Henning Hager, Stefan Hennig, Marc Seißler, Annerose Braune

Modelltransformationen in nutzer-zentrierten Entwurfsprozessen der Automation

Modeltransformation in User-Centered Design Processes of the Automation Domain

Modellbasierte Entwicklung, nutzerzentrierte Entwicklung, Benutzungsschnittstellen, interaktive Transformation

Zusammenfassung. Die Zusammenführung modellbasierter und nutzerzentrierter Entwurfsprozesse bei der Entwicklung von Benutzungsschnittstellen kann die Gebrauchstauglichkeit der resultierenden Benutzungsschnittstellen verbessern. Aus der Kombination der Entwurfsprozesse erwachsen Anforderungen an die notwendigen Modelltransformationen. Als zentrale Herausforderung wird in diesem Beitrag die Überwindung der Abstraktionslücke zwischen abstrakten und konkreten Modellen betrachtet. Anhand einer Transformation innerhalb des nutzerzentrierten Entwicklungsprozesses der Useware zur Integration einer domänenspezifischen Sprache der Automatisierungstechnik, werden die Herausforderungen der Modelltransformationen analysiert und mit den Mitteln der modellgetriebenen Software-Entwicklung gelöst. Weiterhin wird die Unterstützung von iterativem Vorgehen in modellbasierten Entwicklungsprozessen untersucht und als Lösung das Persistant Transformation Mapping vorgeschlagen.

Summary. Joining model-based and user-centered design processes for the development of user interfaces can improve the usability of the resulting user interfaces. Due to the combination of the design processes challenges arise for the necessary model transformations. Bridging the abstraction gap between abstract and concrete models is the key challenge that is addressed by this paper. By example of a transformation within the user-centered development process to integrate a domain-specific language of the automation challenges are analyzed and solved by means of the model-driven software development. Furthermore, the support of iterative procedures in model-based development processes is examined and as a solution the Persistent Transformation Mapping is proposed.

1. Motivation

Die modellbasierte Entwicklung von Benutzungsschnittstellen (MBUID[1]) verspricht in Kombination mit dem Vorgehen der modellgetriebenen Softwareentwicklung (MDSD[2]) ein nachhaltiges Design von Visualisierungslösungen in der industriellen Automatisierungstechnik. Für die Entwicklung gebrauchstauglicher Benutzungsschnittstellen eignen sich nutzerzentrierte Entwicklungsprozesse, in denen das Vorgehen der MBUID und MDSD angewendet wird.

Innerhalb der MBUID-Gemeinschaft werden verschiedene Sichten auf Benutzungsschnittstellen definiert und Modelle für deren Beschreibung zur Verfügung gestellt.

DOI 10.1524/icom.2011.0028

Die MDSD biete Methoden und Werkzeuge, um Modelle zu definieren und zu verarbeiten. Dazu bilden domänenspezifische Sprachen (DSL[3]) Gegenstandsbereiche ab und können eine fachliche Beschreibung in Modellen ermöglichen. Die Überführung und Verarbeitung der Modelle erfolgt mit Transformationen, indem sie die Modelle schrittweise mit technischen Details anreichern bis hin zur Erzeugung des ausführbaren Quellcodes (Hennig, Koycheva und Braune, 2010) konkretisieren. Mit der DSL Movisa[4] wird dieses Vorgehen umgesetzt und es können speziell auf die Anforderungen der Automatisierungstechnik zugeschnittene Benutzungsschnittstellen entwickelt werden.

Bei Anwendung der MBUID und MDSD in nutzerzentrierten Entwicklungsprozessen entstehen Herausforderungen für die Modelltransformation aufgrund des iterativen Vorgehens. Während der gesamten Entwicklung werden die Benutzungsschnittstelle und ihre Prototypen beständig evaluiert und die Erfahrungen werden unmittelbar in die Entwicklungsphasen zurückgeführt, sodass sich ein iterativer, nutzerzentrierter Prozess ergibt (siehe Bild 1). Der Useware-Entwicklungsprozess ist ein solcher nutzerzentrierter Prozess für die Entwicklung von allgemeinen Benutzungsschnittstellen.

