TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Informatik
Lehrstuhl für Wirtschaftsinformatik (I 17)
Univ.-Prof. Dr. Helmut Krcmar

Endbenutzer-Entwicklung mobiler ERP-Applikationen durch den Einsatz eines domänenspezifischen Entwicklungswerkzeuges

Marcus Homann

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Uwe Baumgarten

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Helmut Krcmar
2. Univ.-Prof. Dr. Johann Schlichter

Die Dissertation wurde am 16.10.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26.11.2014 angenommen.

Zusammenfassung

Viele Unternehmen setzen ERP-Systeme zur Unterstützung ihrer betrieblichen Prozesse ein. Durch die weite Verbreitung mobiler Endgeräte wie Smartphones, wächst der Bedarf über diese Geräte auf Daten und Funktionalitäten der ERP-Systeme zuzugreifen. Die Entwicklung zugehöriger Applikation ist jedoch aufgrund der erforderlichen Programmierfähigkeiten und Kenntnisse der entsprechenden Entwicklungswerkzeuge aktuell einer speziellen Gruppe von professionellen Softwareentwicklern vorbehalten.

Forschungsarbeiten im Bereich *Endbenutzer-Entwicklung* beschäftigten sich mit der Befähigung von Personen ohne professionelle Softwareentwicklungsexpertise, selbst Softwareapplikationen zu entwickeln. Hierdurch können Anforderungen besser umgesetzt werden und Kosten eingespart werden. Dies geschieht durch die Bereitstellung geeigneter Entwicklungswerkzeuge, welche die Entwicklung auf einer abstrakteren, domänenspezifischen Ebene ermöglichen. Die vorliegende Arbeit greift Konzepte der Endbenutzer-Entwicklung auf und wendet diese im Bereich mobiler ERP-Applikationen an. In diesem Zusammenhang wird ein domänenspezifisches Endbenutzer-Entwicklungswerkzeug konzipiert, welches es ERP-Anwendern ermöglicht, selbst mobile ERP-Applikationen zu erstellen.

Das entworfene Entwicklungswerkzeug besteht aus einem konzeptuellen Design und einer prototypischen Implementierung. Das Werkzeug-Design setzt sich zusammen aus einer *domänenspezifischen Sprache*, welche die Spezifikation mobiler ERP-Applikationen erlaubt, einem *Codegenerator*, welcher die spezifizierten Applikationen in lauffähige Applikationen überführt, sowie einer *Benutzungsschnittstelle*, welche die Interaktion mit dem Endbenutzer ermöglicht. Die Anforderungen an das Entwicklungswerkzeug wurden durch eine Literaturanalyse, eine Untersuchung existierender mobiler ERP-Applikationen sowie durch semi-strukturierte Expertenbefragungen ermittelt. Für die Konzeption des Werkzeuges wurde ein modellbasierter Entwicklungsansatz gewählt, welcher die Spezifikation der Applikationen auf einer abstrakteren, domänenspezifischen Ebene erlaubt. Die Überführung in lauffähige Applikationen erfolgt vollautomatisiert durch entsprechende Transformationsvorschriften des Codegenerators. Die prototypische Implementierung des Werkzeuges basiert auf dem HTML5-Framework jQuery Mobile unter Nutzung des Apache Tomcat Webservers als technische Basis.

Das entworfene Entwicklungswerkzeug erlaubt die einfache und schnelle Entwicklung von mobilen ERP-Applikationen ohne Einarbeitungsaufwand. Die Tauglichkeit des Entwicklungswerkzeuges wurde im Rahmen der Evaluation durch einen Nutzertest bestätigt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABAP	Advanced Business Application Programming
ADT	Android Development Kit
AG	Aktiengesellschaft
AI	Artificial Intelligence
ALE	Application Link Enabling
API	Application Programming Interface
AS	Application Server
ASF	Atom Syndication Format
ASML	Automotive Service Modelling Language
AtomPub	Atom Publishing Format
AVD	Android Virtual Device
AVI	Audio Video Interleave
BAPI	Business Application Programming Interface
BFA	Business Framework Architecture
BI	Business Intelligence
BM	Bildschirmmaske
BO	Business Object / Business Objekt
BOT	Business Object Type / Business Objekt Typ
BYOD	Bring Your Own Device
B2B	Business-to-Business
B2C	Business-to-Consumer
CAD	Computer-Aided Design
CBD	Component-based Development
CFO	Chief Financial Officer
CI	Corporate Identity
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
CRUD	Create, Read, Update, Delete
CSCW	Computer Supported Cooperative Work
CSS	Cascading Style Sheets
CTT	ConcurTaskTrees
CTTE	ConcurTaskTrees Environment
DCOM	Distributed Component Object Model
DE	Domain Engineering
DIAG	Dynamic Information and Action Gateway
DODE	Domain-oriented Design Environments
DOM	Document Object Model

DSAG	Deutschsprachige SAP Anwendergruppe
DSL	Domain-Specific Language
DSML	Domain-Specific Modelling Language
DTD	Document Type Definition
EA	Evaluationsaspekt
eCATT	extended Computer Aided Test Tool
ECC	Enterprise Core Component
EE	Enterprise Edition
ERP	Enterprise Resource Planning
ESE	Enterprise Server Edition
EUD	End-User Development
FI	Financials
FuBa	Funktionsbaustein
FTP	File Transfer Protocol
GOMS	Goals, Methods, Methods and Selection Rules
GPL	General Purpose Language
GPS	Global Positioning System
GTA	Groupware Task Analysis
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protokoll
IDoc	Intermediate Document
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Services
IS	Information Systems
IT	Informationstechnologie
ITS	Internet Transaction Server
Java EE	Java Enterprise Edition
JCo	Java Connector
JNI	Java Native Interface
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KMU	Kleine und mittlere Unternehmen
LabView	Laboratory Virtual Instrument Engineering Workbench
LBS	Location Based Services
LO	Logistics
LTE	Long Term Evolution
LUW	Logical Unit of Work
MAC	Media Access Control

MADP	Mobile Application Development Platform
MathML	Mathematical Markup Language
MBO	Mobile Business Object
MBOT	Mobile Business Object Type
MBUID	Model-based User Interface Development
MDA	Model-Driven Architecture
MDSD	Model-Driven Software Development
MEAP	Mobile Enterprise Application Platform
MRP	Material Requirements Planning
MRP-II	Manufacturing Resource Planning
MVC	Model-View-Controller
NB	Navigationsblock
OData	Open Data
OMG	Object Management Group
PbE	Programming by Example
PC	Personal Computer
RCP	Rich Client Platform
RDBMS	Relational Database Management System
REST	Representational State Transfer
RFC	Remote Function Call
RPC	Remote Procedure Call
RSS	Really Simple Syndication
RUP	Rational Unified Process
SaaS	Software as a Services
SAX	Simple API for XML Parsing
SCM	Supply Chain Management
SE	Software Engineering
SID	SAP Identifier
SIM	Subscriber Identity Module
SMP	SAP Mobile Platform
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQD	Semantic Query Design
SQL	Structured Query Language
SUP	Sybase Unwired Platform
SVG	Scalable Vector Graphics
TL	Technology Level
UAN	User Action Notation
UCC	University Competence Center
UCD	User-Centered Design

UI	User Interface
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
VBA	Visual Basic for Applications
VEE	Visual Engineering Environment
VIA	View-Inspect-Act
WCP	Widget Composition Platform
WFMS	Workflow Management System
W-LAN	Wireless Local Area Network
WMF	Windows Media Video
WSDL	Web Service Description Language
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language
XSL-FO	XSL Formatting Objects
XSLT	Extensible Stylesheet Language Transformation

1 Einführung

Enterprise Resource Planning (ERP)-Systemen haben sich in den meisten Unternehmen als zentrale IT-Systeme zur Unterstützung betriebswirtschaftlicher Prozesse etabliert (Gronau 2010, 3 ff.; Kurbel 2013, 97). Sie sind durch ihre Integration über mehrere betriebswirtschaftliche Funktionsbereiche, wie beispielsweise dem Finanzwesen, der Personalwirtschaft und der Logistik, charakterisiert (Gronau 2010, 3 f.; Kurbel 2013, v). Für Anwender von ERP-Systemen existieren ERP-Applikationen[1], welche auf die Unterstützung ausgewählter Aktivitäten innerhalb eines Geschäftsprozesses ausgerichtet sind. Zudem werden Programmierschnittstellen bereitgestellt, um auf die Daten und Funktionalitäten von ERP-Systemen über externe Softwareapplikationen zugreifen zu können. Die Bedienung der ERP-Applikationen erfordert jedoch das Verständnis des betriebswirtschaftlichen Sachverhaltes und der jeweiligen Fachbegriffe. Dieses Verständnis wird in der vorliegenden Arbeit als *domänenspezifisches Wissen* in der Domäne „ERP“ bezeichnet. Um dieses Wissen zu erlangen sind häufig spezielle Schulungen notwendig, um den Umgang mit ERP-Applikationen zu erlernen (Oja/Lucas 2010, 2). Zur Nutzung der Programmierschnittstellen ist zusätzlich oftmals die Kenntnis der technischen Spezifika des verwendenden ERP-Systems notwendig.

Seit dem Durchbruch mobiler Geräte im Konsumentenmarkt, insbesondere von *Smartphones* und *Tablet-PCs*, ist der Bedarf über diese auch auf die ERP-Systeme von Unternehmen zugreifen zu können, gestiegen. Der Smartphone- und Tablet-PC-Markt ist jedoch derzeit stark fragmentiert und dynamisch (Gartner 2014). Aktuell sind mehrere Gerätehersteller mit eigenen Modellvarianten am Markt präsent. Zudem nutzen die existierenden Geräte teilweise unterschiedliche Betriebssysteme und die zugehörigen Applikationen werden mit unterschiedlichen Programmiersprachen, Entwicklungsumgebungen und Entwicklungsbibliotheken erstellt (Homann et al. 2013b, 31 ff.).

Unter *mobilen ERP-Applikationen* werden in der vorliegenden Arbeit mobile Applikationen verstanden, die auf Daten und Funktionalitäten eines ERP-Systems zugreifen und die Interaktion mit dem Anwender steuern (Beckert et al. 2012, 138; Homann et al. 2013b, 50). Aufgrund der notwendigen Fähigkeiten zur Entwicklung mobiler Applikationen und dem zusätzlich erforderlichen ERP-Domänenwissen, stellt die Entwicklung mobiler ERP-Applikationen derzeit eine Herausforderung dar. Diese wird aktuell von professionellen Softwareentwicklern in Zusammenarbeit mit ERP-Domänenexperten bewältigt.

Im Folgenden wird die Forschungsarbeit motiviert. Zudem wird das verfolge Forschungsziel und die zugehörigen forschungsleitenden Fragestellungen vorgestellt. Anschließend werden das forschungsmethodische Design, die Limitationen und der Aufbau der Arbeit beschrieben.

[1] In manchen ERP-Systemen, wie beispielsweise SAP-ERP-Systemen werden ERP-Applikationen auch als Transkationen bezeichnet (vgl. z.B.. (Forsthuber 2005, 375))

1.1 Motivation und Relevanz

Die Anforderungen an mobile ERP-Applikationen werden von ERP-Domänen-Experten spezifiziert, welche i.d.R. auch die späteren Nutzer der entwickelten Applikationen darstellen. Die Entwicklung der Applikationen erfolgt hingegen durch professionelle Softwareentwickler. Aufgrund der unterschiedlichen Blickwinkel beider Personengruppen besteht hierbei das Risiko, dass Anforderungen falsch verstanden und umgesetzt werden (Reppening/Ioannidou 2006, 51 f.). Zudem ist der erforderliche Abstimmungsbedarf bei diesem Vorgehen mit hohen Zeit- und Kostenaufwänden verbunden (Lieberman et al. 2006a, 2).

Eine Möglichkeit den beschriebenen Abstimmungsaufwand zu reduzieren besteht darin, die Domänen-Experten zu befähigen, ihre benötigten mobilen Applikationen selbst zu entwickeln. In diesem Fall werden die Anforderungen direkt von den späteren Nutzern umgesetzt. Dadurch können Abstimmungsaufwände mit Dritten vermieden werden. Zudem wird hierdurch der Engpass an verfügbaren, professionellen Softwareentwicklern reduziert. Als Folge wäre es ferner möglich, eine größere Anzahl an benötigten mobilen ERP-Applikationen zu entwickeln.

Domänen-Experten im ERP-Bereich besitzen jedoch entweder keine Programmierkenntnisse oder zumindest nicht die Softwareentwicklungskenntnisse, welche für die Umsetzung mobiler Applikationen notwendig wären. Zudem fehlt ihnen oftmals die Zeit und Motivation sich diese Fähigkeiten anzueignen (Lieberman 2001, ix). Daher besteht die Herausforderung darin, den Domänen-Experten ein geeignetes Werkzeug zur Entwicklung von mobilen ERP-Applikationen bereitzustellen, welches ihre Anforderungen umsetzen kann und dessen Bedienung ihren Fähigkeiten entspricht (Beringer 2004, 40).

Für die Befähigung von Domänen-Experten, ihre gewünschten Applikationen selbst zu entwickeln, hat sich in der Wissenschaft der Begriff *Endbenutzer-Entwicklung* (engl. End-User Development (EUD)) etabliert (Lieberman et al. 2006b, 2 ff.). Konsequenterweise werden Domänen-Experten bzw. spätere Nutzer als *Endbenutzer* bezeichnet (Nardi 1993, 5; Poswig 1996, 2; Spahn 2010, 24). Erfolgreiche Beispiele für die Endbenutzer-Entwicklung sind unter anderem der Entwurf von Tabellenkalkulationen, Statistik- und CAD (Computer Aided Design)-Applikationen (Nardi 1993, xi f.; Lieberman et al. 2006a, 2). Zudem gibt es bereits vielversprechende Endbenutzer-Entwicklungs-Ansätze für die Gestaltung von Webseiten (siehe z.B. (Rode et al. 2006, 168 ff.)). Generell besteht Konsens darüber, dass ein Endbenutzer-Entwicklungs-Ansatz und ein zugehöriges Entwicklungswerkzeug auf eine bestimmte Domäne und damit auf eine Endbenutzergruppe mit einem bestimmten Hintergrundwissen fokussiert sein muss, um erfolgreich zu sein (Reppening/Ioannidou 2006, 82).

Die vorliegende Forschungsarbeit geht der Fragestellung nach, wie Endbenutzer in die Lage versetzt werden können, selbst mobile ERP-Applikationen zu entwickeln. Hierzu soll eine geeignete *domänenspezifische Sprache* konzipiert sowie ein zugehöriges *domänenspezifisches Entwicklungswerkzeug* konzipiert und prototypisch implementiert werden. Dabei liegt die Annahme zugrunde, dass mobile ERP-Applikationen gewisse Gemeinsamkeiten besitzen, welche als Grundlage für zugehörige Sprachelemente der domänenspezifischen Sprache

verwendet werden können. Die Herausforderung dieser Arbeit besteht darin, diese Gemeinsamkeiten zu identifizieren und in einer domänenspezifischen Sprache abzubilden. Zudem sollen die Anforderungen an ein Entwicklungswerkzeug für mobilen ERP-Applikationen identifiziert werden. Dieses Entwicklungswerkzeug soll es Endbenutzern auf Basis der domänenspezifischen Sprache ermöglichen, eigenständig mobile ERP-Applikationen zu entwickeln.

Um ein erfolgreiches Endbenutzer-Entwicklungswerkzeug zu gestalten, müssen Endbenutzern Bausteine zur Verfügung gestellt werden, welche sie aufgrund ihrer Kenntnisse in ihrer täglichen Arbeitsumgebung oder ihrem vorhandenen Allgemeinwissen umgehend verstehen können (Beringer 2004, 40). Hierdurch werden sie in die Lage versetzt, neue Softwareapplikationen durch die Kombination und Modifikation bestehender Bausteine zu erzeugen. Jedoch benötigen Computersysteme Bausteine, welche präzise definiert sind. Daher ist ein Transformationsvorgang von den intuitiv verständlichen Bausteinen zur Unterstützung der Endbenutzer hin zu präzise definierten Bausteinen für Computersysteme notwendig (Berti et al. 2006, 144). Die zu entwickelnde domänenspezifische Sprache für mobile ERP-Applikationen besteht daher aus einer Menge domänenspezifischer Bausteine inkl. ihrer Verknüpfungsregeln. Das zugehörige Endbenutzer-Entwicklungswerkzeug stellt eine Benutzungsschnittstelle für die Nutzung der domänenspezifischen Sprache bereit sowie einen Code Generator zur Generierung des eigentlichen Programmcodes für die mobilen ERP-Applikationen auf Basis der spezifizierten Bausteine.

