Dialogmodell referenziert Elemente des Präsentationsmodells. Normalerweise folgt die Reaktion auf Ereignisse einer festgelegten Beschreibung (man denke an Statecharts oder Aktivitätsdiagramme). Die von uns in Abschnitt 2 aufgestellten Anforderungen sollen Dialogmodelle ermöglichen, die in der Lage sind auszudrücken, wo es in dieser festgelegten Beschreibung zu kontextmotivierten Anpassungen kommen kann.

### 4.3 Das Modell zur kontextsensitiven Dialogmodellierung

Zur Umsetzung der Anforderungen aus Abschnitt 2, wurde das *Guilet*-Dialogmodell aus [RP08] um drei Konzepte erweitert:

- **Konzept 1.** Zwei Untertypen von Exekutoren wurden eingeführt, die die spezielle Semantik *KomponiereSichten* und *SetzeEigenschaften* besitzen.
- **Konzept 2.** Exekutoren, die sich bisher mit Anwendungsdiensten verbunden haben, können nun zusätzlich mit einem *Context Reasoner* verbunden werden, der über die auszulösenden Aktionen entscheidet.
- **Konzept 3.** InEvents, die Ereignissen entsprechen, können nun zusätzlich mit einem *Context Reasoner* verbunden werden, der über deren Auslösungsmöglichkeit entscheidet.

Das *Guilet*-Dialogmodell [RP08] basiert auf der (nicht neuen) Idee, dass atomare Sichten gebildet werden, die danach zu komplexen Sichten zusammengesetzt werden. Neu ist die Art wie das Verhalten der atomaren und der komplexen Sichten modelliert wird. Die in [RP08] vorgestellte Version des *Guilet*-Dialogmodells unterstützt die Berücksichtigung des Kontexts noch nicht, die Lücke soll mit diesem Beitrag geschlossen werden. Im nächsten Absatz wird das *Guilet*-Dialogmodell aus [RP08] vorgestellt, bevor im übernächsten Absatz die kontextsensitiven Erweiterungen genannt werden.
*Guilets* sind Modelle von graphischen Komponenten wie Fenster, darin enthaltenen Sichten und Widgets. Atomare *Guilets* (**BlackBoxGuilets**) modellieren graphische Komponenten deren Verhalten nicht von Interesse ist. Dies ist der Fall bei Verwendung von graphischen Komponenten aus Bibliotheken, die ein zu kompliziertes Verhalten besitzen, um modelliert zu werden. Komplexe *Guilets* (**WhiteBoxGuilets**) modellieren graphische Komponenten, deren Verhalten von Interesse ist. Komplexe *Guilets* bestehen aus komplexen *Guilets* oder aus atomaren *Guilets*. Durch Auswertung des Kontexts zur Laufzeit können die beinhalteten Sichten variieren (dynamische Komposition). Eine In-Schnittstelle (**InEvent**) wird für jede Nutzeraktion modelliert, die die Sicht anbietet. Über die Schnittstelle können Daten übergeben werden, müssen aber nicht. Eine Out-Schnittstelle (**OutEvent**) wird für jede Nachricht modelliert, die die Sicht senden kann und die andere Sichten empfangen und nutzen können (über das Modellierungselement Transition). **Exekutoren** sind Modelle für Dienst-Komponenten und repräsentieren Anwendungsdienste oder BSS-Dienste. Ein Exekutor bietet eine Schnittstelle, die die Ausführung des Exekutors erlaubt. Während der Ausführung können Exekutoren festgelegte globale Daten lesen und festgelegte globale Daten (dieselben oder andere) schreiben. Ein Exekutor kann mehrere Schnittstellen von Sichten nutzen. Durch Auswertung des Kontexts können die tatsächlich genutzten Schnittstellen variieren bzw. erst zur Laufzeit bekannt werden. Alle Modellierungselemente haben eine ID, einen Namen, eine Beschreibung und können beliebig definierbare Eigenschaften (Schlüssel-Wert-Paare) enthalten. Hiermit können beispielsweise Layout-Vorgaben definiert werden (beispielsweise Position, Anzeige in Fenster).
Wir schlagen die folgenden Erweiterungen des *Guilet*-Dialogmodells vor:

- **Realisierungen von Konzept 1.** Die Anpassung der Komposition (Anforderung 1 aus Abschnitt 2) von Seiten und Sichten wird durch einen speziellen BSS-Dienst

beschrieben, den *ComposeGuiletsExecutor*. Die Anpassung der Eigenschaften (Anf. 2 aus Abschnitt 2) von Seiten, Sichten und Widgets wird durch einen speziellen BSS-Dienst beschrieben, den *SetPropertyExecutor*.