In diesem Beitrag werden die Anforderungen an Modelltransformationen in

1 Model-Based User Interface Development
2 Model-Driven Software Development
3 Domain-Specific Language
4 Model Driven Development of Visualization Solutions in Industrial Automation

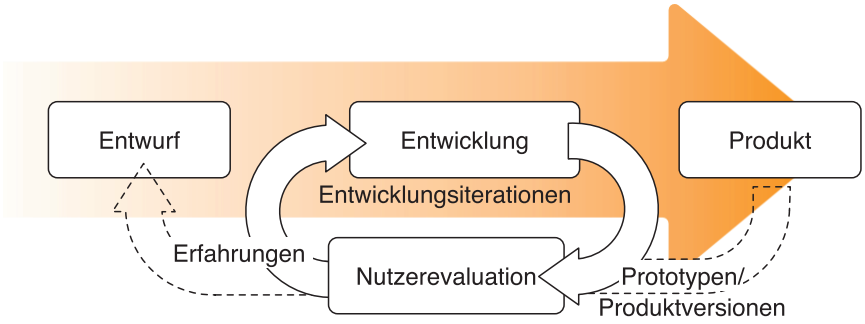

Bild 1: Iterative Entwicklung mit paralleler Nutzerevaluation.

nutzerzentrierten Entwicklungsprozessen am Beispiel der Transformation von Movisa zu DISL[5] (eines der Modelle des Useware-Entwicklungsprozesses) dargestellt. Die vorgestellte Transformation ist ein möglicher Lösungsansatz für die allgemeinen Anforderungen und integriert Movisa in den Useware-Entwicklungsprozess.

Der Beitrag gliedert sich wie folgt: Zunächst werden in Abschnitt 2 die benötigten Grundlagen für Modelltransformationen erläutert und die beteiligten Modelle vorgestellt. In Abschnitt 3 werden die Anforderungen an die Transformation erarbeitet und in Abschnitt 4 durch unseren Ansatz realisiert. Zur Demonstration des Ansatzes wird in Abschnitt 5 eine Fallstudie präsentiert. Der Beitrag schließt mit der Auswertung in Abschnitt 6.

2. Grundlagen

Die Transformation von Movisa zu der im Useware-Entwicklungsprozess verwendeten DISL-Beschreibungssprache erfolgt im Rahmen der MDSD. Dazu werden Modelltransformationen kurz beschrieben und die für die Problemstellung relevanten Eigenschaften aufgegriffen. Anschließend wird die DSL Movisa sowie der nutzerzentrierte Useware-Entwicklungsprozess mit den verwendeten Modellen vorgestellt.

2.1 Transformationen in der MDSD

In der MDSD werden formale Modelle – Modelle, die einem wiederum formalen Metamodell unterliegen – durch Transformationen ineinander überführt. Die Überführung der konkreten Quell- und Zielmodellinstanzen basiert in Transformationen auf Abbildungen (Mappings), die allen Elementen des Quellmetamodells Elemente des Zielmetamodells zuordnen. Neben eindeutigen Mappings treten auch mehrdeutige auf, d.h. dass Elementen des Quellmetamodells verschiedene Elemente des Zielmetamodells zugeordnet werden können. Als Lösungsmöglichkeiten der mehrdeutigen Zuordnungen sind u.a. folgende Varianten möglich:

[Standardzuordnung:] Die Mehrdeutigkeiten werden während der Transformation nicht berücksichtigt, stattdessen werden für mehrdeutige Zuordnungen Standardelemente verwendet. Transformationen der MBUID verwenden aufgrund der leichten Realisierbarkeit häufig diesen Ansatz, z.B. in MOBI-D (Puerta und Eisenstein, 1999), MARIA (Paterno, Santoro und Spano, 2009) und DynaMo-AiD (Clerckx, Luyten und Coninx 2004).

[Annotation des Quellmodells:] Vor jeder Transformation wird das Quellmodell mit zusätzlichen Informationen (Annotationen) bzgl. zu verwendender Mappings angereichert, sodass die Transformation alle Elemente eindeutig zuordnen kann (Hennig et al., 2011).

[Externe Mappings:] Im Sinne der MDSD wird Funktionalität von Technologie getrennt und die Transformation verwendet externe Quellen (z.B. Mapping-Modelle) für die Elementzuordnung (Clerckx, Luyten und Coninx, 2004).

[Interaktive Transformation:] Die Transformation ist zur Laufzeit auf den Entwickler angewiesen, der das zu realisierende Mapping spezifiziert.

Als charakteristische Eigenschaften für die entwickelte Transformation sind die Folgenden von Bedeutung:

[Grad der Automatisierung:] Transformationen sind automatisch und damit selbständig lauffähig oder semi-automatisch, d.h. dass externe Informationen für die Durchführung notwendig sind (Siikarla und Systa, 2008).

[Verfolgbarkeit (Traceability):] Die Reproduktion von Transformationen bzw. deren Ergebnisse sind nur möglich, wenn sie verfolgbar (Czarnecki und Helsen, 2003) sind; dazu müssen die Beziehungen zwischen den Quell- und Zielelementen auch nach der Transformation erhalten bleiben.

