Automatische Generierung voll funktionsfähiger mobiler Bediensoftware aus Benutzungs- und Funktionsmodellen

Kai Breiner[1], Daniel Görlich[2], Oliver Maschino[1], und Gerrit Meixner[2],

[1]AG Software Engineering, Fachbereich Informatik
TU Kaiserslautern
67663 Kaiserslautern
{breiner, maschino}@cs.uni-kl.de
[2] Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Zentrum für Mensch-Maschine-Interaktion (IFS-ZMMI)
Trippstadter Str. 122, 67663 Kaiserslautern
{Gerrit.Meixner, Daniel.Goerlich}@DFKI.de

Abstract: Die zunehmende Anzahl und Komplexität technischer Geräte in Produktionsanlagen erfordert ein Umdenken ihrer Bedienung, hin zu einer flexiblen, nutzerorientierten und weitestgehend automatischen Generierung und Adaption von Bediensoftware. An dafür benötigten, nutzerzentrierten Modellierungssprachen mangelte es bisher, doch für die *SmartFactory*^{KL} wurde eine solche Sprache entwickelt, getestet und demonstriert. Dieses Paper beschreibt, wie im Forschungsprojekt „GaBi" Benutzungs- und Funktionsmodelle verknüpft werden, um voll funktionsfähige Bedienschnittstellen zur Ansteuerung verschiedenartiger Anlagenkomponenten und Feldgeräte unterschiedlicher Hersteller zu generieren.

1 Einleitung

Zahlreiche Zukunftsvisionen gehen von hochgradig instrumentierten Umgebungen aus, in der zahllose Computer, eingebettete Geräte und Smart Items vernetzt sind und ihre Nutzer in intelligenter Weise unterstützen (Ambient Intelligence [AM03]). Eine Vielzahl von Menschen bewegen sich täglich in hochtechnisierten Umgebungen wie bspw. Fabrik- und Produktionsanlagen. Die Anzahl der dort eingesetzten Geräte, der Grad ihrer Vernetzung und die zunehmende Notwendigkeit zur schnellen Umstrukturierung solcher Anlagen und ihrer Prozesse, konfrontieren die dort arbeitenden Menschen mit einer immensen Komplexität und überfordern sie nicht selten.

Gerade im Hinblick auf die Entwicklung der Bediensysteme all jener stationären und mobilen Geräte muss deshalb ein Umdenken stattfinden und eine Lösung dieses Komplexitätsproblems gefunden werden. Zwar gibt es schon seit Jahrzehnten entsprechende Versuche, Geräte bspw. mit Selbstbeschreibungsfähigkeiten auszustatten (vgl. Electronic Device Description Language EDDL [IEC06]) und die Gestaltung ihrer Bediensysteme durch entsprechende Normen zu vereinheitlichen [VDI00], doch fehlt bislang ein formaler Ansatz zur Spezifikation von Bedienschnittstellen, der den Nutzer in das Zentrum der

Entwicklung rückt. Zwar würde sich hier ein modellbasierter Ansatz anbieten, um kontextbasierte Software-Generierung oder gar -Adaption zu ermöglichen, doch mangelt es ebenfalls an geeigneten, nutzerzentrierten Modellen bzw. Modellierungssprachen um einen vollautomatischen Prozess gewährleisten zu können.

Ausgehend von einer Modellierungssprache für Raumbasierte Benutzungsmodelle (Abschnitt 2) und einem entsprechenden Funktions- und Kommunikationsmodell (Abschnitt 3) wurde ein Interpreter entwickelt, der sich anhand von diesen Modellen dynamisch und kontextsensitiv an die jeweilige Produktionsumgebung anpassen kann (Abschnitt 4). Dieser Prototyp wurde mit einem Raumbasierten Benutzungsmodell der intelligenten Fabrik der Zukunft, der *SmartFactory*^{KL}, installiert. Das Modell erlaubt die Ansteuerung von etwa zwei Dutzend verschiedenartiger Geräte unterschiedlicher Hersteller. Die generierte, graphische Bedienoberfläche (GUI) ist nutzer- und aufgabengerecht, aber stilistisch recht einfach und wird im weiteren Verlauf des GaBi-Projekts optimiert werden (Zusammenfassung und Ausblick in Abschnitt 5).