- **Realisierungen von Konzept 2.** Die kontextabhängige Weitergabe von Daten aus Diensten (Exekutoren) an Seiten, Sichten und Widgets (*Guilets*) (Anf. 3 aus Abschnitt 2) wird durch einen *Context Reasoner* entschieden, falls der Exekutor mit Hilfe des *ContextualFlag* markiert wurde. Die kontextabhängige Weitergabe von Daten aus Widgets (*Guilets*) an Dienste (Exekutoren) (Anf. 4 aus Abschnitt 2) wird durch einen *Context Reasoner* entschieden, falls der Exekutor mit Hilfe des *ContextualFlag* markiert wurde.
- **Realisierung von Konzept 3.** Die Verfügbarkeit (Anf. 5 aus Abschnitt 2) von auslösbaren Ereignissen (InEvents) wird durch einen *Context Reasoner* entschieden, falls der InEvent mit Hilfe des *ContextualFlag* markiert wurde.

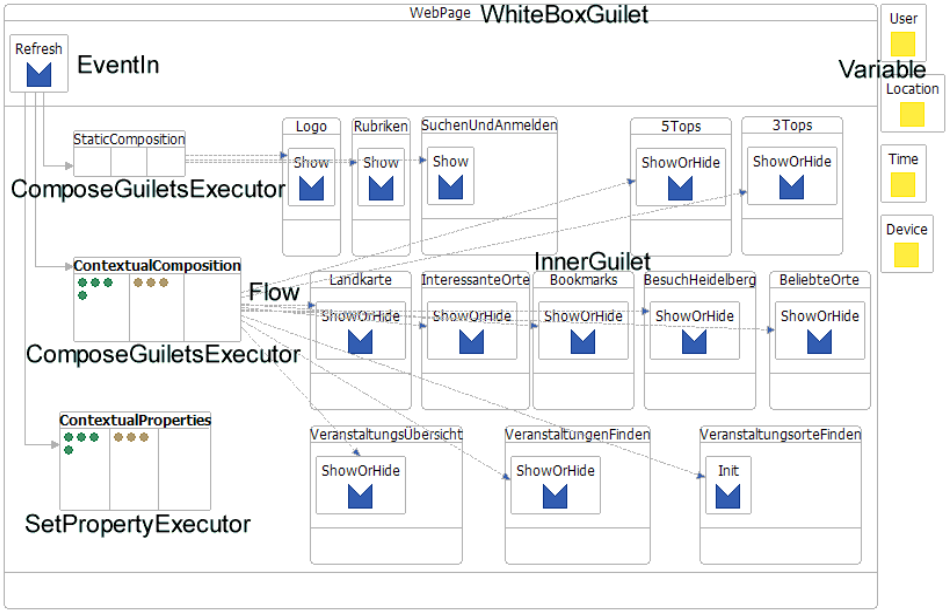


Abbildung 2: Ausschnitt aus dem *Guilet*-Dialogmodell zur Modellierung der Seite (die zugehörigen Typen der Modellelemente sind fettgedruckt)

Abbildung 2 veranschaulicht anhand das *Guilet*-Dialogmodells der Webseite, wie die Anpassung (Tabelle 2) der Sichtenkomposition (1-6, 11) und der Sichteneigenschaften (8-10) modelliert werden. Die Anpassung des Sichtenverhaltens (7, 12-14) erfolgt durch kontextabhängige Verhaltensexekutoren in den einzelnen Sichten und ist hier nicht dargestellt, da im kontextfreien *Guilet*-Dialogmodell nur auf kontextabhängige Exekutoren umgestellt werden muss.
Aus dem Dialogmodell kann folgende Abfolge, zur Durchführung der Anpassung, abge-