2.2 Movisa

Typische Benutzungsschnittstellen in der Automatisierungstechnik sind u.a. Prozessvisualisierungen zur Überwachung und Steuerung von Anlagen, die spezielle Anforderungen an den Entwurf stellen. Die mangelhafte Unterstützung der domänenspezifischen Anforderungen (Braune und Hennig, 2007) etablierter Beschreibungssprachen, wie UIML[6] (Helms et al, 2008), haben die Entwicklung von Movisa (Hennig und Braune, 2011) motiviert.

Mit der DSL Movisa lassen sich Benutzungsschnittstellen der Automatisierungstechnik plattformunabhängig beschreiben. Es werden Panels und Interaktionsobjekte (UIComponents) für die graphische Darstellung der Benutzungsschnittstelle bereitgestellt. Die elementaren UIComponents (ElementaryUIComponent) verfügen neben Standardelementen wie Buttons, Bilder etc. zusätzlich über die domänenspezifischen Elemente Trendgraphiken (Trend) und Analogmeter (Gauge) der Automatisierungstechnik (siehe Tabelle 1). Mit ComplexUIComponents können UIComponents zu Hierarchien aggregiert werden.

Die plattformunabhängige Beschreibung der Benutzungsschnittstelle im Movisa-Modell werden mit Modell-zu-Text-Transformationen in die technische Umsetzung für die ausführende Plattform (Software-Frameworks etc.) überführt.

2.3 Useware

Die Useware (Zühlke, 2004) eines Systems umfasst alle Hard- und Software-Komponenten, die dem Nutzer für die Bedienung

5 Dialog and Interface Specification Language

6 User Interface Markup Language

UIComponent	Beschreibung
AlarmControl	Informiert über Alarme des Prozesses
Button	Ausführung von Funktionen des Visualisierungssystems
CheckBox	Auswahl von beliebigen Werten bzw. Elementen
Gauge	Darstellung von Werten auf einer Skala mit Zeiger
Image	Anzeige von Bildern
Input	Eingabefeld von beliebigen Werten
DropDown	Auswahl eines einzelnen Wertes
RadioButton	Auswahl eines einzelnen Wertes
Slider	Stufenlose Auswahl eines Wertes
TextLabel	Anzeige von Text
Trend	Darstellung eines Wertes in Abhängigkeit der Zeit

Tabelle 1: Movisa UIComponents.

zur Verfügung stehen. Zur Entwicklung dieser Komponenten wird der nutzerzentrierte Useware-Entwicklungsprozess (siehe Bild 2) verwendet, der sich in vier Entwicklungsphasen gliedert, in den die Benutzungsschnittstelle mit aufeinander aufbauenden Konkretisierungsstufen beschrieben wird. Die parallele Nutzerevaluation überprüft die Zwischenergebnisse der Entwicklungsphasen auf Erfüllung der Nutzeranforderungen. Durch unmittelbare Rückführung der Evaluationsergebnisse in die Entwicklungsphasen entstehen Benutzungsschnittstellen mit hoher Gebrauchstauglichkeit.

In der ersten Entwicklungsphase Analyse wird die Datenerhebung bzgl. der Systemnutzer, deren Aufgaben und Arbeitsweisen vorgenommen. Die verwendeten Technologien und Werkzeuge (Meixner, Görlich und Schäfer, 2008) der Analyse dienen der Erfassung und Strukturierung der Nutzeraufgaben. Die folgenden Phasen leiten basierend auf den Daten eine an die Anforderungen abgestimmte Benutzungsschnittstelle her.

Die Strukturgestaltungsphase erstellt zunächst mit UseML[7] das Benutzungsmodell (Meixner, 2010), das die Struktur der Benutzungsschnittstelle mit Nutzeraufgaben und deren zeitlichen Beziehungen darstellt. Aus dem Benutzungsmodell wird außerdem in der Strukturgestaltungsphase das Dialogmodell abgeleitet und mit DISL (Schäfer, 2007) beschrieben, das in diesem Beitrag als Quellmodell für die Transformation nach Movisa dient.

Das modalitätenunabhängige DISL-Modell beschreibt die statische Struktur und die dynamischen Eigenschaften der Benutzungsschnittstelle. Die Struktur erfasst den Aufbau der Benutzungsschnittstelle mit Zuständen, die in der Dynamikbeschreibung genutzt werden. Zustände beinhalten abstrakte Interaktionsobjekte (Widget; siehe Tabelle 2) mit spezifischen Eigenschaften. Die Dynamik der Benutzungsschnittstelle wird in einer Zustandsmaschine abgebildet, die auf Ereignisse der Widgets reagiert.

Der nächste Schritt der Konkretisierung findet in der Bediensystemgestaltungsphase statt, in der die Darstellung und das damit verbundene Layout festgelegt wird. Derzeit unterstützt die Useware UIML zur Darstellungsbeschreibung, deren Interaktionselemente jedoch nicht den Anforderungen der Automatisierungstechnik genügen (siehe Abschnitt 2.2).

