Kontextbasierte Auflösung von Mehrdeutigkeiten beim iterativen Entwurf von Benutzungsschnittstellen

Matthias Freund, Christopher Martin, Annerose Braune

Institut für Automatisierungstechnik
Technische Universität Dresden
01062 Dresden
{matthias.freund, christopher.martin, annerose.braune}@tu-dresden.de

Abstract: Modelltransformationen stellen das verbindende Element zwischen den Modellebenen innerhalb der modellbasierten Entwicklung von Benutzungsschnittstellen dar. Sie bilden Quell- auf Zielelemente ab und enthalten Vorschriften zur Umsetzung dieser Abbildungen. Ansätze zur Wiederverwendung von Transformationsvorschriften nutzen formale Transformationsmodelle, die nur die Abbildungen beschreiben und von deren Umsetzung abstrahieren. Vorhandene Lösungen können nur eindeutige (1-zu-1) Abbildungen beschreiben. Im Allgemeinen existieren aber mehrdeutige (1-zu-n) Elementzuordnungen, aus denen erst im Entwurfsprozess vor allem basierend auf dem Nutzungskontext eindeutige Abbildungen auszuwählen sind. Das Wissen, welche Quellelemente prinzipiell auf welche Zielelemente abbildbar sind und wie diese vom Nutzungskontext abhängen, wird bisher nicht formalisiert. Der vorliegende Beitrag stellt als Lösungsansatz ein bedingtes mehrdeutiges Abbildungs- und Transformationsmodell vor und beschreibt dessen Wiederverwendung in einem iterativen Entwurfsprozess sowie eine Methodik zur kontextbasierten Auflösung der Mehrdeutigkeiten.

1 Einleitung

Vorschriften zur Modelltransformation haben zwei grundlegende Aufgaben zu beschreiben [GPR06]: (1) die Abbildung der Quell- auf Zielelemente und (2) die Umsetzung der Abbildungen, d.h. die Durchführung der gesamten Transformation (z.B. auch das Einlesen, Erzeugen der Modelle etc.). Während fest codierte, d.h. ausprogrammierte, Transformationen im Allgemeinen nicht zwischen Abbildung und Durchführung unterscheiden, favorisieren aktuelle Ansätze die Formulierung von formalen Abbildungsmodellen unabhängig von der Implementierungstechnologie der Durchführung. Diese Modelle beschreiben, welches Modell- oder Metamodellelement der Quellsprache auf welches einer Zielsprache abgebildet werden kann bzw. soll. Der Vorteil solcher Abbildungsmodelle ist, dass diese Zuordnungen nicht programmiert, sondern konfiguriert werden und somit nur geringe Implementierungskenntnisse erforderlich sind. Aktuelle Lösungen erlauben aber im Wesentlichen nur eindeutige (1-zu-1) Abbildungen, die ein Quellelement auf genau ein Zielelement abbilden [Po12]. Bei der modellgetriebenen Entwicklung von Benutzungsschnittstellen (User Interfaces, UIs)

treten aber im Allgemeinen mehrdeutige (1-zu-n) Elementzuordnungen auf [PE99]. So lässt sich beispielsweise eine abstrakt formulierte Anweisung „wähle Prozessvariable aus“ durch eine Dropdown-Liste, durch Radio-Buttons aber auch durch eine Sprachauswahl realisieren. Dabei ist es ebenfalls möglich, dass mehrere Quellelemente dem/den gleichen Zielelement(en) zugeordnet werden können, sodass sich sogar m-zu-n-Zuordnungen ergeben. Diese lassen sich jedoch durch mehrere 1-zu-n-Zuordnungen darstellen. Welches der möglichen Zielelemente zur Entwurfszeit einer Benutzungsschnittstelle gewählt wird, hängt neben den Vorlieben des Entwicklers oder von Firmenstandards vor allem vom Kontext ab, in dem die Benutzungsschnittstelle später eingesetzt wird. Dieser setzt sich laut [Ca03] zusammen aus der Plattform, auf der das UI ausgeführt wird, dem Nutzer, der mit dem UI interagiert, und der Umgebung, in der das UI eingesetzt wird. Vorausgesetzt wird in jedem Fall, dass der Entwickler einer Benutzungsschnittstelle (UI-Designer) für jedes Quellelement alle möglichen Zuordnungen von Zielelementen kennt und daraus das jeweils geeignete auswählt. Dieses Wissen hängt allerdings sehr von dessen Erfahrungen allgemein und speziell von den Erfahrungen im Umgang mit der Quell- und Zielsprache ab.

Die Verfügbarkeit einer formalisierten und damit wiederverwendbaren Beschreibung aller zulässigen Abbildungen eines Sprachenpaars – die mehrdeutigen (1-zu-n) Zuordnungen – könnte den Entwurfsprozess erleichtern und damit die Qualität von Benutzungsschnittstellen verbessern. Eine weitere Verbesserung des Entwurfsprozesses sollte sich ergeben, wenn die 1-zu-n-Abbildungen mit Kontextbedingungen annotiert werden, sodass während der Transformation lediglich ein Kontextmodell, welches den aktuellen Zielkontext beschreibt, ausgewertet werden muss. Anhand der Auswertung dieses Kontextmodells und der bedingten mehrdeutigen Abbildungen kann dann im Idealfall komplett automatisch ein passendes Zielelement ausgewählt werden. In jedem Fall wird die Liste möglicher Zielelemente aber auf (für den gewählten Kontext) sinnvolle Möglichkeiten beschränkt, sodass bspw. eine anschließende Auswahl durch den UI-Designer effizientere Ergebnisse erwarten lässt. Das Wissen um alle zulässigen (mehrdeutigen) Abbildungen hängt im Wesentlichen vom Sprachenpaar ab, ist unabhängig von einer konkreten Benutzungsschnittstelle und kann somit für alle Applikationen dieses Sprachenpaars wiederverwendet werden. Um im Falle iterativer Verbesserungen des entworfenen UIs die einmal getroffenen Zuordnungen wiederverwenden zu können, sollten diese geeignet beschrieben und für weitere Transformationen gespeichert werden.