1.2 Forschungsziel der Arbeit

Das Ziel der vorliegenden Dissertation ist es, die Entwicklung mobiler ERP-Applikationen für Endbenutzer zu ermöglichen. Hierzu soll ein *domänenspezifisches Entwicklungswerkzeug* für mobile ERP-Applikationen konzipiert werden. Zielgruppe des Entwicklungswerkzeuges sind ERP-Domänenexperten, welche keine professionelle Ausbildung in der Softwareentwicklung besitzen. Um die Umsetzbarkeit des entwickelten Konzeptes zu demonstrieren sowie eine Evaluation mit potentiellen Nutzern zu ermöglichen wird eine prototypische Implementierung des Entwicklungswerkzeuges durchgeführt.

Um Anforderungen und Gestaltungsideen für das gewünschte Werkzeug zu sammeln, sollen zunächst gemeinsame Charakteristiken mobiler ERP Anwendungen identifiziert werden. Die dabei gewonnen Erkenntnisse sollen anschließend genutzt werden, um daraus eine domänenspezifische Sprache (engl. Domain-Specific Language (DSL)) für mobile ERP Anwendungen abzuleiten. Die domänenspezifische Sprache soll als Beschreibungssprache die Konstruktion mobiler ERP Anwendungen ermöglichen. Eine über diese Sprache definierte Applikation soll schließlich über einen *Code Generator* in eine lauffähige Applikation transformiert werden können. Hierzu soll ein geeigneter Transformator die DSL-Beschreibung in eine lauffähige mobile ERP-Applikation überführen. Durch die Trennung zwischen dem eigentlichen Applikationscode und einem abstrakteren Beschreibungsformat durch die DSL entsteht die Möglichkeit, die über die DSL spezifizierte Applikation in unterschiedliche Ausgabeformate zu transformieren. Um die Spezifikation einer mobilen ERP-Applikation durch Endbenutzer zu

erleichtern soll zusätzlich ein geeignetes Entwicklungswerkzeug konzipiert und prototypisch implementiert werden. Für die prototypische Implementierung des Entwicklungswerkzeuges soll ein SAP-ERP-System als beispielhaftes ERP-System verwendet werden. Die Kommunikation der mobilen ERP-Applikationen mit dem SAP-ERP-System soll über dessen Standard-Programmierschnittstellen, den Business Application Programming Interfaces (BAPIs) erfolgen. Abbildung 1-1 illustriert die beschriebene Vision der Arbeit.

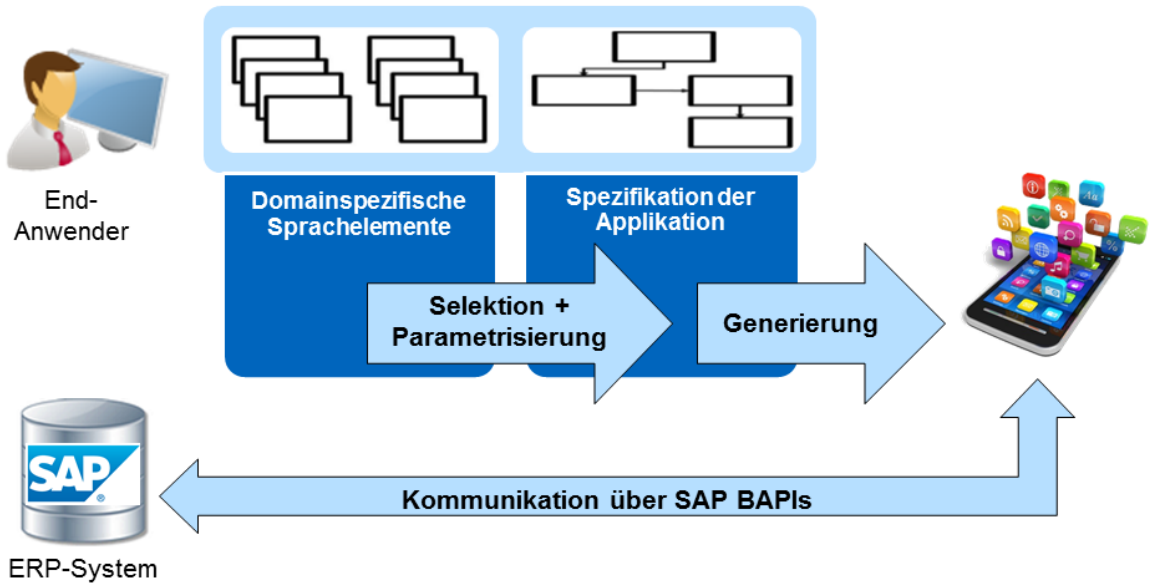

Abbildung 1-1: Vision der Arbeit
Quelle: Eigene Darstellung

Mit dem angestrebten Entwicklungswerkzeug werden insbesondere zwei Anwendungsfälle fokussiert. Im ersten Anwendungsfall soll ein *ERP-Administrator* die Möglichkeit bekommen mobile ERP Anwendungen zu entwickeln und für eine Mitarbeitergruppe bereitzustellen. Bei einem ERP-Administrator können gewöhnlich fundierte IT-Kenntnisse vorausgesetzt werden. Jedoch umfasst der Aufgabenbereich eines ERP-Administrators gewöhnlich nicht die Implementierung neuer Applikationen, sondern vielmehr die Sicherstellung des Betriebs sowie die Wartung der ERP-Systeme. Ein SAP-Administrator könnte beispielsweise eine mobile ERP-Applikation für alle Berater des Unternehmens entwickeln, mit welcher diese ihre Projektzeiten auf ihrem mobilen Endgerät erfassen und an das ERP-System schicken können.

Der zweite Anwendungsfall bezieht sich auf den *Nutzer* der mobilen ERP Anwendung. Dieser soll durch ein geeignetes Entwicklungswerkzeug in die Lage versetzt werden selbst individuelle mobile ERP-Applikationen für seinen Eigenbedarf zu entwickeln. Beispielsweise könnte sich ein Vertriebsmitarbeiter mit einem solchen Werkzeug eine mobile ERP Anwendung für seine Kundenbesuche entwickeln. Er hätte die Möglichkeit sich Funktionalitäten wie die Liste seiner Kundenkontakte, deren Präferenzen und Aufträge sowie einen Produktkatalog des

Unternehmens zu einer individuellen Applikation zusammenzubauen und notwendige Eingaben durch häufig genutzte Standardwerte zu belegen.

Sowohl bei SAP-Administratoren als auch bei den Nutzern der mobilen Applikation wird in dieser Arbeit angenommen, dass diese keine professionellen Kenntnisse in der Programmierung mobiler Applikationen besitzen und folglich als Endbenutzer betrachtet werden können. Die fehlenden professionellen Softwareentwicklungskenntnisse müssen bei der Gestaltung des fokussierten Entwicklungswerkzeugs berücksichtigt werden.

1.3 Forschungsleitende Fragestellungen

Um die skizzierte domänenspezifische Sprache sowie das zugehörige Endbenutzer-Entwicklungswerkzeug zu konzipieren, wird die Arbeit anhand von drei aufeinander aufbauenden, forschungsleitenden Fragestellungen betrachtet. Zunächst soll die Domäne der mobilen ERP-Applikationen systematisch untersucht werden. Ziel ist es die speziellen Charakteristiken von mobilen ERP-Applikationen zu identifizieren. Zudem sollen die aktuellen Herausforderungen bei der Entwicklung mobiler ERP-Applikationen untersucht werden. Daraus leitet sich die erste Forschungsfrage dieser Arbeit ab, welche wie folgt lautet:

1. Welche gemeinsamen Charakteristiken besitzen mobile ERP-Applikationen und welche Herausforderungen existieren aktuell bei ihrer Entwicklung?

Bei der Beantwortung der ersten Forschungsfrage wird ein besonderes Augenmerk auf die technischen und funktionalen Gemeinsamkeiten existierender mobiler ERP-Applikationen gelegt. Diese sollen im weiteren Verlauf der Forschungsarbeit als Grundlage für wiederverwendbare Entwicklungsbausteine dienen.

Die Untersuchung der aktuellen Herausforderungen bei der Entwicklung mobiler ERP-Applikationen, dient zum einen zur Verdeutlichung der praktischen Relevanz des Forschungszieles. Zum anderen sollen hierdurch die aktuellen Problemstellungen bei der Entwicklung mobiler ERP-Applikationen aufgezeigt werden, um diese bei der Konzeption des Entwicklungswerkzeuges berücksichtigen zu können.

Auf Basis der identifizierten Charakteristiken und Herausforderungen, sollen in einem nächsten Schritt Anforderungen an das gewünschte Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen abgeleitet werden. Diese sollen in einem darauffolgenden Schritt als Grundlage zur Konzeption einer geeigneten Softwarearchitektur für das Entwicklungswerkzeug sowie zur Auswahl einer zugehörigen technischen Basis verwendet werden. Daraus leitet sich die zweite Forschungsfrage der Arbeit ab, welche wie folgt lautet:

2. Welche Anforderungen hinsichtlich der Gestaltung eines Endbenutzer-Entwicklungswerkzeuges für mobile ERP-Applikationen ergeben sich aus den identifizierten Charakteristiken und Herausforderungen und welche konzeptuelle Softwarearchitektur und technische Basis ist geeignet um diese Anforderungen umzusetzen?

Aufbauend auf dem in der ersten Forschungsfrage entwickeltem Problemverständnis beschäftigt sich die zweite Forschungsfrage mit der Anforderungsermittlung und Konzeption des gewünschten Entwicklungswerkzeuges. Hierbei sollen auftretende Designalternativen diskutiert und getroffene Designentscheidungen entsprechend begründet werden. Neben dem eigentlichen Entwicklungswerkzeug sollen die mobilen ERP-Applikationen in Form einer domänenspezifischen Sprache spezifiziert werden. Hierfür werden zunächst wissenschaftliche Erkenntnisse zur Gestaltung domänenspezifischer Sprachen untersucht sowie ausgewählte domänenspezifische Sprachen und zugehörige Entwicklungswerkzeug analysiert. Auf Basis der gewonnenen Erkenntnisse werden die zentralen Gestaltungselemente einer domänenspezifischen Sprache für die Domäne „mobile ERP-Applikationen“ konzipiert.

Anschließend wird die konzeptuelle Architektur eines zugehörigen Entwicklungswerkzeuges für Endbenutzer entworfen. Hierfür wird sowohl der Codegenerator als auch die zugehörige Benutzungsschnittstelle des Werkzeuges konzipiert. Zudem werden mögliche technische Umsetzungsvarianten vorgestellt und diskutiert.

Im Rahmen der dritten Forschungsfrage werden die in der zweiten Forschungsfrage vorgestellten Konzepte prototypisch implementiert und anschließend evaluiert. Die zugehörige Fragestellung lautet:

3. Welche Implikationen hinsichtlich der Weiterentwicklung und Nutzung des vorgestellten Entwicklungswerkzeuges ergeben sich aus dessen praktischer Nutzung durch Endbenutzer?

Ziel der dritten Forschungsfrage ist es, die Umsetzbarkeit der vorgestellten Konzepte zu demonstrieren. Zudem soll mit Hilfe der prototypischen Implementierung des Werkzeuges ein kontrolliertes Experiment mit potentiellen Endbenutzern durchgeführt werden, um die Umsetzung der Anforderungen und die Benutzungsfreundlichkeit direkt durch die Zielgruppe zu bewerten.

1.4 Forschungsmethodisches Design

Im Bereich der *Wirtschaftsinformatik* bzw. dem angelsächsischen *Information Systems (IS) Research* lassen sich generell zwei Forschungsrichtungen unterscheiden. Im angelsächsischen Raum ist der empirisch orientierte *Behaviorismus* verbreitet (Bichler 2006, 133). Ziel zugehöriger Forschungsarbeiten ist die „Beobachtung von Eigenschaften von Informationssystemen und des Verhaltens von Benutzern“ (Österle et al. 2010, 1). Im Mittelpunkt der Forschung steht die Messung der „Verwendung und Auswirkung von existierenden Informationssyste-

men auf Individuen, Gruppen und Organisationen“ (Bichler 2006, 133). Behavioristisch orientierten Forschungsansätzen werden jedoch teilweise aufgrund ihrer mangelnden Praxisrelevanz kritisiert (Khazanchi 2000, 27 f.; Kock et al. 2002, 332 f.).

In der zweiten Forschungsrichtung, der *gestaltungsorientierte Forschung* bzw. dem angelsächsischen *Design Science*, liegt der Fokus hingegen auf der Entwicklung innovativer *Artefakte* zur Lösung aktueller und zukünftiger *praxisrelevanter Problemstellungen* (Bichler 2006, 135; Österle et al. 2010, 4). In der Wirtschaftsinformatik werden folgende Artefakttypen unterschieden: *Konstrukte* (Vokabulare und Symbole), *Modelle* (Abstraktionen und Repräsentationen), *Methoden* (Algorithmen und Praktiken) sowie *Instanziierungen* (implementierte und prototypische Systeme) (March/Smith 1995, 253; Hevner et al. 2004, 77). Der eigentliche Wertbeitrag der gestaltungsorientierten Forschung entsteht jedoch nicht durch die Entwicklung eines Artefakts, sondern durch die verallgemeinerbaren Erkenntnisse, welche durch das Artefakt oder dessen Entwicklungsprozess gewonnen werden können (Gregor 2006, 629).

Gemäß dieser Definitionen gliedert sich die vorliegende Arbeit in den Bereich der gestaltungsorientierten Wirtschafsinformatik ein: Ausgehend von einer konkreten Problemstellung (Endbenutzer-Entwicklung mobiler ERP-Applikationen) werden innovative Artefakte (domänenspezifische Sprache, Softwarearchitektur und prototypische Implementierung) geschaffen, aus denen verallgemeinerbare Erkenntnisse gewonnen werden sollen.

Neben der Adressierung einer praxisrelevanten Problemstellung ist in der gestaltungsorientierten Wirtschaftsinformatik ein *stringentes Forschungsvorgehen* wichtig. Hierbei sollten existierende Grundlagen und Methoden der *Wissensbasis* bei der Entwicklung des Artefakts berücksichtigt werden. Zudem sollte die Wissensbasis um die aus der Forschung gewonnen Erkenntnisse erweitert werden. Letzteres ist beispielsweise durch Vorträge auf wissenschaftlichen Konferenzen oder Publikationen in wissenschaftlichen Magazinen möglich. Neben der Entwicklung der Artefakte ist deren *Evaluation* ein zentraler Bestandteil des gestaltungsorientierten Forschungsprozesses (Hevner 2007, 90 f.; Österle et al. 2010, 4 f.). Durch die Evaluation lassen sich Erkenntnisse für weitere Optimierungen sowohl für das entwickelte Artefakt als auch dessen Erstellungsprozess ableiten (Hevner et al. 2004, 78). Um eine kontinuierliche Optimierung zu ermöglichen, sollten idealerweise mehrere Evaluationsphasen durchlaufen werden (Hevner 2007, 90 f.). Aufgrund der sich abwechselnden Entwicklungs- und darauffolgenden Evaluationsphasen, spricht Hevner (2007, 90 f.) in diesem Zusammenhang von sogenannten *Design-Zyklen*. Abbildung 1-2 illustriert die beschriebenen Zusammenhänge in Form eines Rahmenkonzepts für Design-Science-Forschung nach Hevner et al. (2004, 80).

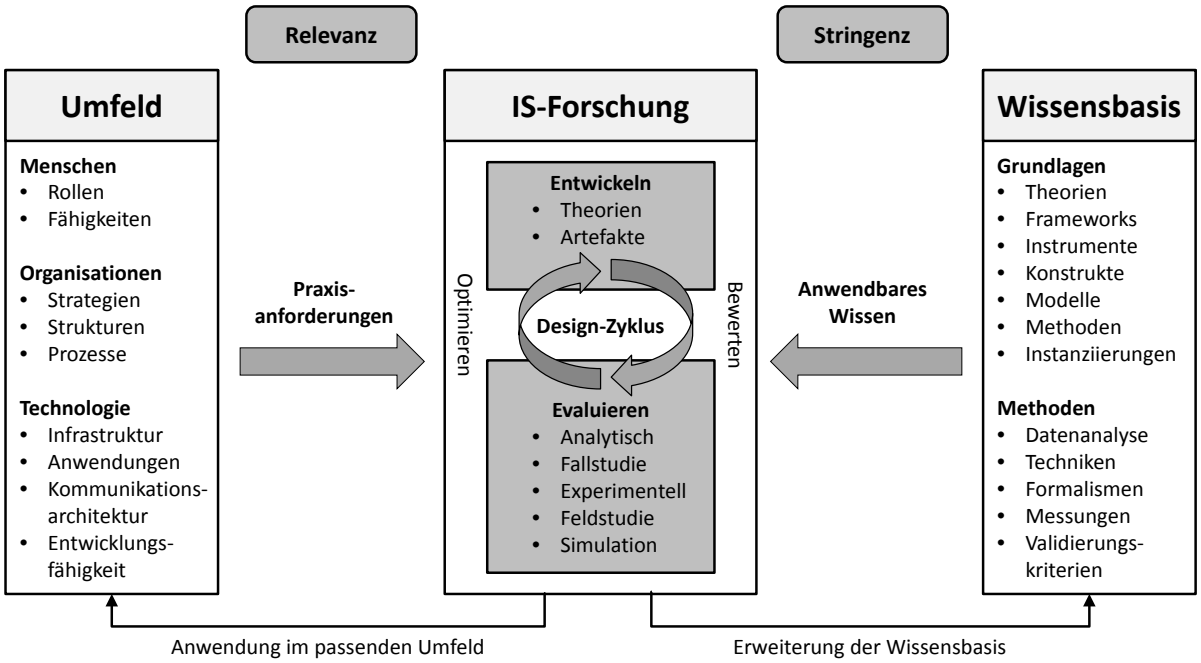

Abbildung 1-2: Rahmenkonzept für gestaltungsorientierte Forschung
Quelle: Eigene Darstellung in Anlehnung an (Hevner et al. 2004, 80)

Übertragen auf die vorliegende Arbeit wird das *Umfeld* zum einen durch Menschen charakterisiert, welche keine Kenntnisse und Erfahrungen im Bereich der Entwicklung mobiler Anwendungen besitzen, sogenannte Endbenutzer (engl. End-User). Diese Personengruppe nutzt ein ERP-System als Unterstützung bei der täglichen Aufgabenbewältigung und besitzt demzufolge entsprechende Kenntnisse in der Nutzung eines solchen Systems und ist mit dessen Terminologie vertraut. Die vorliegende Arbeit konzentriert sich auf das ERP-System der SAP AG (Aktiengesellschaft). Daher können die vom SAP-ERP-System bereitgestellten *Technologien*, insbesondere die existierenden Programmierschnittstellen, genutzt werden, um auf dessen Daten und Funktionalitäten zuzugreifen.