In der letzten Phase des Entwicklungsprozesses, der Realisierung, findet die Umsetzung in Quellcode statt. Die für die nutzerzentrierte Entwicklung wesentliche Evaluation verwendet die Prototypen der verschiedenen Entwurfsphasen und Nutzer bewerten diese. Alle gesammelten praktischen Erfahrungen der Benutzer werden in die jeweiligen Entwicklungsphasen zurückgeführt und eine neue Entwicklungsiteration wird eingeleitet. Auf diese Weise erhalten die Entwickler qualitative Auswertungen der entworfenen Benutzungsschnittstelle, mit denen Fehler und mögliche Verbesserungen frühzeitig erkannt werden können.

3. Anforderungen an die Transformation

Die Transformation von DISL nach Movisa basiert auf Abbildung der DISL-Widgets auf die UIComponents von Movisa. Tabelle 3 stellt die Mappings der Elemente dar und zeigt, dass sich nur das Gruppierungselement widgetlist eindeutig einer UIComponent (ComplexUIComponent) zuordnen lässt. Bei den übrigen Elementen liegt ein Mehrdeutigkeitsproblem vor: Die Zuordnungen zwischen den DISL- und Movisa-Elementen sind nicht eindeutig. Eine Herausforderung der Transformation ist somit der zielgerichtete Umgang mit den mehrdeutigen Zuordnungen zur Erzeugung des Movisa-Modells. In der MBUID ist diese Problemstellung als Abstraktionslücke bekannt und tritt generell bei der Überführung von Modellen unterschiedlichen Abstraktionsgrades auf (Puerta, 1999).

Das Vorgehen des Useware-Entwicklungsprozess ist iterativ, sodass Änderungen am ursprünglichen DISL-Modell (z.B. aufgrund von Ergebnissen aus der Evaluation) im Movisa-Modell berücksichtigt werden müssen. Gleichzeitig müssen die unveränderten Modellteile vollständig erhalten bleiben, d.h. dass die Lösungen der Mehrdeutigkeiten reproduziert werden müssen. Damit besteht eine weitere Herausforderung der Transformation in der Reproduktion der gelösten Mehrdeutigkeiten, um eine konsistente und durchgängige Entwicklung zu sichern. Auch

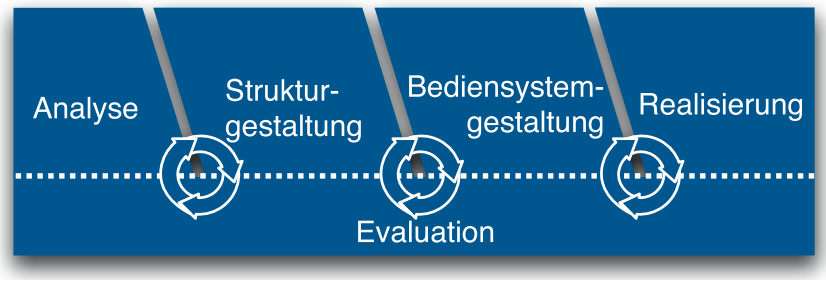

Bild 2: Die Phasen des Useware-Entwicklungsprozesses.

7 Useware Markup Language

Widget	Beschreibung
variablefield	Ausgabe einzelner Werte
textfield	Textausgabe
variablebox	einfache Werteeingabe
textbox	Texteingabe
command	Ausführung einer Aktion/Funktion
confirmation	Ausführung einer Aktion/Funktion nach Bestätigung
choicegroup	Gruppierung und Auswahl von Widgets
widgetlist	Reine Gruppierung von Widgets

Tabelle 2: Übersicht und Verwendung der DISL-Widgets nach (Meixner, 2010).

		DISL Widgets							
		variablefield	textfield	variablebox	textbox	command	confirmation	choicegroup	widgetlist
Movisa UICompsonents	AlarmControl	✔	✔			✔	✔		
	Button	✔	✔			✔	✔		
	CheckBox	✔	✔			✔	✔	✔	
	Gauge	✔	✔			✔	✔		
	Image	✔	✔			✔	✔		
	Input	✔	✔	✔	✔	✔	✔		
	DropDown	✔	✔			✔	✔	✔	
	RadioButton	✔	✔			✔	✔	✔	
	Slider	✔	✔			✔	✔		
	TextLabel	✔	✔			✔	✔		
	Trend	✔	✔			✔	✔		
	ComplexUIComponent	✔	✔	✔	✔	✔	✔	✔	✔

Tabelle 3: Zuordnung der DISL- und Movisa-Elemente.

diese Anforderung ist nicht auf die verwendeten Modelle beschränkt und tritt generell bei modellbasierter iterativer Entwicklung auf.