Gegenstand dieses Beitrages ist deshalb die Entwicklung allgemeiner Beschreibungsmittel zur Formulierung sowohl der zulässigen mehrdeutigen Abbildungen als auch der ausgewählten eindeutigen Abbildungen für ein UI, d.h. zu entwickeln ist ein Metamodell zur Formulierung von Abbildungsmodellen. Des Weiteren beschreibt dieser Beitrag eine Möglichkeit zur formalen Definition von Kontextbedingungen basierend auf Kontextmodellen sowie zur Zuordnung dieser Bedingungen zu den mehrdeutigen Abbildungen. Abschnitt 2 ordnet die geplanten Entwicklungen in den Forschungskontext ein. Abschnitt 3 stellt ein Vorgehen und die Integration von Abbildungsmodellen vor. Die Konzeption der benötigten Sprachmittel wird anschließend in Abschnitt 4 dargelegt. Die Anwendbarkeit des Vorgehens in Verbindung mit dem entwickelten Metamodell wird dann in einer Fallstudie in Abschnitt 5 untersucht. Abschnitt 6 wertet die Erfahrungen aus und leitet weiteren Handlungsbedarf ab.

2 Einordnung in den Forschungskontext

In konventionellen Transformationen sind sowohl die Abbildungsregeln zwischen Elementen des Quell- und des Zielmetamodells als auch die Erzeugung der Elemente des Zielmodells fest kodiert. Dies wird mit Implementierungstechnologien wie ATL [Ec12], QVT [Ob11] oder XSLT [W3C99] für genau ein Paar von Metamodellen und genau einen speziellen Kontext realisiert. Da diese Transformationen nur eindeutige Abbildungsregeln nutzen, führt ein Wechsel des Kontextes zu einem neuen Satz von Abbildungsregeln und daher zu einer neuen Transformation. Die auftretenden möglichen (mehrdeutigen) Abbildungen werden bisher entweder gar nicht berücksichtigt, sodass lediglich eines der möglichen Zielelemente genutzt wird, oder nur auf informale Art fest in die Transformation programmiert [Ha11].

Bézivin et Al. [Bé06] zeigen in einer Grundsatzarbeit, dass Abbildungen – im Gegensatz zu fest codierten Transformationen – auch formal in sogenannten Transformationsmodellen definiert werden können. Dabei werden die Abbildungen zwischen den Metamodellelementen beschrieben, indem Quell- und Zielelemente formal aufeinander abgebildet werden. Dieser Ansatz wird auch von der UI-Beschreibungssprache UsiXML verwendet, die ein Transformationsmodell definiert, welches die Beschreibung von Modelltransformationen basierend auf Graph-Grammatiken ermöglicht [Li05]. Da auch ein Kontextmodell an solch ein Transformationsmodell angeheftet werden kann, ist es theoretisch möglich, bedingte Abbildungen bzw. Transformationen zu definieren. Dies benötigt allerdings für jeden Kontext ein eigenes vollständiges Transformationsmodell. Folglich können Kontext-Abhängigkeiten nur implizit spezifiziert werden, da jeder neue zu unterstützende Kontext ein neues Abbildungsmodell benötigt. Es handelt sich folglich lediglich um eine Sammlung an eindeutigen Abbildungsmodellen, wobei jedes Modell zu einem bestimmten Kontext gehört.

Die Sprachfamilie MARIA [PSS09] ermöglicht ebenfalls die formale Definition eindeutiger Abbildungen mithilfe des grafischen Editors MARIAE [Li11]. Die so definierten Abbildungen werden jedoch ohne Einflussmöglichkeit auf jede Instanz eines Typs (Metamodellelements) angewandt. Das bedeutet, dass z.B. jedes „Auswahlelement“ einer kompletten Benutzungsschnittstelle auf eine Dropdown-Liste abgebildet werden muss. Eine alternative Abbildung einzelner Elemente auf einen Radio-Button oder auf eine Sprachauswahl ist folglich nur durch eine manuelle Änderung des Zielmodells möglich. Es existieren demnach keine wiederverwendbaren Informationen über mehrdeutige Abbildungen und mögliche Kontextabhängigkeiten. Des Weiteren ist MARIAE auf die Definition von Abbildungen zwischen den Sprachen der MARIA-Familie limitiert und kann nicht für andere UI-Beschreibungssprachen verwendet werden.

Einen ersten Ansatz für mehrdeutige Abbildungsmodelle wird von Aquino et Al. [APP10] durch den Vorschlag von sogenannten TransformationTemplates gemacht, die das UsiXML-Transformationsmodell [Li05] erweitern. Sie ermöglichen die Definition von (möglicherweise mehrdeutigen) Abbildungen durch Angabe eines Selektors, der auf das Quellmetamodell angewendet wird. Des Weiteren können Gewichtungsfaktoren definiert werden, die das UsiXML-Kontextmodell referenzieren und die genutzt werden, falls mehrere mögliche Zuordnungen für ein Quellelement identifiziert werden. Aller-

dings wird das Transformationsmodell imperativ spezifiziert, da es Implementierungsdetails der Transformation beinhaltet. Durch die imperative Beschreibung der Abbildungen in diesem Ansatz ist die Nutzung von Variablen, Bedingungen und Schleifen-Konstrukten möglich. Dadurch wird allerdings tiefes Wissen der Syntax und Semantik der TransformationTemplates benötigt, was einen Transformationsexperten voraussetzt und somit die Anwendung durch UI-Designer erschwert. Darüber hinaus ist auch bei diesem Ansatz nur die implizite Definition von Kontextabhängigkeiten möglich, da wiederum jeder Kontext seine eigenen TransformationTemplates benötigt.

Im Gegensatz zur imperativen Modellierung beschreibt eine deklarative Modellierung von Transformationen lediglich, welche Elemente unter welchen Bedingungen aufeinander abgebildet werden sollen. Es wird nicht beschrieben, wie diese Abbildungen durch die Transformation umgesetzt werden. Folglich wird nur das was aber nicht das wie einer Transformation beschrieben. Dies erleichtert die Nutzung durch UI-Designer ohne tiefgreifendes Wissen der Implementierungstechnologien. Ein solcher deklarativer Modellierungsansatz wird bereits von dem grafischen Editor MARIAE umgesetzt (s.o.). Im Umfeld adaptiver Benutzungsschnittstellen definieren bspw. [SCF06] und [Bl09] allgemeine Metamodelle zur deklarativen Beschreibung von Abbildungen. Eine Annotation mit Kontextbedingungen, die bei deklarativen Ansätzen leicht möglich ist, ist bei diesen Ansätzen jedoch nicht vorgesehen. Ein erster Ansatz für die deklarative Modellierung mehrdeutiger Abbildungen und den Einsatz von Kontextbedingungen wird in [Po12] vorgestellt. Allerdings ist der vorgestellte Ansatz auf eine bestimmte Kombination von Ein- und Ausgangsmetamodellen festgelegt – andere Metamodelle würden auch ein neues Abbildungsmetamodell bzw. eine neue Transformationssprache erfordern.