Der Forschungsprozess der vorliegenden Arbeit gliedert sich in Anlehnung an Becker (2010, 13 ff.) in die vier Phasen Analyse, Entwurf, Evaluation und Diffusion. Übertragen auf die vorliegende Arbeit sollen in den einzelnen Phasen folgende Ergebnisse erzielt werden:

- **Analyse:** Erhebung des Stands in der Wissenschaft und Praxis, Ermittlung der Anforderungen
- **Entwurf:** Konzeption und prototypische Implementierung der Softwareartefakte
- **Evaluation:** Bewertung und Ableitung von Verbesserungsvorschlägen
- **Diffusion:** Vorträge und Publikationen

Um den aktuellen Stand der Wissenschaft zu erheben wird in der vorliegenden Arbeit eine *Literaturanalyse* durchgeführt. Um zusätzlich den Stand der Praxis zu ermitteln, werden bei dieser Analyse auch praxisbezogene Literaturquelle einbezogen. Zudem werden im Rahmen einer *Werkzeuganalyse* existierende Entwicklungswerkzeuge für mobile ERP-Applikationen sowie Endbenutzer-Entwicklungswerkzeuge für mobile Applikationen identifiziert und untersucht. Ergänzend hierzu, werden die im Rahmen der Literatur- und Werkzeuganalyse gewonnen Erkenntnisse in zwei Befragungsrunden mit unterschiedlichen Schwerpunkten in Form von *semi-strukturierten Experteninterviews* validiert und erweitert. Schließlich werden aus den insgesamt gewonnen Erkenntnissen, Anforderungen an das zu konzipierende Endbenutzer-Entwicklungswerkzeug abgeleitet.

In der darauffolgenden Entwurfsphase werden zunächst wiederkehrende Elemente mobiler ERP-Applikationen identifiziert sowie deren Verknüpfungsregeln untereinander festgelegt. Hieraus wird anschließend eine *domänenspezifische Sprache* für mobile ERP-Applikationen abgeleitet. Zur Konzeption des Entwicklungswerkzeuges werden die einzelnen Elemente der DSL in wiederverwendbare Softwarebausteine überführt. Zudem werden die Softwarearchitektur und die Benutzungsschnittstelle des auf diesen Bausteinen basierenden Entwicklungswerkzeuges entwickelt.

Um eine kontinuierliche Optimierung des Artefakts zu gewährleisten werden insgesamt drei Design-Zyklen durchlaufen. Im ersten Zyklus werden die Elemente und Verknüpfungsregeln der DSL entwickelt. Zudem wird ein zugehöriger Codegenerator konzipiert und anschließend prototypisch implementiert. Mit Hilfe mehrerer funktionaler- und statischer Tests werden verschiedene Aspekte der domänenspezifischen Sprache und des Codegenerators evaluiert. Anschließend wird ein erster Entwurf für die Benutzungsschnittstelle des Entwicklungswerkzeuges konzipiert. Dieser wird in Form von semi-strukturierten Interviews mit potentiellen Endbenutzern evaluiert. Schlussendlich werden die aus den vorherigen Evaluationen gewonnen Erkenntnisse genutzt, um einen funktionstüchtigen Prototypen des Entwicklungswerkzeuges zu implementieren. Dieser wird schließlich im Rahmen eins kontrollierten Laborexperimentes mit potentiellen Endbenutzern evaluiert. Der beschriebene Forschungsprozess ist in Abbildung 1-3 illustriert.

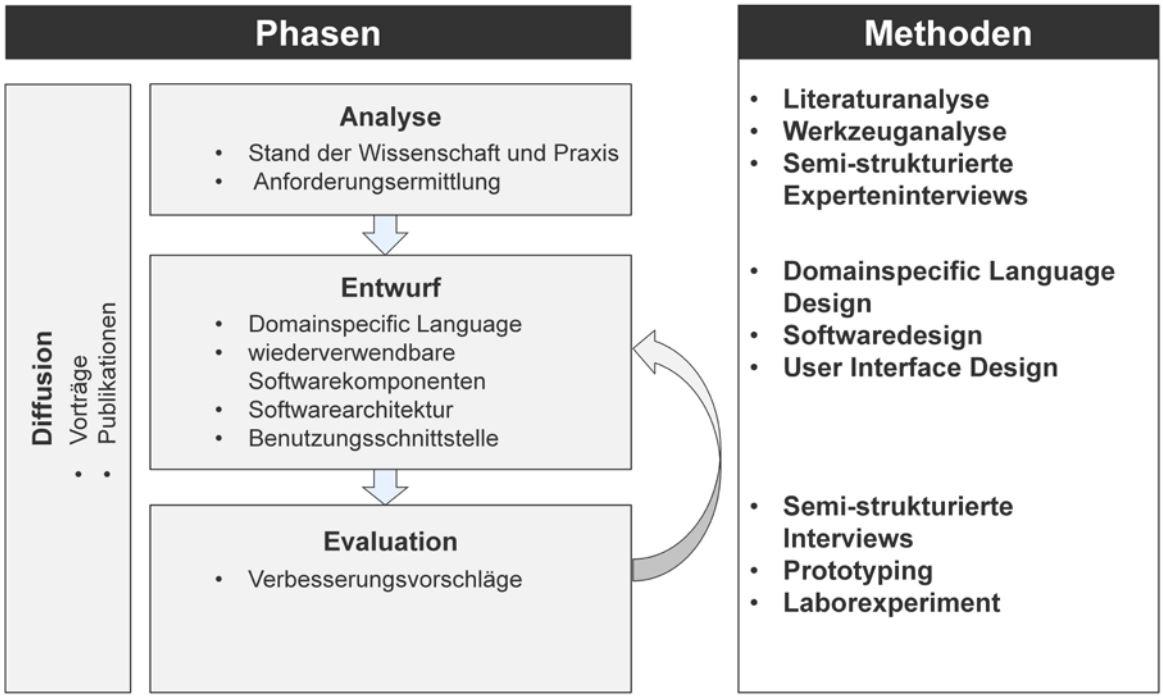

Abbildung 1-3: Forschungsprozess und –methoden
Quelle: Eigene Darstellung

1.5 Limitationen

Diese Arbeit fokussiert sich auf die Werkzeugunterstützung zur Entwicklung von mobilen ERP-Applikationen durch Endbenutzer. Im Rahmen der Nutzung mobiler Applikationen in Unternehmen sind jedoch neben der Entwicklung weitere Aspekte zu berücksichtigen, welche in dieser Arbeit nicht oder nur rudimentär betrachtet werden.

Ein Aspekt ist die Bereitstellung und Verwaltung der mobilen Applikationen nach deren Entwicklung. In der Regel ist es nicht erwünscht, dass jeder Mitarbeiter Zugriff auf jede mobile ERP-Applikation hat. Daher muss ein Bereitstellungs- und Verwaltungsmechanismus umgesetzt werden, mit welchem der Zugriff von Mitarbeitergruppen auf mobile Anwendungen konfiguriert werden kann (Beckert et al. 2012, 72ff.; Mall et al. 2012, 135 ff.). In der vorliegenden Arbeit wird hingegen nur ein rudimentärer Verteilungsmechanismus für die entwickelten Applikationen bereitgestellt.

Ein weiterer Aspekt ist die Gewährleistung der Unternehmenssicherheit. Mobile Endgeräte greifen über mobile Datennetze auf die Backendsysteme des Unternehmens zu. Hierbei muss sichergestellt werden, dass die Datenübertragung hinreichend verschlüsselt wird. Da auf den mobilen Endgeräten i.d.R. sicherheitskritische Daten gespeichert werden, wie beispielsweise Kennwörter oder lokal gespeicherte Geschäftsdaten, müssen diese vor unbefugtem Zugriff geschützt werden. Zusätzlich entstehen durch Trends wie „Bring Your Own Device“ (BYOD) weitere Sicherheitsrisiken. Bei BYOD handelt es sich um eine Organisationsrichtlinie, bei welcher Mitarbeiter private (mobile) Endgeräte für geschäftliche Zwecke nutzen dürfen

(Beckert et al. 2012, 75; Mall et al. 2012, 34). Um dies zu ermöglichen, muss insbesondere der Zugriff auf die Netzwerkdienste und Backendsysteme über diese Endgeräte ermöglicht werden. In diesen Fällen muss sichergestellt werden, dass auch die privaten Endgeräte hinreichend abgesichert sind, so dass kein unbefugter Zugriff auf Geschäftsdaten stattfinden kann (Mall et al. 2012, 130 ff.). Diese Themenbereiche werden in der vorliegenden Arbeit nicht adressiert. In der prototypischen Umsetzung dieser Arbeit werden Kennwörter unverschlüsselt auf einem Webserver abgelegt und auch die Datenkommunikation zwischen den mobilen Applikationen und dem Webserver erfolgt über ein nicht verschlüsseltes HTTP-Protokoll.

Durch den Zugriff mobiler Applikationen auf die Backendsysteme des Unternehmens entsteht zusätzliche Last auf den Backendsystemen. Zudem besitzen mobile Datennetze i.d.R. eine geringere Stabilität und Leistungsfähigkeit als kabelgebundene Datennetze. Daher müssen geeignete Lösungen gefunden werden, um einen leistungsfähigen und skalierbaren Datenverkehr zwischen den mobilen Applikationen und den Backendsystemen zu gewährleisten (Beckert et al. 2012, 74). In der vorliegenden Arbeit werden keine speziellen Konzepte entwickelt, um das Antwortverhalten der entwickelten mobilen ERP-Applikationen zu steigern. Der Zugriff auf das SAP-ERP-System erfolgt über Standardschnittstellen und existierenden Programmierbibliotheken. Mechanismen zur Vermeidung von Abfragen, wie bspw. eine lokale Datenspeicherung auf dem mobilen Gerät, werden ebenfalls nicht berücksichtigt.

Der Fokus dieser Forschungsarbeit liegt auf der technischen Umsetzung des Entwicklungswerkzeuges. Aspekte jenseits der technischen Umsetzung werden nicht betrachtet. Beispiele hierfür sind die Einführung des Werkzeuges in ein Unternehmen oder die für die erfolgreiche Nutzung notwendigen Rahmenbedingungen in einem Unternehmen.

1.6 Aufbau der Arbeit

Ziel der vorliegenden Arbeit ist die Konzeption und prototypische Implementierung eines Entwicklungswerkzeuges zur Entwicklung mobiler ERP-Applikationen durch Endbenutzer. Im ersten Kapitel wird die Problemstellung dargestellt und die Notwendigkeit des angestrebten Entwicklungswerkzeuges motiviert. Daraus wird die Vision der Arbeit abgeleitet und die zugehörige Forschungsarbeit in drei forschungsleitende Fragestellungen gegliedert.

In Kapitel 2 wird die Bedeutung von ERP-Systemen für Unternehmen dargestellt. Darauffolgende werden die Eigenschaften von ERP-Systemen sowie die Vorteile und Herausforderungen beim Einsatz von ERP-Systemen erläutert. Zudem wird die technische Architektur von ERP-Systemen beschrieben. Hierbei liegt der Schwerpunkt auf der Beschreibung der Benutzungs- und Programmierschnittstellen. Aufgrund der Spezifika unterschiedlicher ERP-Systeme erfolgt die Beschreibung am Beispiel des weit verbreiteten ERP-Systems der SAP AG. Im Anschluss an den Überblick über ERP-Systeme wird das Themengebiet „Mobile Computing“ behandelt. Hierbei werden unterschiedliche Klassen mobiler Endgeräte vorgestellt und der Begriff des „mobilen Systems“ eingeführt. Anschließend wird die Geräteklasse „Smartphone“ aufgrund ihrer Bedeutung für die vorliegende Forschungsarbeit detaillierter vorgestellt. Daraufhin werden mobile Unternehmensapplikationen charakterisiert und deren

Einbettung in die Geschäftsprozesse eines Unternehmens erläutert. Bei der Beschreibung mobiler Unternehmensapplikationen wird die Benutzungsschnittstelle als Schwerpunkt dieses Applikationstyps dargestellt. Aus diesem Grund werden anschließend existierende Gestaltungsempfehlungen für die Benutzungsschnittstellen von IT-Systemen im Allgemeinen sowie von Smartphone-Applikationen im Speziellen vorgestellt. Schließlich wird das Themengebiet „Endbenutzer-Entwicklung“ vorgestellt. Neben einer Motivation dieses Entwicklungsansatzes werden Interaktionstechniken, Herausforderungen und zugehörige Lösungsansätze aufgezeigt.

In Kapitel 3 werden Charakteristiken mobiler ERP-Applikationen identifiziert und vorgestellt. Hierbei werden zunächst Charakteristiken aus wissenschaftlichen und praxisnahen Literaturquellen herausgearbeitet. Diese werden anschließend um Charakteristiken ergänzt, welche aus einer Untersuchung existierender mobiler ERP-Applikationen abgeleitet werden.

Kapitel 4 umfasst die Beschreibung von zwei durchgeführten Befragungen mit relevanten Interviewpartnern. Hierbei soll die aktuelle Vorgehensweise bei der Entwicklung mobiler ERP-Applikationen in der unternehmerischen Praxis aufgezeigt werden sowie die hohe Praxisrelevanz der fokussierten Problemstellung fundiert werden. Zudem sollen Anforderungen an das zu entwickelnde Werkzeug abgeleitet werden. Hierzu werden leitfadengestützte Experteninterviews mit Experten aus dem Umfeld mobiler ERP-Applikationen durchgeführt und anschließend ausgewertet.

Kapitel 5 beschäftigt sich mit der Anforderungsermittlung an das umzusetzende Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen. Hierzu werden zunächst theoretische Grundlagen der Anforderungsermittlung beschrieben. Anschließend werden Anforderung an das umzusetzende Entwicklungswerkzeug sowie die damit entwickelbaren mobilen ERP-Applikationen identifiziert und beschrieben. Hierzu wird auf die gewonnen Erkenntnisse aus den früheren Kapiteln der vorliegenden Forschungsarbeit zurückgegriffen.

In Kapitel 6 werden existierende Entwicklungswerkzeuge hinsichtlich der in Kapitel 5 angeführten Anforderungen untersucht. Ziel ist es zum einen, den Bedarf des umzusetzenden Werkzeuges aufzuzeigen. Zum anderen sollen aus der Untersuchung Umsetzungsideen für einzelne Aspekte des eigenen Werkzeuges gesammelt werden. Aufgrund des unterschiedlichen Funktionsumfanges sowie der unterschiedlichen Anwendungsfälle existierender Werkzeuge wird bei der Untersuchung zwischen Entwicklungswerkzeugen für mobile Unternehmensapplikationen und Endbenutzer-Entwicklungswerkzeugen für mobile Applikationen unterschieden.

Die gesammelten Umsetzungsideen werden in Kapitel 7 verwendet, um ein erstes Architekturkonzept für das umzusetzende Entwicklungswerkzeug zu entwickeln. In diesem Zusammenhang wird zudem detaillierter erläutert, wie die weitere Vorgehensweise bei der schrittweisen Weiterentwicklung des vorgestellten Konzeptes und dessen Evaluation ist.