4. Realisierung der Transformation

Zur Lösung der Anforderungen wurden in Abschnitt 2.1 Methoden und Eigenschaften von Transformationen vorgestellt. Die Verwendung von Standardzuordnungen mit statischer Festlegung einer konkreten Zuordnung ohne Berücksichtigung der Mehrdeutigkeiten ist für den Umgang mit mehrdeutigen Elementzuordnungen ungeeignet, da nach jeder Transformation manuelle Nacharbeiten am generierten Modell zur Anpassung der erzeugten Standardelemente notwendig sind. Diese Anpassungen gehen bei erneuter Transformation verloren.

Bei Annotierung der Quellmodelle erfolgen manuellen Eingriffe vor der Transformation; zwar bleiben die gewählten Mappings bei erneuter Transformation bestehen, doch die Eingriffe auf (Meta-) Modellebene erfordern spezifisches Fachwissen über die Metamodelle und deren Transformation.

Daher werden die übrigen Methoden – interaktive Transformation und externe Mappings – für die Transformation des DISL-Modells genutzt.

Die Modelltransformation erfolgt semi-automatisch, da der Entwickler für jedes Widget, das nach Tabelle 3 über kein eindeutiges Mapping verfügt, aufgefordert wird die zu erzeugenden Movisa-UI-Component zu wählen. Dabei werden die Optionen nach Tabelle 3 eingeschränkt, sodass nur gültige Mappings ausgewählt werden können.

In Abschnitt 2.1 wird die Verfolgbarkeit von Transformationen zur Unterstützung der iterativen Entwicklung gefordert und wird in unserem Lösungsvorschlag durch das Persistant Transformation Mapping (Petmap) realisiert. Auf diese Weise können die bereits in den vorangegangenen Iterationen bzw. Transformationen spezifizierten Mappings wiederverwendet werden und der Entwickler muss nur Mappings für neu hinzugekommene Elemente festlegen. Die Petmap speichert nach der Elementauswahl des Entwicklers das eindeutige Mapping für die konkreten Elemente. Bei erneuten Transformationen dient die Petmap als externes Mapping-Modell und die Transformation erfolgt für bekannte Elemente ohne erneute Entwicklerinteraktion.

Die Petmap ist nicht auf die Modelle der Useware oder Movisa festgelegt, es lassen sich Mappings zwischen Elementen aus beliebigen Metamodellen beschreiben. Der Aufbau des Petmap-Modells gliedert sich in drei Teile zur Beschreibung bzw. Referenzierung der Metamodelle, Modelle und der Transformationen mit den gewählten Mappings.

Die vollständige Realisierung fasst Abbildung 6 zusammen: Bei erstmaliger Transformation (1) sind für alle mehrdeutigen Elemente des DISL-Modells Entwicklerinteraktionen notwendig, die in der Petmap gespeichert werden. Innerhalb einer weiteren Entwicklungsiteration werden Veränderungen im DISL-Modell (2) vorgenommen und durch eine erneute Transformation (3) in das Movisa Modell weitergereicht. Für die erneute Transformation werden die gespeicherten Mappings aus der Petmap wiederverwendet und die Entwicklerinteraktionen reduzieren sich auf neue DISL-Elemente. Als Ergebnis entsteht ein Replikat des Modells mit Berücksichtigung der Veränderungen aus dem DISL-Modell.

5. Fallstudie

Die Anwendung der entwickelten Transformation erfolgt anhand einer Prozessvisualisierung der Verfahrenstechnik beispielhaft für die vereinfachte Überwachung des Füllstandes eines Chemiereaktors. Der Nutzer soll für die Überwachung u.a. die Pumpendrehzahl überprüfen, den Füllstand ablesen und die Zuflüsse regulieren. Da alle Aufgaben gleichzeitig verfügbar sein müssen, beinhaltet das

DISL-Modell (siehe Bild 4 (1)) einen eigenen Zustand (interface) für die Füllstandsüberwachung mit zwei variablefields für die Anzeige der Werte (vgl. Tabelle 2) und ein command zur Regulierung.

Die entwickelte DISL-Movisa-Transformation legt für jedes DISL-interface ein Movisa-Panel an. Bei der Transformation der DISL-Widgets kann das variablefield nach Tabelle 3 zwölf verschiedenen Movisa-UIComponents zugeordnet werden, daher wird der Entwickler zur Auswahl einer entsprechenden UIComponent aufgefordert (siehe Bild 4 (2)). Für das Überwachen des Füllstandes wählt der Entwickler ein Movisa-Trend, für die Zuflussregulierung einen Button und für die Pumpendrehzahl ein Textlabel. Damit wurden für das textfield Widget zwei unterschiedliche Movisa UIComponents gewählt.