2 Benutzungsmodelle in useML

Die Useware Markup Language (useML) [MGS08] ist eine aufgabenorientierte Modellierungssprache, die komplexe Aufgaben als so genannte Benutzungsobjekte repräsentiert und diese in immer kleinere bis hin zu elementaren Teilaufgaben bzw. Benutzungsobjekten unterteilt. UseML kennt fünf verschiedene Typen elementarer Benutzungsobjekte, die direkt auf Interaktionselemente (widgets) einer Zielplattform abgebildet werden könnten. Mit useML lassen sich also Benutzungsmodelle formalisieren, die zunächst unabhängig von Hardware- oder Software-Plattform die Aufgaben und Vorgehensweise aller potenziellen Benutzer bei der Bedienung eines Gerätes strukturieren. Ursprünglich für die Entwicklung von Einzelgeräten und Gerätefamilien gedacht, wurde useML so weit überarbeitet, dass es nunmehr ganze räumliche Strukturen mit allen darin enthaltenen Geräteverbünden und deren Benutzungsmodellen in Abhängigkeit von Interaktionszonen und Kontextelementen wie dem Aufenthaltsort des Nutzers repräsentieren kann [Br09a]. Solche Raumbasierte Benutzungsmodelle können also ganze instrumentierte Umgebungen wie bspw. Produktionsumgebungen beschreiben [Br09b]. Im Folgenden soll vor allem das für diese Domäne entwickelte Funktionsmodell vorgestellt werden (Abschnitt 3), dass eine Verknüpfung von Nutzeraufgaben und funkbasierter Datenkommunikation über bereits bestehende Kommunikationsplattformen/-protokolle erlaubt.

3 Funktionsmodell

Gerade in verteilten Umgebungen, wie es in der *SmartFactory*^{KL} der Fall ist, ist es wichtig, um eine funktionsfähige Benutzungsschnittstelle zur Laufzeit generieren zu können, eine detaillierte Beschreibung der Anbindung der auszuführenden Applikationslogik mit in die Modelle zu integrieren. Die meisten Benutzungsschnittstellenbeschreibungssprachen beschränken sich z.B. auf die grafische Präsentation; Ansätze, welche aus Be-

schreibungen Benutzungsschnittstellen generieren, sind oft nur semi-automatisch, so dass die Anbindung letztendlich wieder manuell erfolgen muss. Deswegen haben wir in unserem Ansatz die bestehende Sprache useML um das so genannte Funktionsmodell erweitert, welches uns ermöglicht, Aufrufe der Applikationslogik automatisch mit in die Software zu generieren. Dies hat den Vorteil, dass der Prozess nun automatisch ablaufen kann, jedoch unter der Bedingung, dass die Kommunikation detailliert in dem Modell spezifiziert wird. Wie oben erwähnt, beinhaltet das Benutzungsmodell Informationen über die komplette Umgebung, d.h. die einzelnen Geräte, zu denen die Kommunikation aufgebaut werden soll, sowie die einzelnen Benutzeraufgaben, zu denen jeweils eine Kommunikation ausgeführt werden kann. Zusätzlich erweitert das Funktionsmodell das Benutzungsmodell auf zwei unterschiedlichen Ebenen. Auf Geräteebene bietet es Informationen, wie eine Kommunikation zu diesem Gerät aufgebaut werden kann, und beschreibt die Struktur der Kommunikation; auf Ebene der elementaren Benutzungsobjekte lassen sich Verknüpfungen zum Funktionsmodell anlegen, die dann letztendlich die Kommunikation auslösen oder beeinflussen können.