Bestandteil von Kapitel 8 ist die Gestaltung einer domänenspezifischen Sprache für mobile ERP-Applikationen. Hierzu werden zunächst theoretische Grundlagen domänenspezifischer Sprachen behandelt. Anschließend wird die verwendete Vorgehensweise bei der Gestaltung

einer eigenen domänenspezifischen Sprache beschrieben. Zudem werden ausgewählte domänenspezifische Sprachen aus verwandten Forschungsarbeiten vorgestellt und untersucht. Schließlich werden die gewonnen Erkenntnisse genutzt, um eine domänenspezifische Sprache für mobile ERP-Applikationen zu konzipieren.

Kapitel 9 beschreibt die Konzeption des Codegenerators. Als Bestandteil des angestrebten Entwicklungswerkzeuges hat der Codegenerator die Aufgabe, aus der mit Hilfe der domänenspezifischen Sprache beschriebenen Applikationen, lauffähige mobile ERP-Applikation zu generieren. Die Umsetzbarkeit des vorgestellten Konzeptes wird anschließend durch eine prototypische Implementierung demonstriert. Zudem werden Anforderungen an den Codegenerator durch funktionale Tests und statische Analysen evaluiert.

Im nächsten Schritt wird die Architektur des Entwicklungswerkzeuges erweitert. Die Beschreibung dieser Erweiterung ist Bestandteil von Kapitel 10. Schwerpunkt der Erweiterung ist die Konzeption der Benutzungsschnittstelle. Hierzu wird ein erster Entwurf der Benutzungsschnittstelle im Rahmen von semi-strukturierten Interviews mit potentiellen Endbenutzern evaluiert. Die dabei gewonnen Erkenntnisse werden genutzt, um die konzipierte Benutzungsschnittstelle zu verbessern. Die konzipierte Benutzungsschnittstelle wird schließlich umgesetzt und mit dem in Kapitel 9 vorgestellten Codegenerator integriert. Als Resultat entsteht ein funktionstüchtiger Prototyp des Entwicklungswerkzeuges. Mit Hilfe dieses Prototypen werden anschließend die verbleibenden Anforderungen an das Entwicklungswerkzeug evaluiert. Hierbei kommen statische Analysen, funktionale Tests sowie ein umfangreicheres kontrolliertes Experiment mit potentiellen Endbenutzern zum Einsatz.

Im abschließenden Kapitel 11 werden die erzielten Ergebnisse der Arbeit nochmals zusammenfassend dargestellt. Hierbei wird insbesondere der wissenschaftliche und praktische Beitrag der Arbeit thematisiert. Schließlich werden die Limitationen der Arbeit dargestellt und ein Ausblick auf weiteren Forschungsbedarf gegeben.

2 Begriffliche und theoretische Grundlagen

Im folgenden Kapitel werden wichtige Begriffe und theoretische Grundlagen für das Verständnis der vorliegenden Forschungsarbeit erläutert. Dabei wird zunächst der Systemtyp „ERP-System“ beschrieben. Neben allgemeinen Eigenschaften, Vorteilen und Herausforderungen von ERP-Systemen wird die technische Architektur des SAP-ERP-Systems detaillierter vorgestellt. Anschließend werden die Begriffe „mobiles System“ und „mobile Unternehmensapplikation“ definiert und daraus ein Verständnis für den Begriff „mobile ERP-Applikation“ abgeleitet. Aufgrund der zentralen Bedeutung der Benutzungsschnittstelle bei mobilen ERP-Applikationen, werden anschließend verschiedene Gestaltungsempfehlungen für Benutzungsschnittstellen von mobilen Applikationen angeführt und diskutiert. Schließlich werden unterschiedliche Aspekte der Endbenutzer-Entwicklung behandelt. Dies umfasst u.a. die Motivation, Herausforderungen und verfügbare Interaktionstechniken der Endbenutzer-Entwicklung. Abschließend werden Beispiele der Endbenutzer-Entwicklung zur Umsetzung von ERP-Applikationen auf Desktop Computern sowie mobiler Applikationen (ohne ERP-Bezug) beschrieben.

2.1 ERP-Systeme

Der Begriff ERP steht für die englischsprachige Abkürzung „Enterprise Resource Planning“. Hierunter wird die unternehmerische Aufgabe verstanden, die in einem Unternehmen vorhandenen relevanten Ressourcen zu planen (Zelewski et al. 2008, 761). Dabei umfasst die Planungsaufgabe nicht nur den Ressourceneinsatz zur Durchführung von Geschäftsprozessen, sondern auch die kurzfristige Ressourcenbereitstellung sowie die langfristige Ressourcenentwicklung (Zelewski et al. 2008, 761). Beispiele für Ressourcen im Unternehmen sind Kapital, Betriebsmittel oder Personal (Hessler/Görtz 2008, 4). Die Planung von Ressourcen im Unternehmen erfolgt häufig mit Unterstützung von Informationstechnologie (IT). Hierbei werden oftmals sogenannte ERP-Systeme eingesetzt. Da ERP-Systeme i.d.R. auf einem Server im Rechenzentrum des Unternehmens betrieben werden, werden sie auch als sogenannte *Backendsysteme* bezeichnet (siehe z.B. (Martino/Philipp 2012, 33)).

Der Fokus von ERP-Systemen liegt auf der Unterstützung und Optimierung von Geschäftsprozessen (Schwarzer/Krcmar 2010, 82). Dabei decken ERP-Systeme Geschäftsprozesse aus unterschiedlichen Aufgabenbereich ab, beispielsweise der Materialwirtschaft, Produktion, Vertrieb, Kostenrechnung, Finanzbuchhaltung und dem Personalwesen (Kurbel 2010, 4; Schwarzer/Krcmar 2010, 153). Allgemein werden ERP-Systeme als integrierte, betriebswirtschaftliche Standardsoftware bezeichnet (Hessler/Görtz 2008, 2). Daraus lassen sich die Merkmale *Integration*, *betriebswirtschaftliche Orientierung* und *Standardisierung* ableiten, welche im Folgenden erläutert werden. Anschließend werden Vorteile und Herausforderungen von ERP-Systemen diskutiert sowie deren historische Entwicklung geschildert. Danach wird die technische Architektur von ERP-Systemen vorgestellt. Hierbei wird ein Schwerpunkt auf die Beschreibung der Benutzungs- und Programmierschnittstellen gelegt, da diese für die vorliegende Arbeit von besonderem Interesse sind. Bei den Ausführungen wird jeweils

zunächst eine allgemeine Darstellung gegeben, welche anschließend am konkreten Fallbeispiel des ERP-Systems der SAP AG (im Folgenden SAP-ERP-System genannt) beispielhaft veranschaulicht wird. Das Fallbeispiel SAP-ERP wurde gewählt, da entsprechende Softwarelizenzen und Systemzugänge im Rahmen des Lehrstuhlprojektes „SAP University Competence Center“ (SAP UCC) zur Verfügung standen. Aufgrund der marktführenden Stellung von SAP-ERP (vgl. z.B. Pang et al. 2013) kann zudem von einer hohen Relevanz des gewählten Fallbeispiels ausgegangen werden.

2.1.1 Eigenschaften von ERP-Systemen

Im Folgenden werden wesentliche Eigenschaften von ERP-Systemen erläutert.

2.1.1.1 Betriebswirtschaftliche Orientierung

Die betriebswirtschaftliche Orientierung bezieht sich auf die von ERP-Systemen unterstützten Aufgabenklassen. Nach Hessler und Görtz (2008, 9) können Softwaresysteme technische-, betriebswirtschaftliche- oder gemischte Aufgabenklassen unterstützen. Ein Beispiel für eine technische Aufgabenklasse ist das Computer Aided Design (CAD). Betriebswirtschaftliche Aufgabenklassen beziehen sich hingegen auf die Abwicklung von Geschäftsprozessen (Hessler/Görtz 2008, 9). Typische Geschäftsprozesse, welche von ERP-Systemen unterstützt werden, sind die Materialdisposition, die Lieferantenangebotsbearbeitung, die Debitoren- und Kreditorenbuchhaltung oder die Lohn- und Gehaltsabrechnung (Hessler/Görtz 2008, 9).

2.1.1.2 Integration

In größeren Unternehmen kommen verschiedene Applikationssysteme zum Einsatz, die jeweils auf bestimmte Funktionsbereiche, Organisationseinheiten oder Geschäftsprozesse ausgerichtet sind. Ein automatisierter Datenaustausch der eingesetzten Applikationssysteme wird selten direkt unterstützt und muss daher zunächst mit oftmals größeren Aufwänden ermöglicht werden. Erfolgt dieser Datenaustausch jedoch nicht, so fehlt eine integrierte Sicht auf die unterschiedlichen Daten des Unternehmens. Um in diesem Fall ein Gesamtbild der Daten eines Unternehmens sowie der Zustände der Geschäftsprozesse zu erhalten, müssten die Daten aus den unterschiedlichen Applikationssysteme mühselig zusammengestellt werden (Laudon et al. 2010, 479). ERP-Systeme stellen hierfür eine Problemlösung in Form eines „unternehmensweiten und Unternehmensfunktionen integriertem Applikationssystem“ (Krcmar 2010, 236) dar. Dadurch werden komplexe und teure Schnittstellen zwischen verschiedenen Applikationssystemen vermieden (Laudon et al. 2010, 483). Zudem können die in ERP-Systemen abgebildeten Geschäftsprozesse auf eine gemeinsame Datenbasis zugreifen. So können beispielsweise Materialverbrauchsdaten aus der Produktion direkt in der Buchhaltung berücksichtig werden.

2.1.1.3 Standardisierung

Im IT-Bereich wird unter einem *Standard* oftmals eine (technische) Spezifikation durch ein offizielles Gremium verstanden (Buxmann et al. 1999, 157). Im Kontext von ERP-Systemen existiert hingegen ein anderes Begriffsverständnis. Standard bezieht sich hier darauf, dass

ERP-Systeme durch ihren Funktionsumfang und ihre implementierten Geschäftsprozessabläufe eine allgemeine Einsetzbarkeit in unterschiedlichen Unternehmen ermöglichen sollen (Hessler/Görtz 2008, 14). Daher werden ERP-Systeme als *Standardsoftware* bezeichnet (Mertens et al. 2010, 22). Das Gegenstück zu Standardsoftware ist sogenannte *Individualsoftware*. Individualsoftware bezeichnet das Ergebnis einer Eigenentwicklung von Softwaresystemen gemäß den speziellen Anforderungen eines konkreten Unternehmens (Krcmar 2010, 167; Mertens et al. 2010, 24). Dadurch wird ein höherer Abdeckungsgrad der Anforderungen eines Unternehmens erzielt, als dies bei Standardsoftware der Fall ist (Hessler/Görtz 2008, 14). Der höhere Abdeckungsgrad wird jedoch durch höhere Entwicklungskosten erkauft (Mertens et al. 2010, 24).

Standardsoftware wird hingegen für den Massenmarkt und damit den Einsatz in unterschiedlichen Unternehmen zur Abdeckung unterschiedlicher Anforderungen konzipiert und entwickelt (Mertens et al. 2010, 138; Hessler/Görtz 2008, 14). Da die Anforderungen unterschiedlicher Unternehmen jedoch voneinander abweichen und sich teilweise sogar wiedersprechen können, müssen bei der Entwicklung von Standardsoftware bereits programmtechnische Möglichkeiten vorgesehen werden, um den Anpassungsaufwand möglichst gering zu halten (Hessler/Görtz 2008, 14). Der Prozess der Anpassung einer Standardsoftware an unternehmensspezifische Anforderungen wird allgemein als *Customizing* bezeichnet (Mertens et al. 2010, 138). Damit können die Abweichungen zwischen den unternehmensspezifischen Anforderungen und dem Funktionsumfang der Standardsoftware reduziert werden (Mertens et al. 2010, 138). Beim Customizing werden vom ERP-System bereitgestellte Optionen gewählt und Wertzuweisungen von bereitgestellten Parametern vorgenommen. Damit kann beispielsweise die konkrete Unternehmensstruktur im ERP-System hinterlegt werden oder verfügbare ERP-Funktionalitäten aktiviert oder deaktiviert werden (Hessler/Görtz 2008, 224 ff.). Neben dem Customizing sind oftmals weitere Formen der Anpassung möglich. Eine weitere Anpassungsform ist die *Modification*, bei welcher der Programmcode des ERP-Systems an die eigenen Bedürfnisse angepasst wird (Glass 1998, 15; Brehm et al. 2001). Dies kann jedoch zu Problemen bei zukünftigen Aktualisierungen des ERP-Systems führen. Die Anpassung durch sogenannte *Extensions* hat hingegen den Vorteil, dass der eigene Programmcode an definierten Stellen aufgerufen wird und somit der eigentliche Programmcode des ERP-Systems nicht angepasst werden muss (Glass 1998, 15). Die aufwändigste Anpassungsform stellt die *Eigenentwicklung* dar, bei welcher die gewünschten Funktionalitäten implementiert und dem ERP-System hinzugefügt werden (Hessler/Görtz 2008, 230 f.). Neben den Standardfunktionalitäten und den unterstützten Anpassungsformen bieten einige ERP-Hersteller zudem sogenannte *Branchenlösungen* an. Dabei handelt es sich generell nicht um eine eigenständige Lösung, sondern um eine Ergänzung eines ERP-Systems. Diese Ergänzungen werden aufgrund der charakteristischen Besonderheiten von Branchen, welche sich auf die Geschäftsprozesse der Unternehmen auswirken, notwendig (Hessler/Görtz 2008, 61 ff.).

2.1.2 Historische Entwicklung von ERP-Systemen

Als erste Generation betrieblicher Applikationssysteme und somit als Vorgänger von ERP-Systemen gelten die in den sechziger Jahren entwickelten *Material-Requirements-Planning-Systeme*, kurz *MRP-Systeme* (Gronau 2010, 3). MRP-Systeme unterstützen die sogenannte

Materialbedarfsplanung. Dabei wird der Materialbedarf (auch Sekundärbedarf genannt) auf Basis der von der Auftragserfassung, Absatz- und Primärbedarfsplanung zur Verfügung gestellten Anzahl an Endprodukten berechnet (Zelewski et al. 2008, 761; Mertens et al. 2010, 95). Diese Funktionalitäten wurden im Laufe der Jahre um weitere für die Produktion notwendige Funktionalitäten ergänzt, beispielsweise der Kapazitäts- und Terminplanung. Ziel war vor allem Realisierbarkeit der Planung zu überwachen. Dies wurde durch eine Integration von Materialwirtschaft und Kapazitätsplanung erreicht. Dabei wird die Ausführung der Pläne und der daraus resultierende Korrekturbedarf in die Planung rückgekoppelt und die Kapazitätsplanung bereits bei Planerstellung berücksichtigt (Kurbel 2010, 129). Diese Erweiterung führte zu dem neuen Begriff *Manufacturing-Resource-Planning-Systeme*, kurz *MRP-II-Systeme* (Zelewski et al. 2008, 761 f.). MRP-II-Systemen waren jedoch weitgehend auf den Produktionsbereich fokussiert (Zelewski et al. 2008, 762; Kurbel 2010, 129 ff.). Nach und nach wurden jedoch auch andere Unternehmensbereiche in die IT-gestützte Ressourcenplanung integriert, wie beispielsweise das betriebliche Rechnungswesen oder das Personalwesen. Diese integrierte Ressourcenplanung über verschiedene Unternehmensbereich hinweg führte in den neunziger Jahren schließlich zum Begriff *ERP-Systeme* (Zelewski et al. 2008, 761 ff.; Kurbel 2010, 227 ff.).

ERP-Systeme wurden ursprünglich nur innerbetrieblich konzipiert (Laudon et al. 2010, 483 f.). Nach und nach erfolgte jedoch auch eine Integration mit den ERP-Systemen von Lieferanten und Kunden oder sogar über mehrere Wertschöpfungsstufen hinweg (Stahlknecht/Hasenkamp 2005, 327). Um diesem überbetrieblichen Aspekt Rechnung zu tragen wird in der Literatur teilweise der von der Gartner Group geprägte Begriff *ERP-II-Systeme* verwendet (Koh et al. 2008, 4; Gronau 2010, 4). In der Praxis hat sich für die überbetriebliche Integration ein weiterer Systemtyp etabliert, die sogenannten *Supply-Chain-Management* (SCM) *-Systeme*. SCM-Systeme werden nicht als Ersatz, sondern vielmehr als Ergänzung zu ERP-Systemen angesehen (Gronau 2010, 273). Typische Aufgabenstellungen von SCM-Systemen sind die unternehmensübergreifende Bedarfsplanung oder die operative Verknüpfung von einzelnen Beschaffungs-, Produktions-, Versand- und Transportaufträgen in der Ausführungsphase (Kurbel 2010, 5). Ein weiterer Systemtyp sind die sogenannten *Customer-Relationship-Management* (CRM*)-Systeme*. CRM-Systeme unterstützen die Planung, Durchführung, Kontrolle und Anpassung aller Unternehmensaktivitäten, die zu einer Erhöhung der Profitabilität des Kundenportfolios beitragen (Gronau 2010, 296 f.). Sie werden notwendig, da ERP-Systeme oftmals nur eingeschränkte Funktionalitäten zur Verwaltung der Kundenbeziehungen bereitstellen, nämlich diejenigen, bei denen ein Geld- und Leistungsaustausch stattfindet (Gronau 2010, 295). Ebenso wie SCM-Systeme sind auch CRM-Systeme eine Ergänzung zu ERP-Systemen und besitzen sinnvollerweise eine Integrationsbeziehung zu ERP-Systemen.