Unmittelbar nach der Elementauswahl werden die Mappings in der Petmap gespeichert (siehe Bild 4 (3.1)). Dort wird dem textfield zur Überwachung des Füllstands („fuellevel") der Movisa Trend („View__fuellevel") zugeordnet (siehe Bild 4 (3.2)). Als Ergebnis der Transformation entsteht ein Movisa-Modell (siehe Bild 4 (4)), mit den gewählten UIComponents. Aus dem DISL-Modell werden Information für Namen und Interaktionsverhalten abgeleitet; die Anordnung der Komponenten legt der Entwickler manuell im Movisa-Editor fest.

Im Verlauf der weiteren Entwicklung und Evaluation der Benutzungsschnittstelle stellt sich heraus, dass die Regulierung des Zuflusses mit einem command Widget („start_pump") unzureichend spezifiziert ist. Daher wird in einer weiteren Entwicklungsiteration im DISL-Modell ein zusätzliches command („stop_pump") angelegt. Zu diesem Zeitpunkt sind das DISL- und Movisa-Modell inkonsistent, da das neue command Widget im Movisa-Modell nicht berücksichtigt ist. Durch erneute Transformation des DISL-Modells wird die Konsistenz wiederhergestellt. Im Unterschied zu der ersten Durchführung verwendet die Transformation die in der Petmap hinterlegten Elementzuordnungen. Somit wird der Entwickler nur für das neue command widget („stop_pump") zur Auswahl aufgefordert – die spezifischen Zuordnungen der anderen Widgets werden vollständig reproduziert. Auf diese Weise sichert die Petmap in Kombination mit der interaktiven Transformation die durchgängige und iterative Entwicklung.

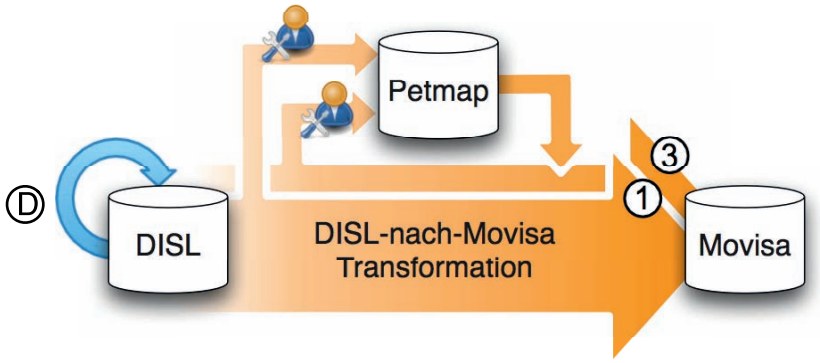

Bild 3: Verwendung der interaktiven Transformation und der Petmap.

Der von uns vorgeschlagene interaktive Ansatz verzichtet, im Gegensatz zu den genannten Arbeiten, auf detailliertes Fachwissen der Entwickler über (Meta-) Modelle sowie Transformationen und erspart die manuelle Korrektur des erzeugten Modells. Die Verwendung des Ansatzes mit Ergänzung der Petmap für die iterative Entwicklung stellt einen zusätzlichen Mehrwert gegenüber den anderen Arbeiten dar.

6. Auswertung und zukünftige Herausforderungen

Am Beispiel der Transformation von DISL nach Movisa wurden die Anforderungen an Transformationen in nutzerzentrierten Entwicklungsprozessen herausgearbeitet. Die Überbrückung der Abstraktionslücke bei Modellen unterschiedlichen Abstraktionsgrades und die Unterstützung iterativer Entwicklung wurden als Herausforderungen identifiziert. Als Lösung wurde ein Ansatz mit interaktiver Transformation in Ergänzung mit der Petmap, einem Mittel zur persistenten Speicherung von Mappings, vorgestellt.

Die Interaktionen beziehen den Entwickler der Benutzungsschnittstelle direkt in die Transformation ein, sodass erwartungskonforme Elemente erzeugt werden und der manuelle Austausch zur Korrektur der erzeugten Elemente überflüssig wird. Die adäquate Wahl der Elemente ist bewusst dem Entwickler überlassen, der bei Verwendung von Movisa ein Experte der Automatisierungstechnik ist und somit über das notwendige Domänenfachwissen verfügt.

Interaktive Transformationen sind ein vielversprechender Ansatz zur Überwindung der Abstraktionslücke, da keine Fachkenntnisse der Benutzungsschnittstellenentwickler bzgl. Transformationen notwendig sind. Die Anwendung ist nicht auf die hier verwendeten Metamodelle oder die MBUID beschränkt, sondern ist ein allgemeingültiger Lösungsansatz für Modelltransformationen mit mehrdeutigen Elementzuordnungen.