lesen werden: 1. Die Webseite empfängt das Ereignis *Refresh*. 2.1 Der Exekutor *StaticComposition* wird aufgerufen, um die 3 immer vorhandenen Sichten anzuzeigen (statische Komposition). 2.2 Der Exekutor *ContextualComposition* (vom neuen, speziellen Typ *ComposeGuiletsExecutor*) wird aufgerufen, um die durch den Kontext bestimmten Sichten zu ermitteln (kontextsensitive Komposition). 2.3 Der Exekutor *ContextualProperties* (vom neuen, speziellen Typ *SetPropertyExecutor*) wird aufgerufen, um die Eigenschaften (Farben und Positionen) der Webseite zu setzen. Bei beiden Exekutoren ist das neue *ContextualFlag* gesetzt, damit diese den *Context Reasoner* ansprechen, der die anzuzeigenden Sichten (IDs) und die Eigenschaften (Schlüssel-Wert-Paare) der Sicht ermittelt. Dem *Context Reasoner* stehen als Datenbasis der Kontextregeln die 4 globalen Variablen *User*, *Location*, *Time* und *Device* zur Verfügung (Punkte der linken Sektion). 3. Die einzelnen Sichten erhalten ein Ereignis, welches nach dem Dialogmodell der einzelnen Sichten abgearbeitet wird (letzteres Modell ist nicht dargestellt).
Bei der modellbasierten Entwicklung des Portals *Heidelberg Mobil* haben wir positive und negative Erfahrungen gesammelt. Dies ist einerseits auf die generelle Verwendung von Modellen zurückzuführen und andererseits auf die spezielle Verwendung eines kontextsensititen Dialogmodells. (+) Die Erweiterung des Dialogmodells auf Kontextsensitivität war einfach durch die bestehende Abstraktion der Dienste (Executors) und Ereignisauslöser (InEvents). (+) Das Dialogmodell dokumentiert diejenigen Sichten, die kontextsensitiv sind, in transparenter Art und Weise. Eine Suche im Quellcode nach den von Anpassung betroffenen Sichten und den zur Anpassung ausgewerteten Daten entfällt. (-+) Einerseits müssen die Kontextregeln des *Context Reasoner* mit dem Dialogmodell synchronisiert werden, weil die Kontextregeln Elemente des Dialogmodells referenzieren. Andererseits ist die Synchronisierung einfach, weil man an zentraler Stelle (im Dialogmodell) nach den Elementen suchen kann. (+) Weder Quellcode noch Dialogmodell mussten zur Realisierung von prototypischen Anpassungsszenarien verändert werden. Die Änderung der Kontextregeln (in einem *Context Reasoner*) war ausreichend. (-) Die erstmalige Erstellung des Modells und die Entwicklung der Transformation von Modell in Quellcode hat Aufwand verursacht. (+) Das Dialogmodell umfasst nun gleichzeitig Sichten der mobilen und der nicht-mobilen Version der Webseiten. Eine Erweiterung um neue Sichten kann automatisch beiden Versionen zu Gute kommen. Ebenso verhält es sich bei dem Umstieg auf ein alternatives Web-Rahmenwerk.

## 5 Zusammenfassung und Ausblick

In diesem Beitrag haben wir Anforderungen an kontextsensitive Dialogmodelle aufgestellt (Abschnitt 2). Wir haben zwei bestehende Ansätze zur kontextabhängigen Anpassung von BSS vorgestellt (Abschnitt 3) und unsere Lösung zur kontextsensitiven Dialogmodellierung präsentiert (Abschnitte 4.1 und 4.3). Unsere Lösung haben wir anhand des realen Web Portals *Heidelberg Mobil* (Abschnitte 4.2 und 4.3) veranschaulicht.
Das *Guilet*-Dialogmodell ermöglicht die Modellierung der Anpassung der Komposition, der Eigenschaften und des Verhaltens von Sichten, wie sie für kontextabhängige BSS notwendig sind. Zur Anpassung der Komposition von Sichten haben wir einen speziellen BSS-Dienst definiert. Zur Anpassung der Eigenschaften von Seiten, Sichten und graphischen Komponenten haben wir ebenfalls einen speziellen BSS-Dienst definiert. Die

Modellierungselemente für BSS-Dienste und Anwendungsdienste können *kontextsensitiv* sein. Zur Laufzeit verbinden sich die Dienst-Komponenten, die auf Basis dieser kontextsensitiven Modellierungselemente implementiert werden, zu externen *Context Reasoner* Komponenten, die die gewünschten auszulösenden Ereignisse der BSS ermitteln. Das *Guilet*-Metamodell wurde erfolgreich verwendet zur Modellierung des Web Portals *Heidelberg Mobil* im Rahmen eines Reverse Engineering. Wir haben, wie uns die Anwendung auf das Web Portal *Heidelberg Mobil* gezeigt hat, ein attraktives Dialogmodell geschaffen, welches kontextabhängige BSS modellieren kann und die Lücke zwischen abstrakten Modellen (wie Aufgaben- und Domänendatenmodellen) und Quellcode schließen kann, indem es Präsentationsmodelle um Verhaltensbeschreibungen ergänzt und einen Entwurf ermöglicht, der mit der Implementierung synchron ist.
Wir planen die weitere Evaluation des *Guilet*-Metamodells, um die Verwendbarkeit für unterschiedliche Implementierungstechnologien und weitere kontextsensitive Anwendungsszenarien zu überprüfen. Erstens soll unsere Entwurfsentscheidung gefestigt werden: *Es ist für die Wartbarkeit des Dialogmodells vorteilhafter, wenn die Kontextregeln nicht Bestandteil des Dialogmodells sind.* Zweitens soll die Eignung des Dialogmodells für die *kontextabhängige Auswahl von Anwendungsdiensten* (Anforderung 4 aus Abschnitt 2) in einem dienstorientierten Umfeld gezeigt werden.
Wir danken Matthias Jöst von *Heidelberg Mobil International* für die freundliche Zusammenarbeit und Unterstützung der Arbeit.