2.1.3 Vorteile von ERP-Systemen

Die Einführung und Nutzung eines ERP-Systems ist mit einer Reihe von Vorteilen verbunden. Durch die funktionsübergreifende Sicht werden die *Planungsprozesse verbessert*. So ist es dem Produktionsbereich beispielsweise möglich die Bestellungen der Kunden einzusehen und somit die Produktion bedarfsgerechter auszugestalten (Laudon et al. 2010, 489).

Zudem wird eine *einheitlichere Unternehmensstruktur* begünstigt. Der Einsatz von ERP-Systemen bedingt eine Standardisierung der Geschäftsprozesse und Daten über unterschiedliche Geschäftseinheiten und geografische Lokationen hinweg. Diese Vereinheitlichung auf der Ebene von Geschäftsprozessen und Daten unterstützt eine einheitliche Unternehmensstruktur und damit eine effizientere Kommunikation und Planung im gesamten Unternehmen (Laudon et al. 2010, 488). Bedingt durch die durch ERP-Systeme erzielten Standardisierungen wird zudem die *überbetriebliche Integration* begünstigt (Hessler/Görtz 2008, 48).

Durch die von ERP-Systemen bereitgestellten, funktionsübergreifenden, einheitlichen Daten sowie der bereitgestellten Analysewerkzeuge lassen sich Daten besser aggregieren und analysieren. Dadurch werden *Entscheidungsprozesse verbessert* (Laudon et al. 2010, 488 f.).

Die *Lizenzkosten* von ERP-Systemen betragen oftmals nur einen Bruchteil der Entwicklungskosten von Individualsoftware. Jedoch müssen bei einer Kostenbetrachtung auch die Kosten für den notwendigen Anpassungsaufwand von ERP-Systemen einbezogen werden (Hessler/Görtz 2008, 48). Jedoch liegen die Kosten für Wartung und notwendiger Erweiterungen der Software bei den Herstellern der ERP-Systeme. Hierdurch reduzieren sich die Kosten für die Wartung der Software auf das Einspielen von Korrekturen und neuer Funktionen im Rahmen von Versionswechseln.

Ferner ist die *Qualität und Stabilität* von ERP-Systemen i.d.R. höher als bei Individualsoftware. Dies liegt vor allem an der größeren Nutzerzahl, welche sich in verschiedenen Unternehmen befinden. Dadurch werden die Funktionalitäten des ERP-Systems umfassender getestet, als das bei Individualsoftware der Fall wäre (Hessler/Görtz 2008, 48).

2.1.4 Herausforderungen von ERP-Systemen

Neben den Vorteilen von ERP-Systemen ist deren Nutzung auch mit einigen Herausforderungen verbunden. Eine große Herausforderung ist die *aufwendige und riskante Einführung* eines ERP-Systems in ein Unternehmen. Um ein ERP-System nutzen zu können sind oftmals grundlegenden Anpassungen der Arbeitsabläufe notwendig. Zudem müssen unternehmensweit gültige Definitionen von Daten erstellt werden sowie die benötigten Applikationsfunktionalitäten aus den bereitgestellten Modulen des ERP-Systems ermittelt werden. Da von der Einführung eines ERP-Systems i.d.R. alle Kernbereiche eines Unternehmens betroffen sind, besteht ein nicht unerhebliches Risiko für den reibungslosen Ablauf der betroffenen Geschäftsprozesse. (Laudon et al. 2010, 489 f.; Hessler/Görtz 2008, 49).

Aufgrund der aufwändigen und riskanten Einführung eines ERP-Systems versuchen Unternehmen i.d.R. den Umstieg auf ein anderes ERP-System zu vermeiden. Hierdurch ergibt sich ein *Abhängigkeitsverhältnis zum jeweiligen Hersteller des ERP-Systems* und dessen zukünftiger Strategie (Hessler/Görtz 2008, 49).

ERP-Systeme gelten allgemein als *inflexibel*. Durch die enge Verzahnung der Geschäftsprozesse eines Unternehmens mit dem ERP-System können Geschäftsprozesse nicht ohne weiteres angepasst werden und erfordern oftmals eine Anpassung des ERP-System. Dies ist jedoch

aufgrund des hohen Funktionsumfanges und der daraus resultierenden Komplexität von ERP-Systemen keine einfache und risikolose Aufgabe (Laudon et al. 2010, 491; Hessler/Görtz 2008, 49).

Zudem werden oftmals die *hohen Kosten* von ERP-Systemen bemängelt. Als größter Kostenblock gilt allgemein die Einführung eines ERP-Systems. Die zugehörigen Projekte erstrecken sich teilweise über mehrere Jahre. Für eine ERP-System Einführung ist spezielles Expertenwissen notwendig, welches oftmals über externe Berater mit hohen Tagessätzen erworben werden muss. Außerdem muss die Ausbildung der Mitarbeiter finanziert und die Lizenzkosten für das ERP-System bezahlt werden (Laudon et al. 2010, 490 f.).

Weiterhin wird teilweise der Verlust von Wettbewerbsvorteilen bemängelt. Dies wird durch die Standardisierung von Geschäftsprozessen bei der Nutzung von ERP-Systemen begründet. Dadurch wird die spezifische Ausprägung von Geschäftsprozessen, welche zu einem Wettbewerbsvorteil führen kann verhindert oder zumindest reduziert (Laudon et al. 2010, 491).

2.1.5 Technische Architektur von ERP-Systemen am Fallbeispiel SAP-ERP

Im Folgenden wird die technische Architektur von ERP-Systemen beschrieben. Da sich die verfügbaren ERP-Systeme in Ihrer Realisierung voneinander unterscheiden, wird das ERP-System der SAP AG (im Folgenden SAP-ERP genannt) als Fallbeispiel verwendet. Das SAP-ERP-System wurde deshalb gewählt, da dies auch das zugrundeliegende ERP-System der in dieser Arbeit erstellten prototypischen Implementierung des fokussierten Endbenutzer-Entwicklungswerkzeuges ist. Ein weiteres Entscheidungskriterium war die Verfügbarkeit der Softwarelösung und die anfallenden Lizenzkosten. Die vorliegende Dissertation ist in das Lehrstuhlprojekt SAP University Competence Center (UCC) des Lehrstuhls für Wirtschaftsinformatik (I17) der Technischen Universität München eingebettet. Im Rahmen dieses Projektes standen sowohl die Softwarelösung als auch die zugehörigen Lizenzen für das SAP-ERP-System zur Verfügung.

ERP-Systeme basieren auf einem Paket integrierter Softwaremodule (Laudon et al. 2010, 485 f.). Die einzelnen Softwaremodule sind i.d.R. jeweils auf einen bestimmten betriebswirtschaftlichen Aufgabenbereich ausgerichtet, wie beispielsweise auf das Finanz- und Rechnungswesen oder die Personalwirtschaft. Dieser modulare Aufbau ermöglicht die sukzessive Einführung von ERP-Systemen (Mertens et al. 2010, 23; Hessler/Görtz 2008, 100). Die Integration der einzelnen Module wird durch eine gemeinsame Datenbank erreicht (Laudon et al. 2010, 485 f.; Schwarzer/Krcmar 2010, 152). Durch die gemeinsame Datenbank wird zudem eine redundante Speicherung von Daten vermieden (Mertens et al. 2010, 37ff.). Im SAP-ERP-System besitzen die einzelnen Module eine aus zwei Ziffern bestehende Bezeichnung, beispielsweise SD (Sales and Distribution) für das Vertriebswesen oder HR (Human Resources) für die Personalwirtschaft (Hernandez et al. 2005, 30 ff.). Abbildung 2-1 skizziert den modularen Aufbau eines ERP-Systems mit einer zentralen Datenbasis.

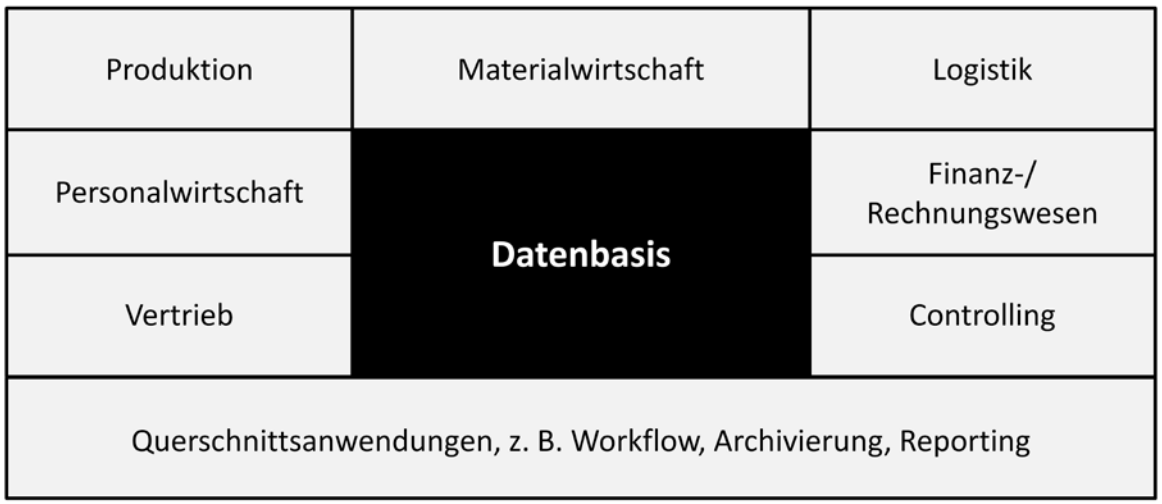

Abbildung 2-1: Modularer Aufbau von ERP-Systeme
Quelle: (Schwarzer/Krcmar 2010, 153)

Unabhängig von Ihrer konkreten Ausprägung sind heutige ERP-Systeme derzeit mehrschichtig aufgebaut und realisieren eine sogenannte Client-Server-Architektur (Gronau 2010, 27 ff.). Bei diesem Architekturtyp wird eine Trennung zwischen Präsentations-, Applikations- und Datenbankschicht durchgeführt. In ERP-Systemen wird oftmals noch eine vierte Schicht, die sogenannte Adaptionsschicht verwendet (Gronau 2010, 9 f.). Diese dient der Anpassung der Standard-Funktionalitäten des ERP-Systems an die unternehmensspezifischen Bedürfnisse. Abbildung 2-2 illustriert die verschiedenen Architekturschichten eines ERP-Systems.

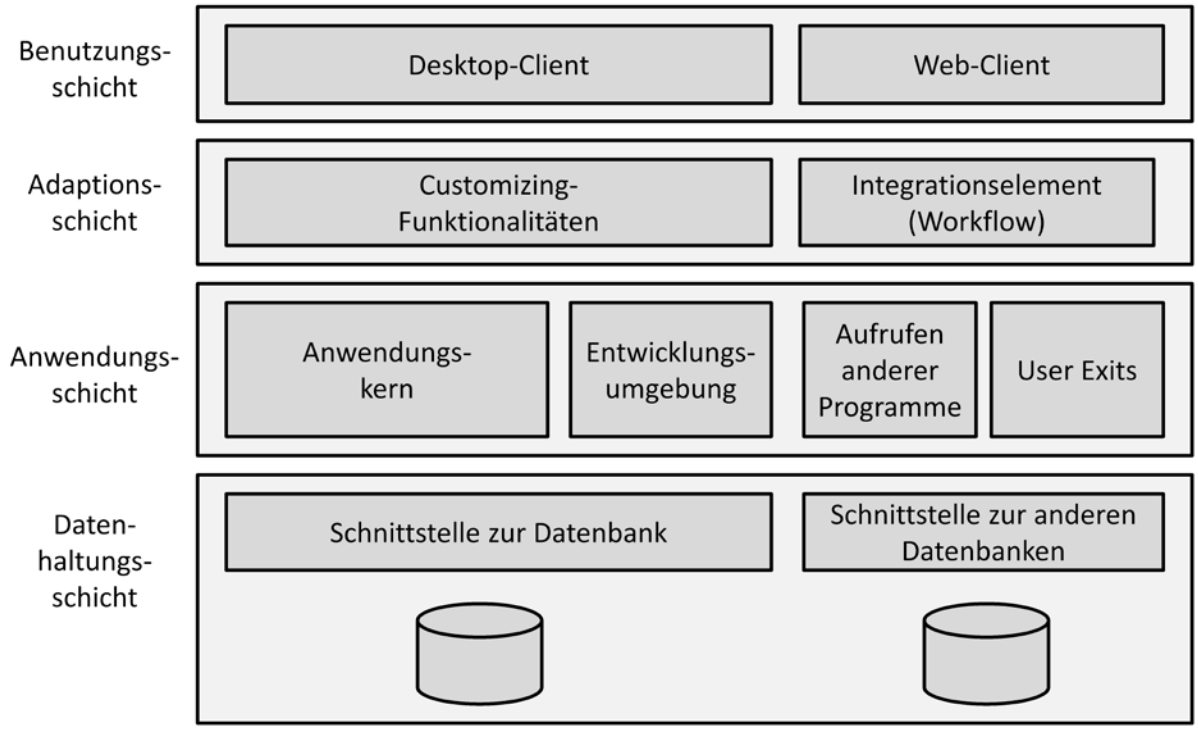

Abbildung 2-2: Schichtenarchitektur eines ERP-Systems
Quelle: (Gronau 2010, 9)

Die Datenbestände von ERP-Systemen werden in einer *Datenbank* gespeichert. Die *Datenhaltungsschicht* enthält neben der Datenbank, eine zugehörige Schnittstelle des ERP-Systems zur Datenbank. Hierzu werden typischerweise relationale Datenbankmanagementsysteme (RDBMS) verwendet. Im Falle von SAP werden RDBMS mehrerer Hersteller, wie beispielsweise von Oracle, IBM oder Microsoft unterstützt. Um diese RDBMS-übergreifende Unterstützung zu ermöglichen hat SAP eine spezielle Schnittstelle spezifiziert, die sogenannte *Open SQL*-Schnittstelle. Diese wird von den Datenbankherstellern implementiert, indem die in Open SQL geforderten Methoden in RDBMS-spezifische Befehle überführt werden (Föse et al. 2008, 54 f.). Zusätzlich zur Datenbankschnittstelle sind bei einigen ERP-Systemen auch Schnittstellen vorhanden, welche den Zugriff auf Datenbanken anderer Informationssysteme zulassen (Gronau 2010, 9 f.).

Die *Applikationsschicht* umfasst im Wesentlichen den Applikationskern sowie oftmals eine spezifische Entwicklungsumgebung. Zusätzlich wird der Aufruf externer Programme über sogenannte Remote Procedure Calls (RPC) (Hansen/Neumann 2005, 853 f.; Gronau 2010, 10) und die Integration von in anderen Programmiersprachen entwickelten Programmen, über sogenannte *User Exits*, unterstützt. Der Applikationskern repräsentiert die implementierte Applikationslogik des ERP-Systems. Er besteht aus den einzelnen ERP-Applikationen zur Implementierung der unterstützten Geschäftsprozesse. Im SAP-ERP-System werden die einzelnen ERP-Applikationen als *Transaktionen* bezeichnet (Forsthuber 2005, 375; Gronau 2010, 43 f.). Der Transaktionsbegriff wird verwendet, da eine ERP-Applikation in SAP immer als sogenannte *Logical Unit of Work* (LUW) durchgeführt wird. Diese überführt das

SAP-ERP-System von einem konsistenten, betriebswirtschaftlichen Zustand in einen anderen konsistenten, betriebswirtschaftlichen Zustand (Keller/Krüger 2006, 809 ff.). ERP-Applikationen werden in SAP-ERP-Systemen mit Hilfe der proprietären Programmiersprache „Advanced Business Application Programming“, kurz ABAP erstellt. Zur Änderung oder Erweiterung der implementierten ERP-Applikationen wird häufig eine spezielle Programmierumgebung bereitgestellt. Im Falle des SAP-ERP-Systems nennt sich diese Programmierumgebung Object Navigator (Keller/Krüger 2006, 60 f.). Der Object Navigator enthält gängige Funktionalitäten einer Entwicklungsumgebung, wie beispielsweise einen Quellcode-Editor oder einen Debugger.

Die *Adaptionsschicht* ermöglicht die Anpassung der Funktionalitäten des ERP-Systems an die unternehmensspezifischen Anforderungen (Gronau 2010, 10). Dies wird unter Anlehnung an die Ausführungen in Kapitel 2.1.1.3 als *Customizing* bezeichnet. Zusätzlich werden häufig sogenannte Workflow Management Systeme (WFMS) zur Integration bereitgestellt. WFMS stellen Funktionalitäten zur Unterstützung der arbeitsteiligen Koordination im Rahmen der Geschäftsprozessdurchführung. Beispiele sind Weiterleitungs- und Benachrichtigungsmechanismen oder das Versenden benötigter Dokumente (Laudon et al. 2010, 713 ff.).