Die Nachteile deklarativer Ansätze sind im Allgemeinen ihre beschränkten Ausdrucksmittel, die es beispielsweise nicht ermöglichen, Abhängigkeiten zwischen verschiedenen Abbildungsregeln zu beschreiben. Nichtsdestotrotz benötigt die Erstellung imperativer Transformationsmodelle erweitertes Wissen über Syntax und Semantik der Implementierungstechnologie und erschwert dadurch das Hinzufügen expliziter Kontextbedingungen. Daher verwendet der im Folgenden vorgestellte Ansatz ein deklaratives Abbildungsmodell. Des Weiteren benötigen die vorgestellten explizit definierten Kontextabhängigkeiten kein neues Abbildungsmodell für jeden neuen Kontext, der unterstützt werden soll.

3 Vorgehen zur Definition und Verwendung bedingter mehrdeutiger Abbildungen

Die Notwendigkeit zur formalen Definition mehrdeutiger Abbildungen und deren Abhängigkeiten vom Kontext einer Benutzungsschnittstelle führen zu bedingten mehrdeutigen Abbildungen, die während der eigentlichen Transformation ausgewertet und daraufhin für iterative Vorgehensweisen persistiert werden müssen. Dieser Abschnitt stellt einen Arbeitsablauf zur Definition und Anwendung solcher Abbildungen als Realisierung der in den vorigen Abschnitten herausgearbeiteten Anforderungen dar. Abbildung 1 zeigt den gesamten Arbeitsablauf inklusive der beteiligten Quell-, Ziel- und Kontextmodelle und ihrer Metamodelle.

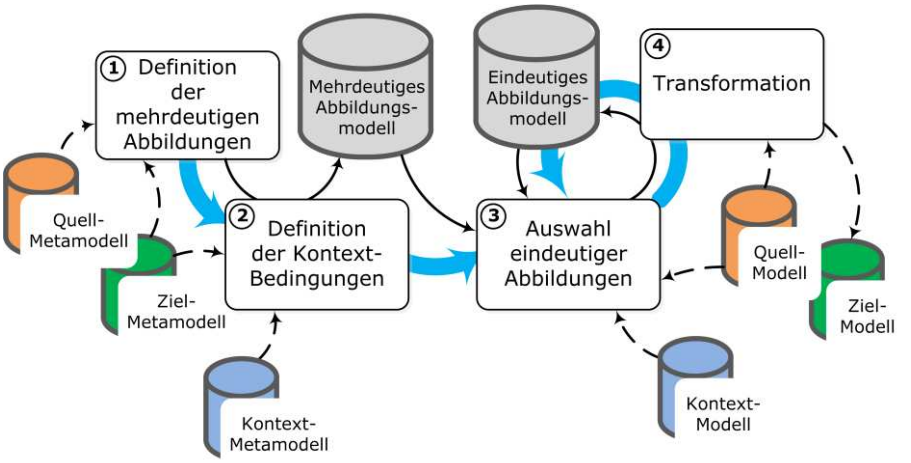

Abbildung 1: Vorgeschlagenes Vorgehen

Zunächst müssen die bedingten mehrdeutigen Abbildungen definiert werden. Dies geschieht in zwei Schritten: In Schritt 1 werden sämtliche Abbildungen von Elementen des Quell- auf Elemente des Zielmetamodells definiert. Beispielsweise könnte ein abstraktes „Auswahlelement“ entweder auf eine Dropdown-Liste, auf Radio-Buttons oder auf eine Sprachauswahl abgebildet werden. Dieser Schritt muss lediglich ein einziges Mal für jedes Paar von Metamodellen bzw. Sprachen durchgeführt werden. Anschließend können Kontextbedingungen an die mehrdeutigen Abbildungen angeheftet werden (Schritt 2 in Abbildung 1). Die in Schritt 1 definierte Abbildung auf eine Sprachauswahl ist beispielsweise nur dann sinnvoll, wenn die spätere Zielplattform mit einem Mikrofon bzw. einer Spracherkennung ausgerüstet ist und die Benutzungsschnittstelle an Orten ohne Störgeräusche eingesetzt werden soll. Die so erhaltenen bedingten mehrdeutigen Abbildungen müssen zur weiteren Verwendung in einem Modell gespeichert werden (mehrdeutiges Abbildungsmodell, vgl. Abbildung 1).

Die Anwendung der definierten Abbildungen auf ein konkretes UI besteht ebenfalls aus zwei Schritten: Zunächst wird basierend auf einer Auswertung der Kontextbedingungen und eines Kontextmodells automatisch für jedes Quellelement ein passendes Zielelement bestimmt (Schritt 3). Besagt das Kontextmodell beispielsweise, dass die Plattform mit einem Mikrofon ausgestattet ist und dass es sich um eine ruhige Umgebung handelt, kann das Sprachauswahlelement aus obigem Beispiel genutzt werden. Dieser Schritt muss im Allgemeinen nicht nur für jeden Element-Typ (Metamodellelement) sondern für jedes Element (Instanz eines Metamodellelements) durchgeführt werden, da nicht davon ausgegangen werden kann, dass alle Elemente eines Quell-Typs auf denselben Ziel-Typ abgebildet werden sollen (vgl. [He11]). Danach werden die bestimmten eindeutigen Abbildungen durch die eigentliche Transformation ausgeführt und das Zielmodell wird erzeugt (Schritt 4). Die eindeutigen Abbildungen müssen für mögliche iterative Vorgehensweisen ebenfalls in einem entsprechenden Modell (eindeutiges Abbildungsmodell, vgl. Abbildung 1) gespeichert werden, sodass sie bei einer Wiederholung der Schritte 3 und 4 wiederverwendet werden können.

4 Das persistente mehrdeutige Abbildungs- und Transformations-Metamodell

In diesem Abschnitt werden die entwickelten Sprachmittel vorgestellt, die das in Abschnitt 3 vorgestellte Vorgehen realisieren. Unterschieden werden die Definition und das Speichern mehrdeutiger Abbildungen (Schritt 1 in Abbildung 1), das Hinzufügen von Kontextbedingungen (Schritt 2) sowie das Speichern ausgewählter eindeutiger Abbildungen im Falle iterativer Vorgehensweisen (Schritt 3 und 4).