Die zugrundeliegende Formalisierung der Kommunikation basiert zum Teil auf dem *Uniform Resource Identifier* (URI) Standard. Dabei wird im Wesentlichen auf Schema, Host und eine Datenreferenz zurückgegriffen. Optional wurden noch Felder für Gerätenummer, Gerätetyp und Priorität vorgesehen. Dies erwies sich als ausreichend, um eine Verbindung herzustellen. Des Weiteren orientiert sich die nachrichtenbasierte Kommunikation in der *SmartFactory*^{KL} auf der Funktionsweise des verbreiteten PROFIBUS-Protokolls. Daher musste ein Modell entwickelt werden, welches es ermöglicht, die zu sendenden und zu empfangenden Datenpakete automatisch zur Laufzeit zu strukturieren und zu verstehen. Hier haben wir ein Datensatz-Objekt definiert, welches die Position, Länge, Bezeichnung, sowie das Datenformat beinhaltet. Zusätzlich besteht die Option, die Information des Paketes durch Angabe von Einheit (z.B. °C, m³), Umrechnungsfaktor, max/min-Wert, signifikanten Dezimalstellen und einer Statusnachricht näher zu spezifizieren.

4 GaBi-Prototyp

Um die Machbarkeit unseres vollautomatischen Generierungsprozesses zu zeigen haben wir exemplarisch einen Prototyp entwickelt. Hierbei werden elementare Benutzungsobjekte des Benutzungsmodells direkt auf eine finale Plattform (Java/Swing) abgebildet. Des Weiteren wurden in den vorgestellten Modellen alle nötigen Informationen formalisiert, so dass der Prozess ohne manuelle Interaktion durchgeführt werden kann. Durch das raumbasierte Benutzungsmodell ist es möglich eine Benutzungsschnittstelle zu erstellen, mittels welcher der Benutzer die Möglichkeit hat, mehrere Geräte in einer einzigen Benutzungsschnittstelle zu steuern. Dadurch konnten wir unsere primäre Anforderung adressieren – die Verbesserung der Benutzbarkeit solcher Umgebungen, in denen die Benutzungsschnittstellen an unterschiedlichen physikalischen Geräten angebracht sind und dadurch auch unterschiedliche Benutzungserfahrungen aufweisen. Mit Hilfe des Funktionsmodells ist es zusätzlich möglich, Aktionen durch den Benutzer konkreten Funktionalitäten der Applikationslogik zuzuordnen und so eine funktionsfähige universelle Benutzungsschnittstelle zu generieren. Geräte sowie Kommunikationsprotokolle

der *SmartFactory*^{KL} wurden in den entsprechenden Modellen formalisiert. Mittels Bluetooth kann mit der Umgebung kommuniziert werden. Nachdem der Generator auf einem Paceblade TabletPC installiert wurde, welcher über direkte Interaktion bedienbar ist, konnte dieser schon anhand der Modelle eine Benutzungsschnittstelle generieren. Diese ermöglicht die Steuerung verschiedener Geräte in der Umgebung. Angezeigt werden nur Elemente von Geräten, welche zum einen in den Modellen spezifiziert wurden und zum anderen, die auch in der Umgebung durch die Software gefunden wurden. Abbildung 1 zeigt ein erstes Ergebnis dieses vereinfachten Generierungsprozesses.

Abbildung 1. Generierte universelle Benutzungsschnittstelle mit Debugging-Informationen (links), installiert auf einem TabletPC wodurch die Produktionsanlage gesteuert werden kann (rechts).

Auf der linken Seite kann man die Navigation zwischen den einzelnen verfügbaren Geräten erkennen, welche durch die Geräteverbünde im Benutzungsmodell spezifiziert wurden und die rechte Seite zeigt die individuelle Benutzungsschnittstelle eines einzelnen Gerätes. Aufgrund des einfachen Abbildungsprozesses wird hierbei genau ein elementares Benutzungsobjekt durch ein widget in Java dargestellt. Ebenso wurde hierbei noch kein Wert auf die optimale Darstellung aus Sicht der Benutzbarkeit gelegt, weil diese Information in Zukunft in die Benutzungsschnittstelle durch Anwendung von sogenannten Usability Patterns generiert werden soll.

Bisher haben wir dadurch gezeigt, dass es prinzipiell möglich ist, unter Verwendung der vorgeschlagenen einfachen Modelle eine voll funktionsfähige Benutzungsschnittelle für intelligente Produktionsumgebungen zu generieren.