Die oberste Schicht bildet die *Benutzungsschicht*. Diese besteht typischerweise aus einem Desktop-Client oder einem Web-Client. Manche ERP-Systeme bieten auch beide Client-Typen an. Ein Desktop-Client ist eine betriebssystemspezifische Applikation, welche vor ihrer Nutzung zunächst über eine Installationsroutine installiert werden muss. Ein Web-Client wird hingegen direkt über einen Web-Browser bedient und erfordert daher keine vorherige Installation. Dafür ist seine Funktionalität gegenüber einem Desktop-Client jedoch i.d.R. eingeschränkt (Gronau 2010, 10 f.). Die Benutzungsschnittstellen des SAP-ERP-Systems werden im folgenden Kapitel detaillierter vorgestellt.

2.1.5.1 Benutzungsschnittstellen

Die Benutzungsschnittstellen von ERP-Systemen stellen grafische Oberflächen für menschliche Benutzer zur Verfügung, um ihre Funktionalitäten zu nutzen. Bei der Umsetzung dieser Benutzungsschnittstellen setzen die Hersteller von ERP-Systemen auf betriebssystemabhängige Desktop-Applikationen und/oder Webapplikationen ein. Während die betriebssystemabhängigen Desktop-Applikationen zunächst installiert werden müssen, sind die Webapplikationen über einen Webbrowser umgehend lauffähig. Eine Studie von Leyh und Heger (2012) kommt zu dem Ergebnis, dass beide Umsetzungsformen eine Existenzberechtigung besitzen. Bei SAP-ERP-Systemen wird die grafische Benutzungsschnittstelle als „SAP GUI“ bezeichnet, wobei GUI für Graphical User Interface steht.

Die betriebssystemabhängige Benutzungsschnittstelle von SAP-ERP-Systemen existiert aktuell in einer Variante für verschiedene Versionen des Microsoft Windows Betriebssystems, als auch einer in der Programmiersprache Java implementierten Variante für andere Betriebssysteme wie Linux oder dem Apple Betriebssystem MAC OS X. Für die Kommunikation mit dem SAP Applikationsserver wird das sogenannte DIAG-Protokoll verwendet (Föse et al. 2008, 625). Die einzelnen ERP-Applikationen werden in der SAP GUI als Transaktionen

bezeichnet (Forsthuber 2005, 375; Gronau 2010, 43 f.). Sie können entweder über ein visuelles, als Baumstruktur organisiertes Menü selektiert werden oder über die Eingabe eines alphanummerischen Transaktionscodes aufgerufen werden. Anschließend öffnet sich die selektierte ERP-Applikation. In vielen Fällen hat eine solche Applikation den Fokus einzelne Stammdatensätze oder Transaktionsdatensätze des SAP-ERP-Systems zu verändern. Dabei werden typischerweise die CRUD-Methoden unterstützt, d.h. das Anlegen (engl. create), Lesen (engl. read), Aktualisieren (engl. update) sowie das Löschen (engl. delete) von entsprechen Datensätzen. Abbildung 2-3 illustriert eine Bildschirmaufnahme der ERP-Applikation „Terminauftrag anlegen“ innerhalb der SAP GUI für Windows.

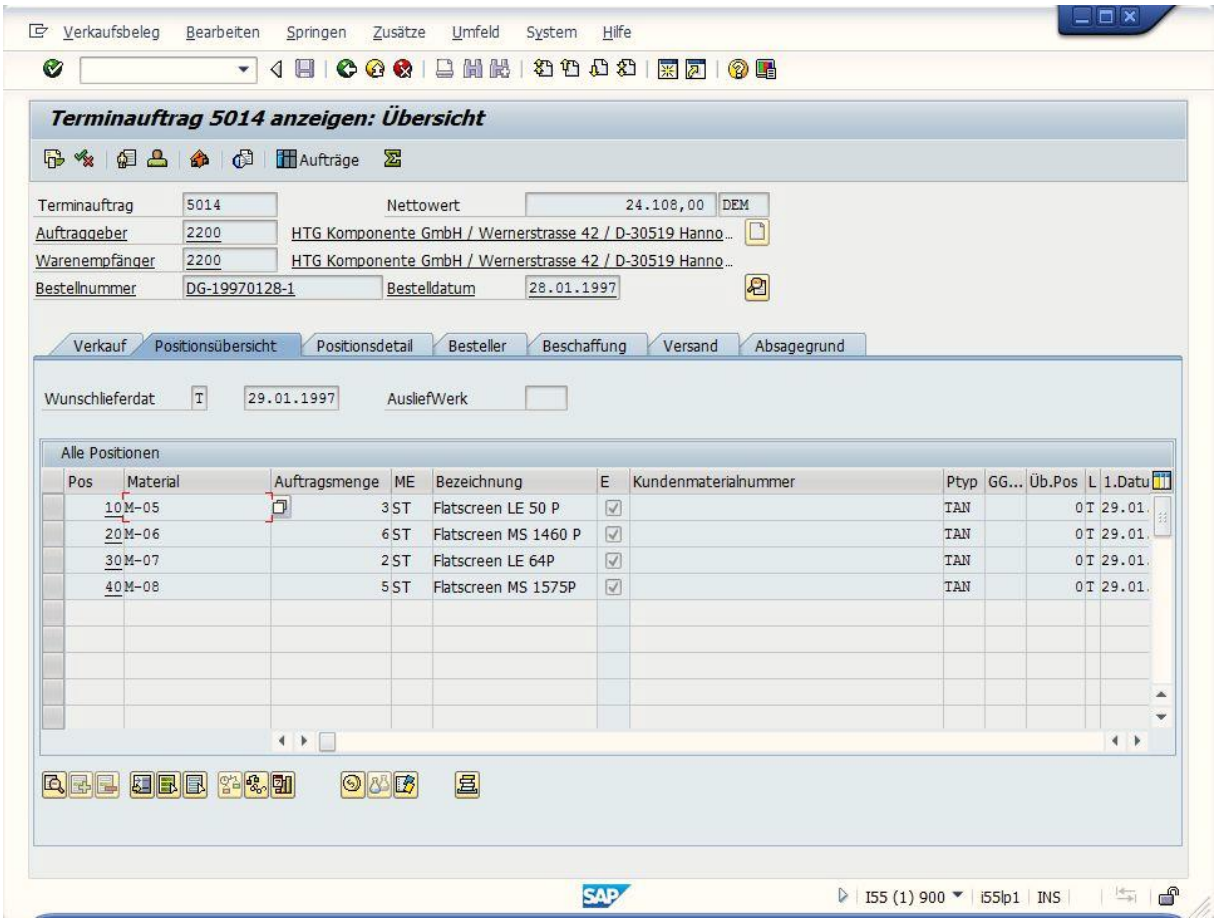

Abbildung 2-3: Beispielhafte ERP-Applikation innerhalb der SAP-GUI für Windows
Quelle: Eigene Darstellung unter Nutzung der SAP GUI für Windows in Version 7.30

Die Webvariante der SAP GUI ist eine auf Basis von Webtechnologien wie HTML und JavaScript umgesetzte Nachbildung der betriebssystemabhängigen SAP GUI Variante. Hierzu wurde im SAP-ERP-System mit dem Internet Transaction Server (ITS) eine spezielle Komponente eingeführt, welche eine Transformation der SAP-ERP-Applikationen in Webtechnologien übernimmt. Abbildung 2-4illustriert die SAP GUI als Webapplikation und verwendet hierfür ebenfalls die beispielhafte ERP-Applikation „Terminauftrag anlegen“.

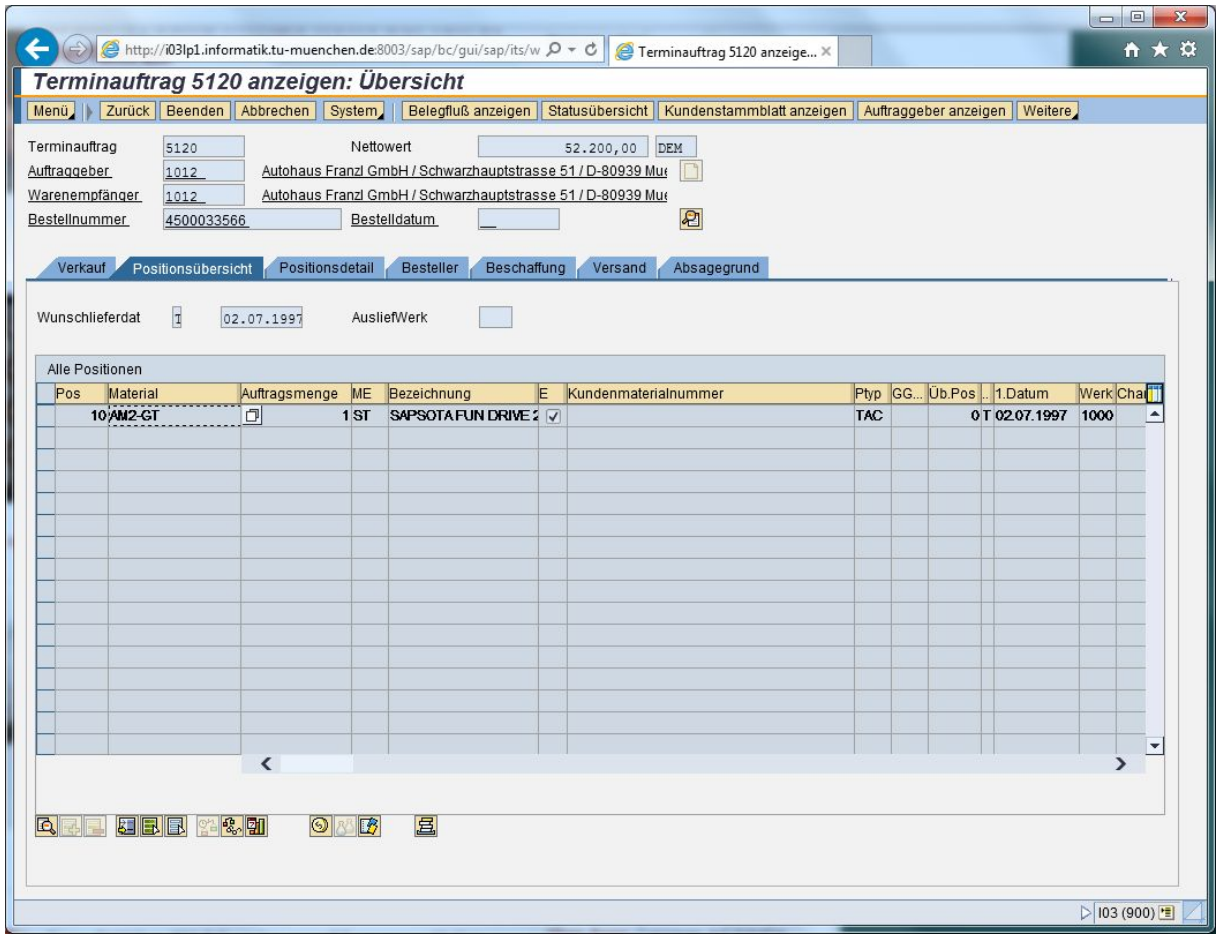

Abbildung 2-4: Beispielhafte ERP-Applikation innerhalb SAP GUI als Webapplikation
Quelle: Eigene Darstellung unter Nutzung des Microsoft Internet Explorers in der Version 10.0

Die ERP-Applikationen der SAP-GUI sind oftmals durch eine Vielzahl von Eingabe- und Ausgabefeldern, Tabulatoren und Tabellen abgebildet und demzufolge oftmals unübersichtlich. Jedoch werden auch Benutzungsschnittstellen anderer ERP-Systeme bemängelt. In zugehörigen Studien werden u.a. folgende Kritikpunkte angeführt (Topi et al. 2005, 130; Singh/Wesson 2009, 89):

- das Auffinden der benötigten Funktionalitäten ist schwierig und aufwändig
- die Unterstützung in Fehlersituationen ist mangelhaft
- die Systemausgaben sind wenig aussagekräftig
- die abgebildete Informationsmenge ist zu hoch
- die Benutzeroberfläche kann zu wenig an die Aufgabendurchführung angepasst werden
- die verwendete Terminologie ist teilweise nicht verständlich
- die allgemeine Systemkomplexität wird als hoch empfunden

2.1.5.2 Programmierschnittstellen

Um auf die Funktionalitäten und Daten eines SAP-ERP-Systems zugreifen zu können, bietet das SAP-ERP-System verschiedene Programmierschnittstellen an. Diese werden zur Integration mit Softwareanwendungen von Drittanbietern, aber auch zum Datenaustausch zwischen verschiedenen SAP-ERP-Systemen verwendet. Die gängigste Programmierschnittstelle sind die sogenannte *Business Application Programming Interfaces* (BAPI). Eine weitere Möglichkeit zum Datenaustausch zwischen SAP-ERP-Systemen und Softwarewendungen von Drittanbietern über sogenannte *Intermediate Documents* (IDoc). Beide Programmierschnittstellen werden im Folgenden erläutert. Dabei liegt der Schwerpunkt jedoch auf den BAPIs, da diese für die vorliegende Forschungsarbeit eine höhere Relevanz besitzen.

2.1.5.2.1 Business Application Programming Interfaces

BAPIs sind Teil der sogenannten Business Framework Architecture (BFA) von SAP-ERP-Systemen (SAP 1998, 5). Die BFA sieht eine Strukturierung der SAP-ERP-Funktionalität in Business-Components und einem Objektmodell vor. *Business-Components* bilden eigenständige funktionale Einheiten und sind die Umsetzung des SAP-ERP-Systems für die in Kapitel 2.1.5 genannten Softwaremodule. Beispiele für Business-Components im SAP-ERP-System sind SAP FI (Financials) für das Finanzwesen, SAP LO (Logistics) für die Logistik oder SAP BASIS für die technologische Plattform des SAP-ERP-Systems (SAP o.J.-a).

Eine Business-Component setzt sich wiederum aus mehreren Business-Objekttypen zusammen. *Business-Objekttypen* repräsentieren in Analogie zur objektorientieren Programmierung eine generische Beschreibung eines Objektes. In objektorientierten Programmiersprachen, wie beispielsweise Java, wird für ein derartiges Konstrukt gewöhnlich der Begriff Klasse verwendet (Ullenboom 2012, 241 ff.). Business-Objekttypen stellen fachliche Entitäten des SAP-ERP-Systems dar, z.B. Mitarbeiter, Materialien oder Kundenaufträge (Gronau 2010, 45; Wegelin/Englbrecht 2011, 134). Die zur Laufzeit erzeugten, spezifischen Instanzen von Business-Objekttypen werden als *Business-Objects* (BO) bezeichnet. Diese enthalten im Gegensatz zu Business-Objekttypen konkrete Datensätze. Um auf die Datensätze eines BO lesend oder schreibend zugreifen zu können, stellt der zugehörige Business-Objekttyp entsprechende Methoden bereit, die sogenannten *Business Application Programming Interfaces*, kurz BAPIs.

Ein Business-Objecttype umfasst insgesamt vier Schichten. Die innerste Schicht wird als *Kern* bezeichnet und enthält die Daten des jeweiligen Business-Objekttypen. Die zweite Schicht, die sogenannte *Integritätsschicht*, beinhaltet die betriebswirtschaftliche Logik des BOs. In der Integritätsschicht werden Geschäftsregeln (engl. Business Rules) und Einschränkungen bzgl. Wertzuweisungen (engl. constraints) implementiert. Die dritte Schicht ist die sogenannte *Schnittstellenschicht*. Sie ermöglicht den Zugriff auf Daten eines BO in Form von Methoden, den BAPIs. Diese Schicht stellt also gewissermaßen die Schnittstelle zur Außenwelt dar. Die vierte Schicht wird als *Zugriffsschicht* bezeichnet. Diese enthält Definitionen für den Zugriff von Applikationen außerhalb des SAP-ERP-Systems über verschiedene Technologien, wie beispielsweise die Common Object Request Broker Architecture (CORBA) oder das Distributed Component Object Model (DCOM). Aufgrund der übergreifenden Gültigkeit

werden diese Zugriffsinformationen derzeit nicht pro Business-Objekttype definiert, sondern nur einmalig pro SAP-ERP-System und sind für sämtliche BOs des jeweiligen SAP-ERP-Systems gültig (Gronau 2010, 46 f.; SAP o.J.-a). Abbildung 2-5 veranschaulicht die beschriebenen Schichten eines Business-Objekttyps.