4.1 Definition mehrdeutiger Abbildungen

Abbildungsregeln werden auf der Ebene von Metamodellen formuliert [GPR06]. Sie definieren Abbildungen von einem Quell- auf ein oder mehrere Ziel-Metamodellelemente. Ein entsprechendes Abbildungsmetamodell muss daher die Definition dieser Abbildungen ermöglichen und dazu sowohl das Quell- als auch das Zielmetamodell referenzieren. Um eine Erstellung des Abbildungsmodells durch einen Domänenexperten bzw. UI-Designer ohne tiefgreifende Kenntnisse über Transformationssprachen zu ermöglichen, soll eine deklarative Beschreibung der Abbildungen genutzt werden (vgl. Abschnitt 2). Solch eine deklarative Beschreibung von Abbildungen erfordert im Vergleich zu imperativen Ansätzen im Allgemeinen einen erhöhten Aufwand bei der Umsetzung durch die Transformationsdurchführung. Allerdings wird dieser Aufwand dadurch relativiert, dass solch eine Umsetzung nur einmal pro Metamodell-Paar zu implementieren ist.

Eine deklarative Modellierung einer Abbildung besteht prinzipiell lediglich aus einem Modellelement Mapping und Referenzen auf das Quell- und das/die Ziel-Metamodellelement(e). Möchte man im weiteren Verlauf Kontextbedingungen zu einem Mapping hinzufügen, reicht diese einfache Struktur nicht aus: Die Bedingungen könnten in diesem Fall lediglich dem gesamtem Mapping und nicht den verschiedenen Zielelementen zugeordnet werden. Abbildung 2(a) zeigt die resultierende, erweiterte Metamodell-Struktur. Diese ermöglicht es, einem Mapping mehrere MappingTargets zuzuordnen. Ein MappingTarget wiederum referenziert genau ein Element eines Zielmetamodells. Durch die Referenzierung beliebiger Metamodellelemente von Quell- und Zielsprachen ist dieser Ansatz allgemeingültig, d.h. für beliebige Quell- und Zielmetamodelle nutzbar.

4.2 Aufbau und Struktur von Kontextbedingungen

Der Zweck von Kontextbedingungen ist es, Einschränkungen basierend auf dem Nutzungskontext des UIs zu definieren, unter denen jede der mehrdeutigen Abbildungen verwendet werden kann. Es muss folglich beschrieben werden, welche Eigenschaften des Kontexts erfüllt sein müssen, um den Einsatz einer bestimmten Abbildung sinnvoll zu gestalten.

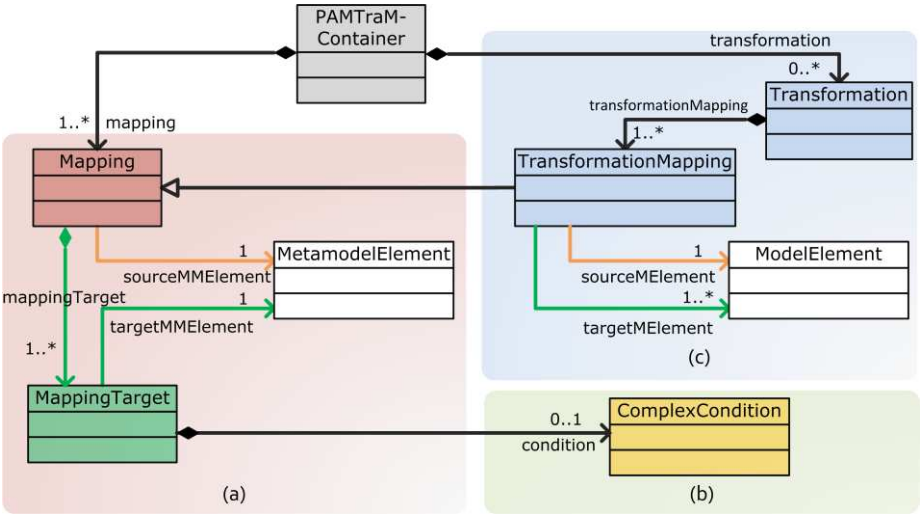

Abbildung 2: Resultierender Aufbau des Metamodells zur (a) Speicherung mehrdeutiger Abbildungen, (b) Beschreibung von Kontextbedingungen und (c) Speicherung durchgeführter Transformationen

Die Charakteristik eines konkreten Kontexts wird durch ein Kontextmodell beschrieben. Typische Kontextmetamodelle sind beispielswiese das UsiXML ContextModel [Us07] oder die Delivery Context Ontology [W3C09]. Ein Ausschnitt aus dem Metamodell für das UsiXML ContextModel ist in Abbildung 3 dargestellt. Es zeigt, dass relevante Charakteristiken von Plattform, Umgebung und Nutzer durch Attribute beschrieben werden (z.B. screenWidth als Bildschirmbreite der Plattform). Um wiederum eine Allgemeingültigkeit für verschiedene Kontextmetamodelle zu erreichen, müssen auch die Kontextbedingungen die Referenzierung allgemeiner Metamodell-Elemente ermöglichen (vgl. Abschnitt 4.1). Dies bedeutet, dass die Kontextbedingungen das verwendete Kontextmetamodell referenzieren müssen. Die entsprechenden Elemente des Kontextmetamodells können folglich zur Definition der Kontextbedingungen genutzt werden. Dazu wurde eine einfache Metamodell-Struktur definiert, die aus drei Teilen besteht: (1) die Referenz eines Attributes des Kontextmetamodells, (2) einem Komparator und (3) einem Vergleichswert. Einfache Beispielbedingungen, im vorliegenden Fall unter Verwendung des in Abbildung 3 dargestellten UsiXML-Kontextmodells, könnten die Forderung nach einer minimalen Bildschirmhöhe von 1080 Pixeln (screenHeight >= 1080px) oder nach einer ruhigen Umgebung (isNoisy == false) sein. Abbildung 2(b) zeigt die Metamodellstruktur zur Zuordnung der Bedingungen (vereinfacht als ComplexCondition dargestellt) zu einem MappingTarget.

Während der Auswahl konkreter (eindeutiger) Abbildungen (Schritt 3 in Abbildung 1), liest der Auswahlmechanismus den Wert des Elements aus dem Kontext-Modell, welcher durch Teil (1) der Bedingung identifiziert wird und vergleicht ihn mit dem durch Teil (3) definierten Vergleichswert. Die Art des Vergleiches wird durch den Komparator (Teil (2) der Bedingung) vorgegeben. Je nach Ergebnis des Vergleiches stellt das referenzierte MappingTarget eine sinnvolle Wahl für den aktuellen Kontext dar oder nicht.