Die iterative Entwicklung wird durch Verwendung der Petmap unterstützt. In dem Petmap-Modell wird der Transformationsverlaufs mit den gewählten Elementzuordnungen gespeichert und somit die Wiederverwendung der Zuordnungen in weiteren Transformationen ermöglicht. Im Gegensatz zu Transformationsmodellen, wie z.B. bei UsiXML (Limbourg und Vanderdonckt, 2004) oder Transformation Templates (Aquino, 2009), werden die Mappings nicht auf Metamodellebene beschrieben, sondern beziehen sich konkret auf Elementinstanzen (siehe Bild 4 (3.1)). Erst durch dieses Vorgehen ist die Wiederverwendung unterschiedlicher Lösungen der mehrdeutigen Zuordnungen möglich, z.B. wurde in der Fallstudie (siehe Abschnitt 5) das DISL Widget textfield auf zwei unterschiedliche Movisa UIComponents (Textlabel und Trend) abgebildet.

Zurzeit unterstützt der präsentierte Lösungsansatz (siehe Bild 4) nur die iterative Entwicklung ausgehend vom übergeordneten DISL-Modell; der Useware-Entwicklungsprozess sieht aber Nutzerevaluationen in allen Entwicklungsphasen vor. Für eine vollständige iterative Entwicklung im Sinne der Definition müssen auch Veränderungen auf Movisa-Ebene gemäß den

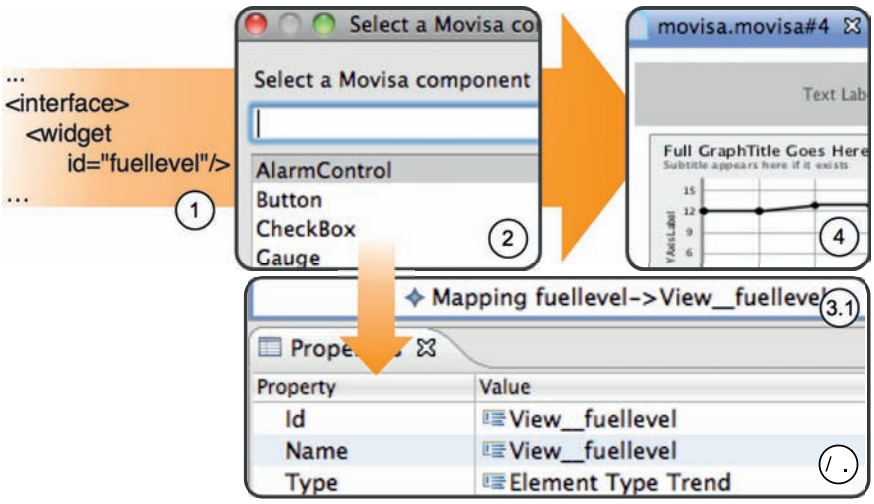

Bild 4: Ablauf der Transformation von DISL nach Movisa.

Ergebnissen der Nutzerevaluation vorgenommen werden können. Auch auf diese Weise können Modellinkonsistenzen (zum übergeordneten DISL-Modell) entstehen. Die Petmap ist mit den Mappings der durchgeführten Transformationen ein mögliches Mittel zur Erkennung von Inkonsistenzen zwischen Modellen. Es ist daher zu untersuchen, inwieweit die Mappings der Petmap genutzt werden können, um direkt auf Movisa-Ebene Nutzerfeedback zu berücksichtigen. Mit entsprechenden Mechanismen (z.B. Wizards) in den Movisa-Editoren können kleine Veränderungen konform zu den Mappings (Wechsel der UIComponents innerhalb der Zuordnungsoptionen nach Tabelle 3) ermöglicht werden. Dieses Vorgehen schränkt jedoch die Möglichkeiten der Benutzungsschnittstellengestaltung erheblich ein. Alternativ können beliebige Veränderungen und somit kurzzeitige Modellinkonsistenzen erlaubt werden, die durch Rücktransformationen und mit Verwendung einer entsprechenden Logik aufgelöst werden. Durch dieses weitere Vorgehen ist es möglich die iterative Entwicklung zu einem Roundtrip-Engineering für modellbasierte Entwicklung von Benutzungsschnittstellen zu überführen.

Eine weitere Anwendung der Petmap ist die Nutzung für wissensbasierte Systeme mit Ableitung von übergeordneten Regeln zur Gestaltung von Benutzungsschnittstellen. Auf Basis der gewählten Elementzuordnungen der Petmap können Metriken für „gute" Mappings ermittelt werden und den Entwickler bei der Auswahl unterstützen.

Literatur

Aquino, N.: Adding flexibility in the model-driven engineering of user interfaces. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing-systems, 2009.