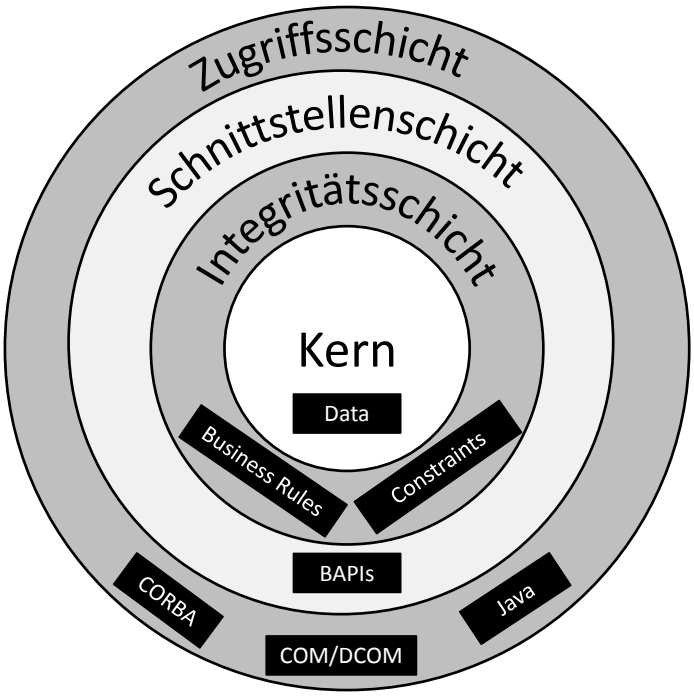

Abbildung 2-5: Schichtenarchitektur von SAP-ERP Business-Objekttypen
Quelle: In Anlehnung an (SAP 1998, 10)

Ein Business-Objecttype besitzt eine Klassendefinition, welche diejenigen Daten eines BO vorgibt, die durch einen BAPI-Aufruf direkt verändern werden können. Demzufolge können alle anderen Daten im SAP-ERP-System nicht von diesem BAPI geändert werden. Zusätzlich gehört zu einem BAPI die Schnittstellendefinition. Diese erfolgt durch die Angabe von Import-, Export- sowie Import-/Exportparametern. Importparameter repräsentieren die Übergabeparameter eines BAPI-Aufrufes; Exportparameter repräsentieren die zugehörigen Rückgabeparameter. Import-/Exportparameter sind als Referenz-Datentyp implementiert und können sowohl die Rolle eines Übergabeparameters als auch die eines Rückgabeparameters einnehmen.

Da das SAP-ERP-System nicht von Anfang an objektorientiert aufgebaut wurde, erfolgt die Implementierung von BAPIs über sogenannte Funktionsbausteine (FuBa). Ein FuBa ist ein wiederverwendbarer Programmcode in der Programmiersprache ABAP, welcher von einem anderen Programm oder Programmteil aufgerufen werden kann (Keller/Krüger 2006, 107; Gronau 2010, 46). Damit auf ein BAPI auch außerhalb des SAP-ERP-Systems zugegriffen werden kann, muss der zugehörige FuBa RFC-fähig sein. RFC steht für Remote Function Call und repräsentiert ein proprietäres Netzwerkprotokoll der SAP (Gronau 2010, 47; Wegelin/Englbrecht 2011, 26). Für RFC gibt es eine Reihe von Programmierbibliotheken, um

RFC-Aufrufe über verschiedene Programmiersprachen durchführen zu können. Eine der am weitesten verbreitete Variante ist der sogenannte Java Connector, kurz JCo (Wegelin/Englbrecht 2011, 279 ff.).

2.1.5.2.2 Webservices

Eine weitere von SAP-ERP-Systemen unterschützte Programmierschnittstelle sind sogenannte Webservices. Im Gegensatz zur RFC-basierten Kommunikation nutzen Webservices offene Internetprotokolle (Snell et al. 2002, 1 ff.). Offen bedeutet in diesem Zusammenhang, dass die Protokolle von einem herstellerübergreifenden Gremium, wie dem World Wide Web Consortium (W3C) festgelegt wurden. SAP-ERP-Systemen unterstützten grundsätzlich zwei verschiedene Typen von Webservices: SOAP-basierte Webservices und RESTful Webservices.

SOAP steht für Simple Object Access Protocol. Es bezeichnet einen vom W3C festgelegten Standard zum XML-basierten Nachrichtenaustausch auf Basis verschiedener Transportprotokolle. Beispiele für unterstützte Transportprotokolle sind das File Transfer Protocol (FTP), das Simple Mail Transfer Protokoll (SMTP) oder das Hypertext Transfer Protokoll (HTTP). Das am häufigsten verwendetet Transfer-Protokoll ist HTTP. SOAP nutzt die Extensible Markup Language (XML) zur Repräsentation der Daten. Das SAP-ERP-System besitzt komfortable Assistenten, mit welchen aus einem existierenden BAPI oder FuBa SOAP-basierte Webservices generiert werden können. Hierbei wird ein sogenanntes WSDL (Web Service Description Language)-Dokument generiert, welches den Webservice definiert. In diesem WSDL-Dokument werden beispielweise der Servicename, sein Uniform Resource Locator (URL), seine verwendeten Datentypen und bereitgestellten Methoden beschrieben (Wegelin/Englbrecht 2011, 401 ff.; Snell et al. 2002, 11 ff.).

RESTful-Webservices haben im Gegensatz zu SOAP-basierten Webservices keine formale Beschreibung und benötigen aufgrund ihres einfacheren Aufbaus weniger Netzwerkbandbreite (Richardson/Ruby 2007, 49 ff.). In SAP-ERP-Systemen werden sie über eine zusätzliche Komponente, dem sogenannten NetWeaver Gateway unterstützt. Diese Komponente kann als Erweiterung (Add-On) in einen bestehenden SAP-ERP NetWeaver Application Server ABAP (AS ABAP) installiert werden. NetWeaver Gateway stellt RESTful-Webservices als sogenannte Gateway Services für andere SAP-ERP-Systeme oder Softwareapplikationen von anderen Herstellern bereit. Dabei wird das offene Internetprotokoll OData verwendet. Ein Gateway Service basiert ebenso wie ein SOAP-basierte Webservice in einem SAP-ERP-System auf einem BAPIs oder einem FuBa. Jedoch ist das Ziel von Gateway-Services deren Komplexität auf das notwendige Maß zu reduzieren. Deshalb werden beispielsweise nur für den jeweiligen Einsatzzweck notwendige Import- oder Export-Parameter über einen Gateway-Service bereitgestellt. Zusätzlich ist durch die Verwendung des offenen OData-Protokolls eine Kommunikation mit Softwareapplikationen von anderen Herstellern einfacher umsetzbar als bei Verwendung des proprietären RFC-Protokolls (Homann et al. 2013b, 44 ff.).

2.1.5.2.3 Application Link Enabling und Intermediate Documents

Bei Application Link Enabling (ALE) handelt es sich um eine Programmierschnittstelle zur Dokumenten-basierten Kommunikation zwischen verschiedenen SAP-ERP-Systemen sowie zwischen einem SAP-ERP-System und Softwareapplikationen anderer Hersteller. Die dabei ausgetauschten Dokumente sind sogenannte Intermediate Documents (IDocs). Diese bestehen aus Zeilen, die sogenannten Segmente. Zudem besitzen sie einen IDoc-Typ. Die Kommunikation erfolgt dabei asynchron, d.h. der Sender wartet mit der Fortführung seiner Aktivitäten nicht bis die Antwort des Empfängers eingegangen ist. Bei der Umsetzung eines IDoc-basierten Kommunikationsszenarios muss ein IDoc-Typ definiert sowie jeweils eine Programmkomponente für die Erzeugung der IDocs und deren Empfang implementiert werden. Schließlich muss das ALE-Subsystem des SAP-ERP-Systems für die Nutzung der entwickelten Programmkomponenten konfiguriert werden. Für die Implementierung der Programmkomponenten können verschiedene Programmiersprachen, beispielsweise ABAP, Java oder C, verwendet werden (Wegelin/Englbrecht 2011, 355 ff.).

2.1.6 Zusammenfassung und Fazit

Aufgrund ihres hohen Funktionsumfangs und des umfangreichen Datenmodells können ERP-Systeme als komplexe Softwaresysteme angesehen werden. Aufgrund ihres hohen Verbreitungsgrades sind sie kritisch für den Geschäftserfolg. Da sie eine große Anzahl von Geschäftsprozessen in Unternehmen unterstützen, sind sie auch für mobile Applikationsszenarien interessant. Die verwendeten Softwarearchitekturen, Technologien und Implementierungscharakteristiken von ERP-Systemen sind stark vom jeweiligen Hersteller abhängig (Gronau 2010, 27 ff.). Daher wird in der vorliegenden Dissertation das SAP-ERP-System als beispielhaftes ERP-System verwendet. Es gibt verschiedene Arten um auf Daten und Funktionalitäten eines SAP-ERP-Systems zuzugreifen. Die Grundlage bilden jedoch in allen Fällen sogenannte BAPIs oder FuBas, welche in der proprietären Programmiersprache ABAP entwickelt werden.

2.2 Mobile Computing

Derzeit existiert eine hohe Heterogenität und Dynamik im Bereich mobiler Geräte. Als Folge existiert keine einheitliche, konsistente Taxonomie zur Klassifikation mobiler Geräte (Gansemer et al. 2007, 699). Um ein Begriffsverständnis für diese Forschungsarbeit zu schaffen, werden zunächst die verschiedenen Ausprägungsformen von *Mobilität* dargestellt. Anschließend werden Charakteristiken mobiler Endgeräte beschrieben, welche sie insbesondere von stationären Desktop-Computern unterscheiden. Schließlich wird die in der Dissertation fokussierte Endgeräteklasse „Smartphone“ detaillierter beschrieben. In diesem Zusammenhang wird auch dargestellt, was im Rahmen der vorliegenden Forschungsarbeit unter einer mobilen Applikation auf einem Smartphone verstanden wird.

2.2.1 Ausprägungsformen der Eigenschaft Mobilität

Je nach eingenommener Perspektive können mobile Einheiten von Informationssystemen verschiedene Ausprägungsformen der Eigenschaft Mobilität besitzen. In der Literatur wird in diesem Zusammenhang zwischen Endgeräte-, Benutzer- und Dienstmobilität unterschieden (Roth 2005, 7 f.).

2.2.1.1 Endgerätemobilität

Endgerätemobilität beschreibt die räumliche Beweglichkeit von Computergeräten, bei welcher die Netzwerkkonnektivität während der Bewegung aufrecht erhalten wird (Roth 2005, 7). In Anlehnung an Gorlenko und Merrick (2003, 640 ff.) kann der Grad der Endgerätemobilität durch die Portabilität des jeweiligen Computergerätes, der Kontinuität der Kommunikationsverbindung und der Notwendigkeit einer Ablagefläche bestimmt werden. Die Eigenschaft *Portabilität* beschreibt in diesem Zusammenhang die Transportfähigkeit eines Computergerätes. Die Portabilität ist dabei vor allem abhängig vom Gewicht, den Abmessungen - dem sogenannten *Formfaktor* - (Salmre 2005, 12) und der notwendigen Verkabelung eines Computergerätes (Gorlenko/Merrick 2003, 640 ff.). Bei der Endgerätemobilität wird jedoch zusätzlich die Kommunikationsverbindung betrachtet. In diesem Zusammenhang wird zwischen einer *kontinuierlichen Endgerätemobilität* und einer *diskreten Endgerätemobilität* unterschieden (Küpper et al. 2007, 69 ff.). Bei einer drahtlosen Kommunikationsverbindung liegt i.d.R. eine kontinuierliche Kommunikationsverbindung vor, d.h. die Kommunikationsverbindung bleibt auch während des Ortswechsels bestehen. Ein Beispiel hierfür sind Mobiltelefone, deren Infrastruktur die ortsunabhängige Nutzung von Telefongesprächen ermöglicht (Roth 2005, 7). Eine diskrete Kommunikationsverbindung liegt üblicherweise bei drahtgebundenen Kommunikationsverbindungen vor. Hierbei ist eine Kommunikationsverbindung nur an festen Zugangspunkten möglich, beispielsweise über LAN-Switche in verschiedenen Räumen.

2.2.1.2 Benutzermobilität

Benutzermobilität bezeichnet die Möglichkeit eines Benutzers, die gewünschten Dienste von beliebigen Geräten und Netzwerken zu nutzen (Roth 2005, 7). Eine notwendige Voraussetzung dieser Eigenschaft ist die eindeutige Identifizierung des Benutzers (Küpper et al. 2007, 71 f.). In Unternehmensnetzwerken wird beispielsweise durch sogenannte Verzeichnisdienste, wie dem Active Directory von Microsoft, gewährleistet, dass sich Mitarbeiter von beliebigen Computern im Firmennetzwerk anmelden können und ihnen anschließend ihre personalisierte Arbeitsumgebung zur Verfügung steht.

2.2.1.3 Dienstmobilität

Dienstmobilität beschreibt den personalisierten Zugriff auf Dienste von beliebigen Orten. Ein Beispiel für eine Dienstmobilität ist der Zugriff auf ein E-Mail-Konto. Benutzer benötigen i.d.R. lediglich ihre Zugangsdaten sowie eine E-Mail Applikation, um geräte- und ortsunabhängig auf ihr E-Mail-Konto zugreifen zu können (Roth 2005, 7). Eine solche E-Mail Appli-

kation steht häufig u.a. auch als webbasierte Applikation zur Verfügung und benötigt somit keine separate Installation.

2.2.2 Eigenschaften und Klassen mobiler Endgeräte

Analog zur Definition Krannich (2010, 37) wird in dieser Arbeit unter einem mobilen Endgerät ein singuläres mit Prozessoren ausgestattetes elektronisches Gerät verstanden, welches folgende Charakteristiken besitzt:

- es besitzt einen drahtlosen Zugang zu einer Netzwerkverbindung
- es wird mittels Batterie(n) betrieben
- es kann an jeden beliebigen Ort transportiert werden
- es kann während des Transports (ohne Stützfläche) genutzt werden
- es verfügt über integrierte Ein- und Ausgabemodalitäten (z.B. Bildschirm, Tastatur, etc.)
- es vereint alle Komponenten in einem Gehäuse

In dieser Arbeit wird die Ausprägungsform Endgerätemobilität zur Differenzierung der verschiedenen Klassen mobiler Endgeräte verwendet. Wie im vorherigen Kapitel beschrieben wird der Grad der Endgerätemobilität von der Portabilität des Gerätes, der Kontinuität seiner Kommunikationsverbindung sowie der Notwendigkeit einer Ablagefläche bestimmt. Nachfolgend werden diese Kriterien beschrieben:

Portabilität: Die Portabilität bzw. Transportfähigkeit eines mobilen Endgerätes wird von seinem Gewicht, seinem Formfaktor, der Anzahl seiner Komponenten und den evtl. notwendigen Transportvorbereitungen bestimmt, wobei der Formfaktor durch die Abmessungen des mobilen Endgerätes festgelegt wird (Salmre 2005, 12). Dabei wird die Transportfähigkeit nicht nur vom mobilen Endgerät alleine bestimmt, sondern auch von dessen notwendigen Peripheriegeräten (z.B. Tastatur, Computermaus, etc.). Zu den Transportvorbereitungen gehören beispielsweise das Abstecken der Verkabelung oder die Beschaffung von Transportmitteln (Krannich 2010, 39 f.).

Ablagefläche: Mobile Endgeräte können in der Regel mit einer oder zwei Händen getragen und gleichzeitig genutzt werden. Bei Desktop-Computern ist im Gegensatz dazu eine größere Ablagefläche notwendig, um neben dem Computer selbst die Tastatur, Maus und den Monitor unterzubringen (Krannich 2010, 40).

Konnektivität: Die drahtlose Netzwerkfähigkeit über Wireless-LAN (W-LAN) oder über Mobilfunktechnologien wie beispielsweise UMTS (Universal Mobile Telecommunications System) oder LTE (Long Term Evolution) ist ein wesentliches Merkmal mobiler Endgeräte im Gegensatz zum kabelgebundenen Netzwerkzugang stationärer Computergeräte (Krannich 2010, 40).

Auf Basis der jeweiligen Ausprägung dieser Eigenschaften können in Anlehnung an Gorlenko und Merrik (2003, 641 f.) zwischen den folgenden mobilen Geräteklassen unterschieden werden: Desktop-Computer, Laptops, Tablets, Smartphones und Wearables. Um die aktuelle Entwicklung im Bereich mobiler Geräte wiederzugeben wurde die ursprünglich von Gorlenko

und Merrik verwendete Geräteklasse Palmtops durch die Geräteklasse Tablets ersetzt sowie die Geräteklasse Handhelds durch die Geräteklasse Smartphones. Abbildung 2-6 illustriert die unterschiedlichen Ausprägungsstufen der Endgerätemobilität verschiedener Geräteklassen.