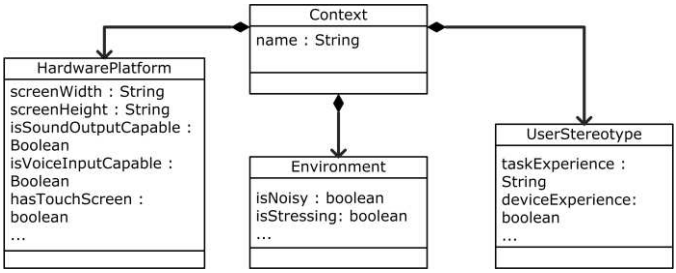

Abbildung 3: Ausschnitt aus dem Metamodell des UsiXML ContextModels

Um die Definition komplexerer Bedingungen zu ermöglichen, können mehrere Bedingungen durch Und- und Oder-Verknüpfungen kombiniert oder auch negiert werden. Dies erlaubt die Definition komplexer Boolescher Ausdrücke. Beispielsweise könnte eine typische Bedingung eine Mindesthöhe und –breite des Bildschirms fordern.

4.3 Disjunktheit von Kontextbedingungen

Die Auswahl eindeutiger Abbildungen für konkrete Kontexte ist nicht zwingend, d.h. die für eine mehrdeutige Abbildung formulierten Bedingungen müssen nicht disjunkt sein. Daraus folgt, dass nach der Auswertung der Bedingungen eine Mehrdeutigkeit bestehen bleiben kann, sodass immer noch mehrere Zielelemente zur Auswahl stehen. Beispielsweise ist eine mehrdeutige Abbildung mit den beiden Zielelementen Dropdown-Liste und Radio-Buttons nicht zwingend durch Kontextmodelle unterscheidbar.

In solchen Fällen müssen weitere Auswahl-Verfahren angewendet werden. Dies könnten beispielsweise statistische Verfahren sein, die frühere Transformationen auswerten, oder eine Nutzerabfrage, bei der der UI-Gestalter entscheiden kann, welche Variante genutzt werden soll. Die vorherige Auswertung der Kontextbedingungen sollte aber in jedem Fall zu einer Verbesserung der Qualität im Auswahlprozess führen, da (für den Kontext) widersinnige Alternativen aussortiert werden. Dadurch kann die Qualität der Resultate statistischer Methoden oder die Gebrauchstauglichkeit bei einer Nutzer-Abfrage erhöht werden, da der Nutzer nur noch zwischen sinnvollen Varianten unterscheiden muss.

4.4 Speichern konkreter Abbildungen

Die Mehrdeutigkeiten der Abbildungen werden während der Transformation durch Auswertung der Kontextbedingungen (vgl. Abschnitt 4.2) und evtl. weitere Auswahl-Verfahren (vgl. Abschnitt 4.3) für jedes Quellelement beseitigt und somit werden eindeutige Abbildungen gebildet. Dieser Prozess ist zeitaufwändig – besonders wenn eine Nutzerabfrage verwendet wird – und muss für jeden Iterationszyklus des Entwurfsprozesses wiederholt werden. Um dies zu verhindern, sollten solche Design-Entscheidungen gespeichert werden, sodass bei einer erneuten Transformation der Entscheidungsprozess nur für diejenigen Elemente des UIs wiederholt werden muss, die während des letzten Iterationsschrittes hinzugefügt oder geändert wurden.

Zu diesem Zweck repräsentiert das Metamodell-Element Transformation in Abbildung 2(c) eine konkrete Transformation, die für ein spezifisches UI und einen spezifischen Kontext ausgeführt wurde. Gruhn et Al. definieren eine Transformation als Instanz von Abbildungen auf Modell-Ebene [GPR06]. Aus diesem Grund werden die zu einer Transformation gehörenden, durch einen Auswahlprozess bestimmten, konkreten Abbildungen als TransformationMappings gespeichert (s. Abbildung 2(c)). Analog zu den Mappings wird auch hier wieder eine deklarative Struktur genutzt – im Gegensatz zu den Mappings bilden die TransformationMappings allerdings konkrete Modellelemente aufeinander ab. Es ist darauf hinzuweisen, dass nicht zwingend nur ein einzige Zielelement ausgewählt werden muss: Besonders bei multimodalen UIs kann es oftmals sinnvoll sein, ein abstraktes Element auf mehrere konkrete Elemente abzubilden, beispielswiese um eine grafische mit einer auditiven Ausgabe zu kombinieren.

4.5 Integration in ein gemeinsames Metamodell

Sowohl ein bedingtes mehrdeutiges Abbildungsmodell als auch ein eindeutiges Abbildungsmodell als Sammlung durchgeführter Transformationen (vgl. Abbildung 1) sind jeweils bezogen auf genau eine Kombination von Quell-, Ziel- und Kontextsprachen. Aus diesem Grund wurden die in den vorherigen Abschnitten besprochenen Sprachmittel in ein gemeinsames Metamodell integriert (vgl. Abbildung 2) – das persistente mehrdeutige Abbildungs- und Transformationsmetamodell (Persistent Ambiguous Mapping and Transformation Meta-Model – PAMTraM). Dessen Wurzelelement (PAMTraMContainer, vgl. Abbildung 2) dient zur Speicherung sowohl der bedingten mehrdeutigen Abbildungen (vgl. Abbildung 2(a) und (b)) als auch der durchgeführten Transformationen inklusive der verwendeten eindeutigen Abbildungen (vgl. Abbildung 2(c)).

5 Fallstudie

In diesem Abschnitt soll der vorgestellte Ansatz anhand einer Fallstudie demonstriert werden. Abbildung 4 zeigt eine Kleinversuchsanlage, deren Füllstände auf einem UI dargestellt werden sollen. Dazu soll der Nutzer in der Lage sein, einen der drei vorhandenen Behälter auszuwählen, dessen aktueller Füllstand daraufhin dargestellt wird. Ausgangspunkt dieser Fallstudie soll ein in MARIA (vgl. Abschnitt 2) beschriebenes Modell auf Ebene des Abstract User Interface (AUI) des Cameleon Reference Frameworks (CRF) [Ca03] sein. Dieses wird unter Nutzung der PAMTraM in ein MARIA-Modell auf Ebene des Concrete User Interface (CUI) im CRF überführt. Die PAMTraM wurde dafür innerhalb des Eclipse Modeling Frameworks (EMF) [Ec13a] realisiert. Um eine Erstellung der Modelle mit höherem Komfort zu ermöglichen, wurden textuelle Editoren mit Hilfe des Werkzeugs EMFText [De13] implementiert. Für die einfachere Verarbeitung der Modelle in der EMF-Umgebung wurden die MARIA XML-Schemata in EMF-basierte Metamodelle überführt. Für die Definition der Kontextabhängigkeiten wird das UsiXML-Kontextmodell (vgl. Abbildung 3) genutzt. Die abschließende Model-to-Text-Transformation zur Generierung der finalen Oberfläche wurde nur prototypisch implementiert, um mögliche Repräsentationen des sich ergebenden UIs zeigen zu können.