5 Zusammenfassung & Ausblick

In unserer Arbeit haben wir zunächst die verschiedenen Modelle beschrieben, welche Grundlagen unseres Generierungsprozesses sind. Insbesondere war es dabei wichtig, die Aufgaben von Benutzern zur formalisieren, damit eine homogenisierte Benutzungsschnittstelle generiert werden konnte. Ein wesentlicher Nachteil heutiger Benutzerinteraktion in intelligenten Produktionsumgebungen ist immer noch, dass die Interaktion direkt an physikalischen Geräten ausgeführt werden muss. Zudem erfolgt dies in unterschiedlichen Modalitäten und führt je nach Hersteller zu unterschiedlichen Benutzungserfahrungen. Dies kann zu einem unnötig komplexen Arbeitsablauf des Benutzers und

dadurch zu vermeidbaren menschlichen Fehler führen. Die durch unseren Ansatz homogenisierte Benutzungsschnittstelle bietet Zugriff auf die unterschiedlichsten Geräte, durch eine einzige Java-basierte Schnittstelle. Ferner konzentrieren sich viele ähnliche Ansätze lediglich auf die Formalisierung der Darstellung (Präsentationsmodelle), weswegen es ein wichtiger Beitrag unseres Ansatzes die Entwicklung eines Metamodells für industrielle mobile Kommunikation über standardisierte Protokolle war, welches in das raumbasierte Benutzungsmodell integriert werden konnte. Die Machbarkeit unseres Ansatzes durch die Umsetzung eines Prototypen gezeigt.

Dies war nur ein erster wichtiger Schritt, aber es bleiben noch wichtige Fragen offen, die in zukünftigen Arbeiten adressiert werden. Ein wesentlicher Punkt ist die Integration von sogenannten Usability Patterns. Diese stellen, ähnlich wie Software Design Patterns, eine etablierte Lösung von häufigen Problemen bei der Entwicklung und der Gestaltung von Benutzungsschnittstellen dar. Momentan existieren eine Reihe umfassender Bibliotheken solcher Patterns, welche aber nur in sehr informeller Notation vorliegen. Gegenwärtig werden unterschiedliche Sprachen zur Formalisierung dieser Patterns entwickelt. Für die Verwendung innerhalb von automatischen Generatoren muss diese Sammlung zusätzlich noch durch Algorithmen interpretiert werden können, was aktuell noch ein Problem darstellt.

Im Allgemeinen wird die Darstellung der Benutzungsschnittstelle durch ein Präsentationsmodell definiert. Es spezifiziert, wie bspw. visuelle, haptische oder auditive Interaktionsobjekte (widgets) an der Benutzungsschnittstelle dargestellt werden [PE99]. Die Art und Weise, wie z.B. eine Benutzungsschnittstelle repräsentiert wird, hängt von den Bedingungen des Ausgabemediums ab (u. a. Displaygröße, Farbdarstellung, etc.). Das Präsentationsmodell wird typischerweise in zwei Modelle, das abstrakte und konkrete Präsentationsmodell aufgeteilt. Die Einführung von Ebenen der abstrakten und konkreten Benutzungsschnittstelle hat weitere Vorteile. Eine abstrakte Benutzungsschnittstelle kann dazu verwendet werden, viele unterschiedliche konkrete Benutzungsschnittstellen (Multimodalität) zu erzeugen. Dabei wird der Wiederverwendungsgrad von Modellen gesteigert und die Komplexität sowie die Kosten der Benutzungsschnittstellenentwicklung gesenkt. Alle an der Entwicklung der Benutzungsschnittstelle beteiligten Personen können Ihre gewohnten Sprachen und Konzepte weiterverwenden. Die modellbasierte Entwicklung von Benutzungsschnittstellen strebt einen möglichst hohen Informationsgehalt der abstrakten Benutzungsschnittstelle an, da die eindeutige Spezifikation der Details auf Modellebene einfacher durchzuführen ist, als auf Quellcodeebene.