Braune, A.; Henning, St. Technologieunabhängiges HMI-Engineering für technische Prozesse. In VDI-Berichte GMA-Kongress in Baden-Baden, 2007.

Clerckx, T.; Luyten, K.; Coninx, K.: The mapping problem back and forth: customizing dynamic models while preserving consistency. In Proceedings of the 3rd annual conference on Task models and diagrams, 2004.

Czarnecki, K; Helsen, S.: Classification of model transformation approaches, in Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, 2003.

Helms, J; Schaefer, R; Luyten, K.; Vermeulen, J.; Abrams, M.; User Interface Markup Language (UIML) Version 4.0. OASIS Standard Specification, OASIS, 2008.

Hennig, St.; Braune, A.: Sustainable Visualization Solutions in Industrial Automation with Movisa – a Case Study, IEEE Conference on Industrial Informatics (INDIN), Lisbon, 2011.

Hennig, St.; Koycheva, E; Braune,A: Domänenspezifische Sprachen und deren Bedeutung für die modellgetriebene Softwareentwicklung in der Automatisierungstechnik. VDI-Berichte/VDI-Tagungsbände 2092, AUTOMATION 2010, 415–420

Hennig, St.; Van den Bergh, J.; Luyten K., Braune, A.: User Driven Evolution of User Interface Models – The FLEPR Approach. INTERACT (3) 2011, 610–627

Limbourg, Q ; Vanderdonckt, J.: Addressing the Mapping Problem in User Interface Design with UsiXML. In Proceedings of the tenth ACM international conference on Multimedia, 2004.

Meixner, G.; Görlich, D.; Schäfer, R.: Unterstützung des Useware-Engineering Prozesses durch den Einsatz einer modellbasierten Werkzeugkette. VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik, 2008.

Meixner, G.: Entwicklung einer modellbasierten Architektur für multimodale Benutzungsschnittstellen. Fortschritt-Berichte pak 21 Technische Universität Kaiserslautern, Kaiserslautern, 2010.

Paterno, F.; Santoro, C.; Spano, L. D.: MARIA: A Universal, Declarative, Multiple Abstraction-Level Language for Service-Oriented Applications in Ubiquitous Environments, ACM Transactions on Computer-Human Interaction 16 (2009): 1–30.

Puerta, A.; Eisenstein, J.: Towards a general computational framework for model-based interface development systems. Knowledge-Based Systems, 1999.

Robbie Schäfer. Model-based Development of Multimodal and Multi-device User Interfaces in Context-aware Environments (C-LAB Publication). Dissertation, Universität Paderborn, 2007.

Siikarla, M. P; Systa, T. J.: Decision reuse in an interactive model transformation. in Software Maintenance and Reengineering, 2008, 123–132

Zühlke, D.. Useware-Engineering für technische Systeme. Springer, 2004.


1


2


3


4

1 Dipl.-Ing. Henning Hager ist Doktorand am Institut für Automatisierungstechnik, Technische Universität Dresden. In seiner Promotion ist er auf dem Wissenschaftsgebiet der modellgetriebenen Entwicklung von Benutzungsschnittstellen mit Anwendung in der industriellen Automatisierungstechnik tätig. Schwerpunkt seiner Arbeit sind Modelltransformationen.
E-Mail: henning.hager@tu-dresden.de

2 Dipl.-Ing. Stefan Hennig ist Doktorand am Institut für Automatisierungstechnik, Technische Universität Dresden. Im Rahmen seiner aus den Mitteln des ESF geförderten Landesinnovationspromotion beschäftigt er sich mit der modellgetriebenen Entwicklung von Benutzungsschnittstellen zur Prozessvisualisierung in der industriellen Automatisierungstechnik.
E-Mail: stefan.hennig@tu-dresden.de
Internet:
http://www.et.tu-dresden.de/ifa/index.php?id=731

3 Dipl.-Inf. Marc Seißler studierte Angewandte Informatik an der TU Kaiserslautern und erhielt 2009 sein Diplom für die Einführung eines semiautomatischen Transformationsprozesses von Aufgabenmodellen in Dialogmodelle. Seit Juli 2009 ist Herr Seißler Wissenschaftlicher Mitarbeiter an der TU Kaiserslautern und forscht an der modellbasierten Entwicklung von laufzeitadaptiven Benutzungsschnittstellen für ambient-intelligente Produktionsumgebungen.

4 PD Dr.-Ing. Annerose Braune ist Mitarbeiterin am Institut für Automatisierungstechnik, Technische Universität Dresden. Sie leitet die Arbeitsgruppe Teleautomation, in der die Anwendung moderner Informationstechnologien in der Automatisierungstechnik untersucht werden.
E-Mail: annerose.braune@tu-dresden.de
Internet:
http://www.et.tu-dresden.de/ifa/index.php?id=450a