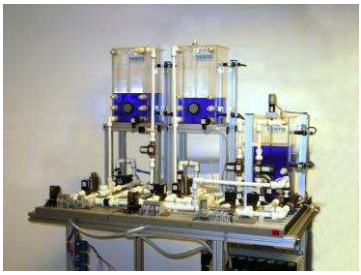

Abbildung 4: Für die Fallstudie verwendete Kleinversuchsanlage

MARIA ist eine XML-basierte Beschreibungssprache für grafische, sprachbasierte und multimodale Benutzungsschnittstellen, welche die obersten drei Ebenen des CRF abdeckt [PSS09]. Während auf der AUI-Ebene ein einzelnes Metamodell existiert, bietet MARIA auf Ebene des CUI mehrere Metamodelle an. Für diese Fallstudie wird das Metamodell des MultimodalDesktopCUIs verwendet, das die Modellierung von multimodalen UIs, wie etwa einer grafischen Oberfläche mit Spracheingabemöglichkeiten, ebenso wie die Definition reiner grafischer oder sprachbasierter UIs erlaubt.

Sowohl auf der AUI- als auch auf der CUI-Ebene setzt sich ein MARIA-Modell aus drei Teilen zusammen: aus mehreren Präsentationen, einer Definition der externen Funktionen des UIs und einem Datenmodell. Der einzige Unterschied zwischen den Modellen der Fallstudie liegt in den Präsentationen, da durch die Modelltransformation von AUI nach CUI zusätzliche Informationen über die genutzten Modalitäten hinzugefügt werden. Daher muss das Abbildungsmodell die Zuordnungen der AUI- auf die entsprechenden CUI-Elemente beschreiben. Tabelle 1 zeigt einen Ausschnitt der in MARIA verfügbaren UI-Elemente auf der AUI- und CUI-Ebene sowie die dazugehörigen möglichen Abbildungen zwischen den Elementen. Dabei wird offensichtlich, dass Mehrdeutigkeiten auftreten. So kann beispielsweise ein Single Choice (AUI) auf eine Drop-down List, einen Radio Button oder eine Single Vocal Selection abgebildet werden.

Tabelle 1: Ausschnitt der möglichen Abbildungen vom MARIA AUI auf das MultimodalDesktopCUI

		AUI		
		Activator	Description	Single Choice
MultimodalDesktopCUI	Audio		x	
	Button	x		
	Drop-down List			x
	Image		x	
	Radio Button			x
	Table		x	
	Text		x	
	Vocal Activator	x		
	Vocal Output		x	
	Vocal Selection			x

Tabelle 1 stellt die Grundlage für die Definition der mehrdeutigen Abbildungen in der PAMTraM (Schritt 1 in Abbildung 1) dar. Abbildung 5 zeigt den entwickelten textuellen Editor, der den Entwickler bei der (deklarativen) Definition der mehrdeutigen Abbildungen unterstützt. Die definierten Abbildungen werden automatisch in einer Instanz der PAMTraM gespeichert. Als nächstes müssen die Kontextbedingungen zu den Abbildungszielen (vgl. Abbildung 2 (a) und (b)) hinzugefügt werden (Schritt 2 in Abbildung 1). Das verwendete UsiXML-Kontextmetamodell wurde ebenfalls als EMF-basiertes Metamodell entsprechend der Beschreibungen in [Us07] umgesetzt. Basierend darauf können nun mithilfe des textuellen Editors Kontextbedingungen erstellt werden. Ein Ausschnitt eines möglichen Satzes solcher Kontextbedingungen ist in Abbildung 6 zu sehen. Sie spezifizieren unter anderem, dass ein Image erst ab einer Bildschirmgröße von 800×600px verwendet werden soll. Des Weiteren wird definiert, dass eine Sprachauswahl (VocalSelectionType) eine Plattform, die zur Spracheingabe fähig ist, und eine ruhige Umgebung benötigt.

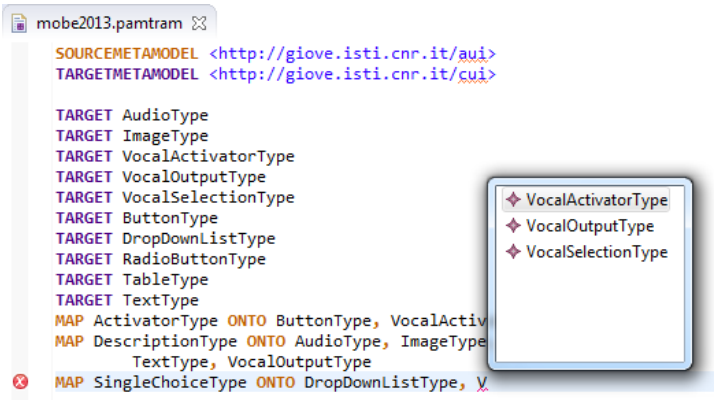

Abbildung 5: Erstellung der PAMTraM mithilfe des textuellen Editors

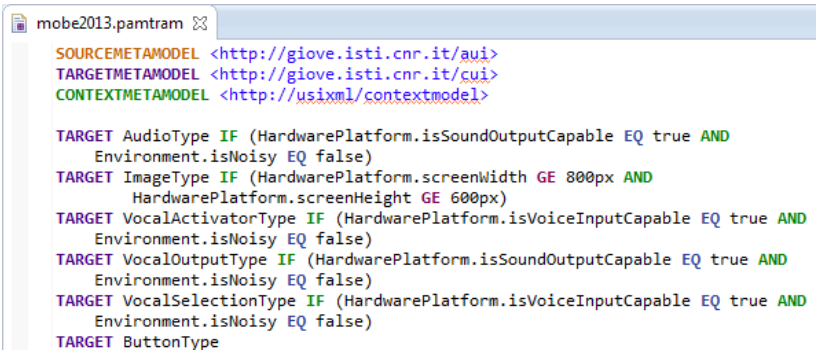

Abbildung 6: Hinzufügen von Kontextbedingungen zu den Abbildungszielen

Um die Funktionalität der PAMTraM zu demonstrieren, wurde ein einfaches AUI-Beispielmodell erstellt und für zwei unterschiedliche Kontexte (jeweils repräsentiert durch ein entsprechendes Kontextmodell) transformiert: Der erste Kontext besteht aus einer Plattform mit einem großen Bildschirm (1600×1200px) aber ohne Mikrofon (bspw. ein Desktop-PC) während der zweite Kontext eine Plattform mit kleinerem Bildschirm (800×480px) aber dafür mit Mikrofon repräsentiert (bspw. ein Smartphone). Für beide Kontexte soll gelten, dass es sich um eine ruhige Umgebung handelt.