## Literatur

[Bal94] H. Balzert. Das JANUS-System. *Inform., Forsch. Entwickl.*, 9(1):22–35, 1994.

[Bas04] R. Bastide. A model-based approach for real-time embedded multimodal systems in military aircrafts. In *ICMI*, Seiten 243–250, 2004.

[BC04] J. van den Bergh und K. Coninx. Contextual ConcurTaskTrees: Integrating dynamic contexts in task based design. *Pervasive Computing and Communications Workshops, IEEE International Conference on*, 0:13, 2004.

[BMP04] S. Berti, G. Mori und F. Paternò. A transformation-based environment for designing multi-device interactive applications. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interfaces*, Seiten 352–353. ACM, 2004.

[Bol07] C. Bolchini. A data-oriented survey of context models. *SIGMOD*, 36(4):19–26, 2007.

[Che04] H. Chen. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *Internation Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, 2004.

[CK00] G. Chen und D. Kotz. A Survey of Context-Aware Mobile Computing Research. Bericht, 2000.

[Goo01] B. Goodger. XML User Interface Language (XUL) Specification 1.0 and Documentation., 2001. http://www.mozilla.org/projects/xul/.

[Har87] D. Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Program.*, 8(3):231–274, 1987.

[HHS05] M. Haft, B. Humm und J. Siedersleben. The Architect's Dilemma - Will Reference Architectures Help?. In *QoSA/SOQUA*, Jgg. 3712 of *LNCS*, Seiten 106–122. Springer, 2005.

[HMI08] Heidelberg Mobil International. Das Web Portal Heidelberg Mobil, Stand vom 25. September 2008. http://www.heidelberg-mobil.de/ http://www.heidelberg-mobil.de/mobile.

[Kap03] G. Kappel. Customisation for ubiquitous web applications a comparison of approaches. *Int. J. Web Eng. Technol.*, 1(1):79–111, 2003.

[KKK07] A. Kraus, A. Knapp und N. Koch. Model-Driven Generation of Web Applications in UWE. In *3rd International Workshop on Model-Driven Web Engineering (MDWE)*, Como, Italy, July 2007.

[LKZ06] S. Lohmann, J. Kaltz und J. Ziegler. Model-Driven Dynamic Generation of Context-Adaptive Web User Interfaces. In *MoDELS Workshops*, Seiten 116–125, 2006.

[LV04] Q. Limbourg und J. Vanderdonckt. USIXML: A User Interface Description Language Supporting Multiple Levels of Independence. In *ICWE Workshops*, Seiten 325–338, 2004.

[Mar97] P. Markopoulos. A compositional model for the formal specifiation of user interface software. PhD thesis at Department of Computer Science, Queen Mary and Westfield College, University of Londo, 1997.

[OAS04] OASIS. User Interface Markup Language Specification (UIML) 3.1 (Working Draft). http://uiml.org/, March 2004.

[Pat99] F. Paternò. *Model-Based Design and Evaluation of Interactive Applications*. Springer, London, 1999.

[RP08] J. Rückert und B. Paech. The Guilet Dialog Model and Dialog Core for Graphical User Interfaces. In P. Forbrig und F. Paternò, Hrsg., *Engineering Interactive Systems 2008*, Jgg. 5247 of *LNCS*, Seiten 197–204. Springer, 2008.

[Sta05] A. Stanciulescu. A transformational approach for multimodal web user interfaces based on UsiXML. In *7th International Conference on Multimodal interfaces (ICMI'05)*, Seiten 259–266. ACM, 2005.

[Træ03] H. Trætteberg. Dialog Modelling with Interactors and UML Statecharts - A Hybrid Approach. In *DSV-IS*, Seiten 346–361, 2003.

[W3Ca] W3C. Composite Capabilities / Preference Profiles: Structure and Vocabularies 2.0. http://www.w3.org/Mobile/CCPP/.

[W3Cb] W3C. State Chart XML (SCXML): State Machine Notation for Control Abstraction. Working Draft, 16 May 2008. http://www.w3.org/TR/scxml/.

[WAP01] WAP Forum. Wireless Application Protocol, User Agent Profile, WAP-248-UAPROF-20011020-a, 2001. http://www.openmobilealliance.org/.

[Wei94] M. Weiser. The world is not a desktop. *Interactions*, 1(1):7–8, 1994.

[WFTS07] M. Wurdel, P. Forbrig, Radhakrishnan T. und D. Sinnig. Patterns for Task- and Dialog-Modeling. In *HCI (1)*, Seiten 1226–1235, 2007.