Ein ausführlicher Vergleich aktueller Beschreibungssprachen findet sich in [SV03] und [Lu04]. Diese Sprachen bilden zur Beschreibung der Benutzungsschnittstelle eine gute Grundlage, da mit Hilfe dieser Sprachen z.B. eine Trennung von Navigationsstruktur, Design, Dialogelementen, dynamischem Verhalten und Lokalisierung erreicht wird.

Intelligente Produktionsumgebungen sind heute keine statischen Konstrukte mehr. Es wird der Fall sein, dass diese sich automatisch während der Laufzeit rekonfigurieren lassen um z.B. mögliche aufgetretende Fehler zu kompensieren. Dadurch ändert sich auch die Struktur der präsenten Geräte. Zusätzlich wird es auch möglich sein, dass einfach zur Laufzeit vorhandene Geräte durch neue ausgetauscht, oder neue Geräte zusätz-

lich eingeführt werden. Dies muss sich dann entsprechend auch im zugehörigen Modell widerspiegeln, was die Grundlage der Generierung ist. Folglich muss sich auch die generierte Benutzungsschnittstelle analog zu dem Modell durch den Generator anpassen lassen. Dieser Aspekt der Adaption wurde in unserem aktuellen Generator nur sehr rudimentär behandelt, da es prinzipiell möglich ist, aufgrund eines veränderten Modells die Benutzungsschnittstelle anzupassen. Da Veränderungen der Benutzungsschnittstelle (die in unserem Kontext unvermeidbar sind) die Benutzungserfahrung negativ beeinflussen, wird ein wesentlicher Fokus unserer Entwicklungen auch darauf liegen, die Adaption benutzerfreundlich zu gestalten. Dabei sollen in Experimenten unterschiedliche Adaptionsmechanismen auf Basis der Integrationsfähigkeit in die Domäne der intelligenten Produktionsumgebungen bzgl. ihrer Effizienz verglichen werden.

Literaturverzeichnis

[AM03] Aarts, E.; Marzano, S.: The New Everyday – Views on Ambient Intelligence. Rotterdam: 010 Publishers, 2003.

[Br09a] Breiner, K.; Maschino, O.; Görlich, D.; Meixner, G.: Towards automatically interfacing application services integrated in a automated model based user interface generation process, Proc. of the 4th International Workshop on Model-Driven Development of Advanced User Interfaces (MDDAUI), Sanibel Island, USA, CEUR Workshop Proceedings Vol. 439, 2009.

[Br09b] Breiner, K.; Maschino, O.; Görlich, D.; Meixner, G., Zühlke, D.: Run-Time Adaptation of a Universal User Interface for Ambient Intelligent Production Environments, Proc. 13th International Conference on Human-Computer Interaction (HCII) 2009, San Diego, USA, 2009.

[IEC06] IEC 61804-3: Function blocks (FB) for process control – Part 3: Electronic Device Description Language (EDDL), 2006.

[Lu04] Luyten, K.: Dynamic User Interface Generation for Mobile and Embedded Systems with Model-Based User Interface Development. PhD thesis, Transnationale Universiteit Limburg, 2004.

[MGS08] Meixner, G.; Görlich, D.; Schäfer, R.: Unterstützung des Useware-Engineering Prozesses durch den Einsatz einer modellbasierten Werkzeugkette, USEWARE-Tagung 2008, Baden-Baden, VDI-Bericht 2041, VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik, Düsseldorf, Germany, VDI-Verlag, S. 219-232, 2008.

[PE99] Puerta, A.; Eisenstein, J.: Towards a General Computational Framework for Model-Based Interface Development Systems, Proc. of the 4th International Conference on Intelligent User Interfaces, S. 171-178, 1999.

[SV03] Souchon, N.; Vanderdockt, J.: A Review of XML-Compliant User Interface Description Languages. Proc. of the 10th International Workshop on Interactive Systems: Design, Specification and Verification, S. 377-391, 2003.

[VDI00] VDI-Richtlinie 3850 Blatt 1: Nutzergerechte Gestaltung von Bediensystemen für Maschinen, 2000.