Die Modelltransformation von AUI nach CUI, welche die PAMTraM auswertet, wurde mithilfe des Transformationsframeworks Epsilon [Ec13b] umgesetzt. Um die nach der Auswertung der Kontextbedingungen eventuell verbleibenden Mehrdeutigkeiten (vgl. Abschnitt 4.3) aufzulösen, wurden Nutzerabfragen implementiert, mit deren Hilfe der Nutzer aus den verbleibenden Elementen auswählen kann. Abbildung 7 zeigt die Aufforderung zu einer Nutzerinteraktion während der Transformation eines Single Choice-Elementes: Während für den zweiten Kontext auch eine VocalSelection realisierbar ist, da es sich um ein Gerät mit Mikrofon und eine ruhige Umgebung handelt, stehen für den erste Kontext lediglich grafische Realisierungsmöglichkeiten bereit, da die entsprechend Plattform nicht über ein Mikrofon verfügt.

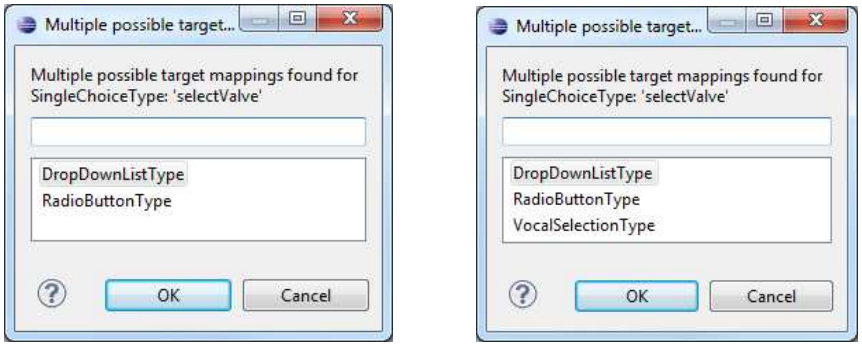

Abbildung 7: Nutzerabfragen während der Transformation eines SingleChoice-Elementes für Kontext 1 (links) und Kontext 2 (rechts)

Die während der Transformation ausgewählten Abbildungen werden, wie in Abschnitt 4.4 beschrieben, als TransformationMappings persistiert. Abbildung 8 zeigt den entsprechenden Teil einer Instanz der PAMTraM (vgl. Abbildung 2(c)): Die obere Hälfte der Abbildung zeigt, dass eine gespeicherte Transformation aus mehreren Abbildungen besteht, wobei für jedes UI-Element ein eigenes TransformationMapping vorgesehen wird. Auf der unteren Hälfte der Abbildung werden die Details des ausgewählten Mappings für ein description-Element gezeigt, welches im CUI-Modell auf ein image abgebildet wurde.

Bei einer erneuten Transformation des AUI-Modells, wie etwa bei einer iterativen Entwicklung, kann der Nutzer eine zuvor durchgeführte Transformation auswählen, sodass erneut die gesicherten eindeutigen Abbildungen für die Transformation genutzt werden. Eine Auswertung der Kontextbedingungen und eine eventuelle Nutzerinteraktion ist in diesem Fall nur für neuhinzugekommene UI-Elemente notwendig (vgl. Abschnitt 4.2

und 4.4). Abbildung 9 zeigt zwei resultierenden finale UIs nach einer Model-to-Text-Transformation der CUI-Modelle, die aus den vorherigen Schritten entstanden sind: Während eines der Description-Elemente für Kontext 1 in ein Image-Element transformiert wurde, wurde für Kontext 2 ein Text-Element und eine (hier nicht darzustellende) Sprachausgabe generiert.

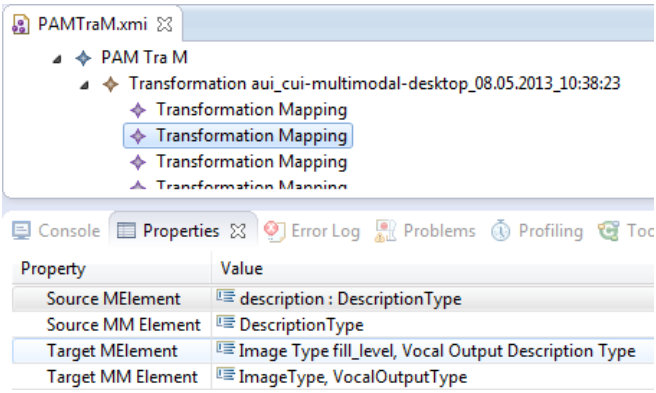

Abbildung 8: Automatische Persistierung der gewählten Abbildungen als TransformationMappings in der PAMTraM

Abbildung 9: Zwei resultierende Versionen des finalen UIs

6 Zusammenfassung und Ausblick

Die in dieser Arbeit vorgestellte PAMTraM sowie die zugehörige Vorgehensweise ermöglichen die deklarative Beschreibung mehrdeutiger Abbildungen zwischen einem Quell- und einem Zielmetamodell sowie die Annotation der Abbildungen mit explizit modellierten Kontextbedingungen. Für jede Kombination aus Quell- und Zielmetamodellen ist eine Transformationsdurchführung zu implementieren, die die bedingten Abbildungen auswertet und die Zielmodelle erzeugt. Obwohl dies zunächst aufwendig ist, sollte es auf Dauer Zeit sparen, da eine Unterstützung zusätzlicher Kontexte (beispielsweise einer neuen Plattform) anschließend lediglich ein neues Kontext-Modell erfordert – die Transformationsdurchführung kann unverändert bestehen bleiben. Des Weiteren können die Abbildungsregeln von Domänenexperten definiert und als Domä-

nenwissen in der PAMTraM gespeichert werden, wohingegen die Transformationsdurchführung von einem Transformationsexperten entwickelt werden kann. Dies fördert die Trennung von Belangen. Ferner kann die Definition der bedingten mehrdeutigen Abbildungen sowie die Auswertung der Kontextbedingungen durch ein einzelnes Werkzeug realisiert werden, welches unabhängig von den verwendeten Quell-, Ziel- und Kontextmetamodellen und damit allgemeingültig ist.

Die allgemeingültige Struktur des Metamodells ermöglicht somit sowohl die Nutzung unterschiedlicher Quell- und Ziel- als auch verschiedener Kontextmetamodelle. In der Fallstudie wurde eine Nutzung von MARIA in Verbindung mit dem UsiXML-Kontextmodell demonstriert. Die Fallstudie hat unter anderem auch gezeigt, dass selbst nach Auswertung der Kontextbedingungen mehrere mögliche Zielelemente verbleiben können, was die Nutzung zusätzlicher Auswahlmechanismen notwendig macht. Daher sollte in weiterführenden Arbeiten existierende Kontextmetamodelle daraufhin untersucht werden, ob mit ihnen sinnvolle Kontextbedingungen definiert werden können. Basierend auf den Resultaten dieser Auswertung sollte ein Referenz-Kontextmetamodell sowie Referenz-Bedingungen basierend auf diesem Metamodell entwickelt werden.

Die PAMTraM unterstützt die iterative Entwicklung von UIs durch das Speichern ausgewählter, eindeutiger Abbildungen. Durch eine Wiederverwendung dieser gespeicherten Abbildungen bei einer erneuten Transformation muss ein eventuell notwendiger zusätzlicher Auswahlmechanismus (z.B. eine Nutzerinteraktion) lediglich für geänderte Modellelemente durchgeführt werden. In weiteren Arbeiten sollen umfangreiche empirische Studien zeigen, ob bzw. in welchem Umfang der vorgestellte Ansatz gegenüber herkömmlichen modellbasierten Ansätzen zu einer erhöhten Effizienz bei der UI-Entwicklung sorgen kann. Dazu muss sowohl die Implementierung von Transformationen für verschieden Metamodell-Paare als auch die Entwicklung spezifischer UIs mit Hilfe der implementierten Transformation evaluiert und mit existierenden Ansätzen bezüglich ihrer Gebrauchstauglichkeit verglichen werden.

Förderhinweis

Das IGF-Vorhaben 16606 BG der Forschungsvereinigung „Gesellschaft zur Förderung angewandter Informatik e.V. (GFaI)“ wurde über die AiF im Rahmen des Programms zur Förderung der Industriellen Gemeinschaftsforschung und -entwicklung (IGF) vom Bundesministerium für Wirtschaft und Technologie aufgrund eines Beschlusses des Deutschen Bundestages gefördert.

Literaturverzeichnis

[APP10] Aquino, N.; Panach, J. I.; Pastor, O.: A Proposal for Enhancing the UsiXML Transformation Meta-Model. In (Faure, D.; Vanderdonckt, J. Hrsg.): Proc. of the 1st International Workshop on User Interface eXtensible Markup Language. Berlin, 2010; S. 195-204.

[Bé06] Bézivin, J. et al.: Model Transformations? Transformation Models! In (Nierstrasz, O.; Whittle, J.; Harel, D.; Reggio, G. Hrsg.): Model Driven Engineering Languages and Systems. Volume 4199. Springer, Heidelberg, 2006; S. 440-453.
[Bl09] Blumendorf, M.: Multimodal Interaction in Smart Environments: A Model-based Runtime System for Ubiquitous User Interfaces. Dissertation. Berlin, 2009.
[Ca03] Calvary, G. et Al.: A Unifying Reference Framework for multi-target user interfaces. In: Interacting with Computers 15(3), 2003; S. 289-308.
[De13] DevBoost: EMFText, http://www.emftext.org, 2013.
[Ec12] The Eclipse Foundation: ATL – a model transformation technology. URL: http://www.eclipse.org/atl/, 2012.
[Ec13a] Eclipse.org: Eclipse Modeling Framework Project. URL: http://www.eclipse.org/modeling/emf, 2013.
[Ec13b] Eclipse.org. Epsilon. URL: http://www.eclipse.org/epsilon/, 2013.
[GPR06] Gruhn, V.; Pieper, D.; Röttgers, C.: MDA - Effektives Softwareengineering mit UML2 und Eclipse. Springer, Heidelberg, 2006.
[Ha11] Hager, H. et al.: Modelltransformationen in nutzerzentrierten Entwurfsprozessen der Automation. In: i-com, 10(3), 2011; S. 19-25.
[He11] Hennig, S. et al.: User Driven Evolution of User Interface Models – The FLEPR Approach. In (Campos, P.; Graham, N.; Jorge, J.; Nunes, N.; Palanque, P.; Winckler, M. Hrsg.): Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III. LNCS, Vol. 6948, Springer, Heidelberg, 2011; S. 610-627.
[Li05] Limbourg, Q. et al.: USIXML: A Language Supporting Multi-path Development of User Interfaces. In (Bastide, R.; Palanque, P.; Roth, J. Hrsg.): Engineering Human Computer Interaction and Interactive Systems. Vol. 3425, Springer, Heidelberg, 2005; S. 200-220.
[Li11] Lisai, M. et al.: Supporting transformations across user interface descriptions at various abstraction levels. In (Campos, P.; Graham, N.; Jorge, J.; Nunes, N.; Palanque, P.; Winckler, M. Hrsg.): Human-Computer Interaction – INTERACT 2011. Volume 6949. Springer, Heidelberg, 2011; S. 608-611.
[Ob11] Object Management Group: Meta Object Facility (MOF) 2.0 Query/View/Transformation, V1.1, 2011.
[PE99] Puerta, A.; Eisenstein, J.: Towards a general computational framework for model-based interface development systems. In: Knowledge-Based Systems, 12(8), 1999; S. 433-442.
[Po12] Popp et Al.: A Transformation Engine for Model-driven UI Generation. In: Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, 2012; S. 281-286.
[PSS09] Paternò, F.; Santoro, C.; Spano, L.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. In: ACM Transactions on Computer-Human Interaction, 16(4), 2009; S. 19:1-19:30.
[SCF06] Sottet, J.-S.; Calvary, G.; Favre, J.-M.: Mapping Model: A First Step to Ensure Usability for sustaining User Interface Plasticity. In: Model Driven Development of Advanced User Interfaces (MDDAUI 2006), 2006.
[Us07] UsiXML Consortium: UsiXML, USer interface eXtensible markup language - reference manual v1.8, 2007.
[W3C99] W3C: XSL Transformations (XSLT), Version 1.0. URL: http://www.w3.org/TR/xslt, 1999.
[W3C09] W3C: Delivery context ontology. URL: http://www.w3.org/TR/2009/WD-dcontology-20090616/, 2009.