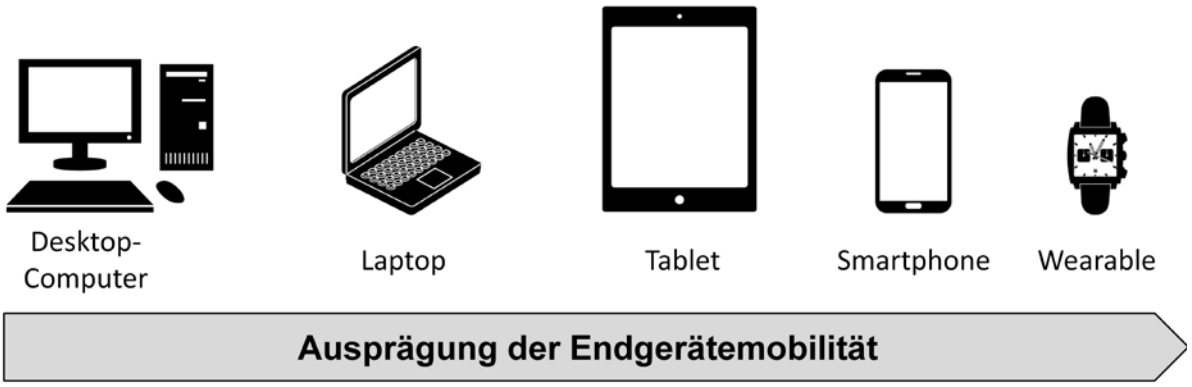

Abbildung 2-6: Ausprägung der Endgerätemobilität verschiedener Geräteklassen

Quelle: In Anlehnung an (Gorlenko/Merrick 2003, 642)

Desktop Computer sind durch ihr vergleichsweise hohes Gewicht charakterisiert. Zudem benötigen sie eine Reihe von Peripheriegeräten zur Eingabe und Ausgabe von Daten, wie beispielsweise eine Tastatur, eine Maus, einen Bildschirm oder einen Lautsprecher. Sie benötigen zusätzlich einen Zugang zu einer Stromquelle sowie einen kabelgebundenen Zugang zu einem Computernetzwerk. Aus diesen Gründen haben Sie eine vergleichsweise geringe Portabilität (Gorlenko/Merrick 2003, 641).

Laptops sind im Gegensatz zu Desktop Computern ohne zusätzliche Peripheriegeräte nutzbar. Sie besitzen oftmals einen Akku, welcher eine Nutzung ohne direkten Zugang zu einer kabelgebundene Stromquelle ermöglicht. Zudem besitzen Laptops ein vergleichsweise geringes Gewicht. Ferner haben heutige Laptops i.d.R. einen integrierten WLAN-Adapter, welcher einen kabellosen Zugang zu einem Funknetzwerk ermöglicht (Gorlenko/Merrick 2003, 641).

Tablets bzw. *Tablet-PCs* sind in der Regel leichter als Laptops. Während Laptops eine feste Unterlage zur Nutzung benötigen, können Tablets hingen auch ohne Unterlage bedient werden. Sie sind ebenfalls mit einem integrierten WLAN-Adapter ausgestattet und besitzen oftmals auch die zusätzliche Möglichkeit auf ein Mobilfunknetz zuzugreifen. Die Tastatur wird bei derzeitigen Tablets bei Bedarf über das Display eingeblendet, weshalb die Notwendigkeit einer separaten Hardwaretastatur entfällt. Während ältere Geräte noch über einen separaten Stift bedient wurden, werden aktuelle Geräte wie das iPad von Apple oder das Galaxy Tab von Samsung über Multi-touch-Display bedient, welches Fingergesten erkennt (Beckert et al. 2012, 46; Homann et al. 2013b, 27).

Heutige *Smartphones* haben ähnliche Charakteristiken wie Tablets, besitzen jedoch einen kleineren Formfaktor. Während Tablets in einer Tasche oder einem Rucksack transportiert werden müssen, passen gängige Smartphones hingegen in die Hosentasche. Zusätzlich besitzen Smartphones im Gegensatz zu Tablets eine Telefoniefunktion und dienen daher als Ersatz für Mobiltelefone. Ein weiteres Unterscheidungsmerkmal zu klassischen Mobiltelefonen sind integrierte Sensoren wie beispielsweise Bewegungs-, Lage-, Licht- oder Näherungssensor. Dadurch können unterschiedliche Datenwerte aus der Umgebung und der aktuellen Nutzungsbedingung ausgewertet werden, wodurch sich neue Applikationsfälle eröffnen. Derartige Sensoren sind oftmals auch in Tablets vorhanden. Beispiele für derzeit weit verbreitete Smartphones sind das iPhone von Apple oder die Galaxy Baureihe von Samsung (Beckert et al. 2012; Homann et al. 2013b, 26 f.).

Unter *Wearables* werden Computergeräte verstanden, welche klein und leicht genug sind, um am menschlichen Körper getragen werden zu können (Gorlenko/Merrick 2003, 642). Wearables benötigen keine Unterstützung jenseits des menschlichen Körpers und besitzen somit den höchsten Grad der Endgerätemobilität der betrachteten Geräteklassen (Gorlenko/Merrick 2003, 642). Im Unterschied zu den anderen Geräteklassen liegt nicht die Bedienung des Wearables im Fokus;. stattdessen wird die eigentliche Arbeitsaufgabe durch das Wearable proaktiv unterstützt (Krannich 2010, 34).

2.2.3 Mobile Systeme

Ein *mobiles System* besteht aus einem mobilen Endgerät und mindestens einer *mobilen Applikation*, welche auf dem mobilen Endgerät läuft (Krannich 2010, 37). Dabei wird zwischen *Einzwecksystemen* und Mehrzwecksystemen unterschieden. Einzwecksysteme sind mobile Systeme, die auf einen bestimmten Zweck ausgerichtet sind. *Mehrzwecksysteme* können hingegen für unterschiedliche Aufgabenstellungen verwendet werden (Krannich 2010, 37).

Ein mobiles System ist mehrschichtig aufgebaut (Krannich 2010, 38). Abbildung 2-7 illustriert die vier Schichten eines mobilen Systems. Auf unterster Ebene befindet sich das mobile Endgerät selbst. Die darüber liegende Ebene stellt die Netzwerkkonnektivität sicher. Abhängig vom mobilen Endgerät stehen hierbei unterschiedliche Verbindungsmöglichkeiten zur Verfügung, beispielsweise über WLAN oder über Mobilfunkstandards wie UMTS oder LTE für Mobilfunknetze. Darüber befindet sich das *mobile Betriebssystem*. In Anlehnung an Tanenbaum (2002, 13) ist es die Aufgabe des Betriebssystems die verschiedenen Hardwarekomponenten des mobilen Endgerätes zu verwalten und eine einfache, abstrahierte Schnittstelle für die Applikationsschicht zur Verfügung zu stellen. Im Folgenden wir unter einem mobilen Betriebssystem, ein Betriebssystem für mobile Endgeräte verstanden. Gängige mobile Betriebssysteme für Tablets und Smartphones sind derzeit Android von der Open Handset Alliance, iOS von Apple oder Windows Phone von Microsoft. Auf der obersten Schicht befinden sich die eigentlichen Applikationen des mobilen Systems. Diese stellen die Schnittstelle zum menschlichen Nutzer dar. Sie stellen dem Nutzer bestimmte Funktionalitäten zur Verfügung, wie beispielsweise den Empfang und das Versenden von E-Mails, das Navigieren World Wide Web (WWW) über einen Webbrowser oder das Telefonieren.

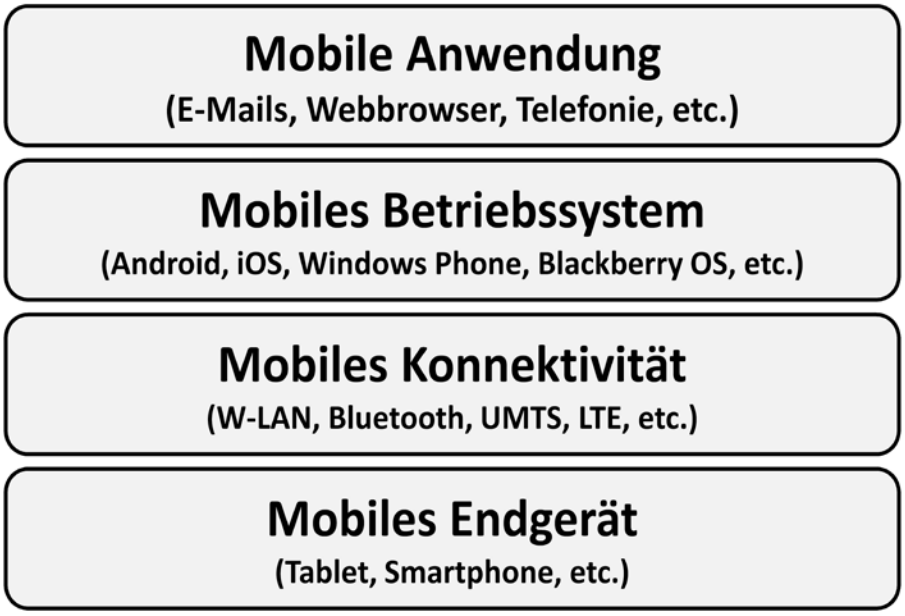

Abbildung 2-7: Architektur mobiler Systeme
Quelle: Eigene Darstellung in Anlehnung an (Krannich 2010, 38)

2.2.4 Smartphones

Als *Smartphones* werden Mobiltelefone bezeichnet, welche im Gegensatz zu klassischen Mobiltelefonen zusätzliche Computerfunktionalitäten bereitstellen (Zheng/Ni 2006, 4 ff.). Diese zusätzlichen Computerfunktionalitäten werden u.a. durch eine Reihe integrierter Sensoren, wie beispielsweise einem Bewegungs-, Lage-, Magnetfeld-, Licht- oder Näherungssensor, ermöglicht (Beckert et al. 2012, 44). Während bei klassischen Mobiltelefonen das Telefonieren im Fokus steht, ist dies bei Smartphones lediglich eine Funktionalität unter vielen. Weit verbreitete Smartphones sind das iPhone von Apple oder das Galaxy S von Samsung. Im Gegensatz zu klassischen Mobiltelefonen besitzen Smartphones vergleichsweise große, hochauflösende Displays. Seit dem Erscheinen des ersten iPhone im Jahr 2007 haben sich Multi-Touch Displays durchgesetzt. Bei Multi-Touch Displays handelt es sich um berührungsempfindliche Displays. Sie ermöglichen die Eingabe von Daten und Manipulation der aktuellen Anzeige mit Hilfe sogenannte Fingergesten. Dabei lösen festgelegte Bewegungsabläufe mit einem oder mehreren Fingern bestimmte Aktionen aus (Saffer 2008, 1 ff.).

Neben den genannten Sensoren besitzen heutige Smartphones i.d.R. auch eine Kamera, welche neben der Fotographie auch die Aufnahme von Videos ermöglicht. Hierdurch ist auch die Videotelefonie über Smartphones möglich (Zheng/Ni 2006, 4 ff.). Im Gegensatz zu Desktop-Computern oder Laptops sind Smartphones schneller Einsatzbereit. Während Desktop-Computer und Laptops nach ihrer Nutzung i.d.R. ausgeschaltet werden, werden Smartphones gewöhnlich nur in Ausnahmefällen ausgeschaltet. Dadurch entfällt bei Smartphones

bei einer gewünschten Nutzung das notwendige Starten und Initialisieren des Betriebssystems, wodurch eine schnelle Einsatzbereitschaft gegeben ist (Ballard 2007, 12 f.).

Smartphones sind im Verständnis dieser Arbeit mobile Systeme, da sie i.d.R. mit einem mobilen Betriebssystem und einer gewissen Anzahl von Standard-Applikationen ausgeliefert werden. Die zugehörigen Betriebssysteme stellen gewöhnlich ein offenes Application Programming Interface (API) zur Verfügung. Dabei handelt es sich um eine Programmierschnittstelle, mit welcher zusätzliche Applikationen für das Smartphone erstellt werden können. Für diesen Zweck stellen die Hersteller der mobilen Betriebssysteme oftmals eigene Entwicklungswerkzeuge und Programmierbibliotheken bereit. Die resultierenden Applikationen werden oftmals als sogenannte „Apps“ bezeichnet (Mertens et al. 2010, 22). Um Applikationen für Smartphones zu verteilen hat sich durch sogenannte *App Stores* ein neues Bereitstellungsmodell etabliert. App Stores ermöglichen ein bequemes suchen, installieren und aktualisieren von mobilen Applikationen (Beckert et al. 2012, 44).

Die Entwicklung mobiler Applikationen für Smartphones unterscheidet sich in vielen Aspekten von der Entwicklung von Applikationen für Desktop-Computer. Dies liegt zum einen an den Geräten selbst. Durch ihre kleineren Displays muss der Inhalt anders angeordnet werden, als bei Applikationen für Desktop-Computer. Beispielsweise ist bei Smartphones zu einem bestimmten Zeitpunkt immer nur eine Applikation sichtbar, während bei Desktop-Computer das gleichzeitige Anzeigen mehreren Applikationen, teilweise auf mehreren Bildschirmen mittlerweile zur gängigen Praxis gehört. Zum anderen erfordert die Steuerung über Fingergesten eine anders Applikationsdesign als bei traditionellen Desktop-Applikationen, welche i.d.R. über eine Hardware-Tastatur und eine Computermaus bedient werden (Homann et al. 2013b, 31 ff.).

Neben den Gerätecharakteristiken sind auch die Applikationsfälle in einem mobilen Kontext anders als in den traditionellen Büroumgebungen von Desktop-Computern. In einer Büroumgebung hat ein Anwender typischerweise lange Interaktionszeiträume mit einer Applikation. Hingegen nutzen Anwender Smartphones oftmals nur für sehr kurze Zeiträume, beispielsweise um eine spezifische Information zu erhalten oder eine Wartezeit mit einer Aktivität zu überbrücken. Die Bedienung von mobilen Applikationen kann zudem parallel zu anderen Aktivitäten erfolgen, z.B. während des Laufens. Zudem ist die Wahrscheinlichkeit von Ablenkungsereignissen im mobilen Kontext höher als in einer Büroumgebung (Homann et al. 2013b, 33).

2.2.5 Eigenschaften und Klassen mobiler Unternehmensapplikationen

Mittlerweile existieren zahlreiche Beispiele für erfolgreiche mobile Applikationen im Unternehmensumfeld. Teilweise wird in diesem Zusammenhang auch vom sogenannten „mobile Business“ gesprochen (Zobel 2001, 3; Lehner 2003, 6 f.). Beispiele sind zugehörige mobile Unternehmensapplikationen sind u.a.(Beckert et al. 2012, 147 ff.; Mall et al. 2012, 161 ff.):

- Mobile CRM Applikationen, z.B. Zugriff auf Kundendaten oder die Daten der Tourenplanung

- Mobile Sales Applikationen, z.B. zur Erfassung von Kundenaufträgen
- Mobile Business Intelligence bzw. mobile Dashboard Applikationen, z.B. zur grafischen Visualisierung wichtiger Unternehmenskennzahlen
- Mobile Workflow Applikationen, z.B. für die Genehmigung von Urlaubs- oder Dienstreiseanträgen
- Mobile E-Mail Clients zum Empfangen und Senden von E-Mails über das mobile Endgerät

Mobile Applikationen besitzen eine Reihe von Eigenschaften, welche Sie insbesondere von Applikationen auf Desktop-Computern unterscheiden. Teilweise sind diese Eigenschaften direkt mit den Eigenschaften mobiler Geräte verbunden. Folgende Eigenschaften zeichnen mobile Applikationen aus (Lehner 2003, 10 ff.; Salmre 2005, 14 ff.):

- **Fokussierung:** Mobile Applikationen sind in der Regel auf einen bestimmten Verwendungszweck ausgerichtet.
- **Kostengünstigkeit:** Aufgrund ihrer klaren Ausrichtung sind mobile Applikationen weniger komplex und daher kostengünstiger in ihrer Entwicklung.
- **Personalisierung:** Mobile Geräte gehören in der Regel einer bestimmten Person und werden selten geteilt. Daher können Applikationen auf die persönlichen Präferenzen des Besitzers ausgerichtet werden.
- **Sofortige Verfügbarkeit:** Mobile Endgeräte sind i.d.R. nahezu durchgehend in Betrieb. Mobile Applikationen auf dem Gerät sind daher umgehend verfügbar.
- **Identifizierbarkeit und Erreichbarkeit:** Mobile Geräte sind durch eindeutige Merkmale wie ihrer SIM (Subscriber Identity Module)-Karte oder ihrer MAC (Media Access Control)-Adresse eindeutig identifizierbar. In Kombination mit dem zuvor angeführten nahezu durchgehenden Betrieb mobiler Geräte haben mobile Applikationen eine hohe Erreichbarkeit. Ein Applikationsbeispiel ist das Senden von Hinweismeldungen an mobile Applikationen (engl. push notifications).
- **Ortsunabhängigkeit:** Mobile Applikationen sind unabhängig von ihrem Standort einsetzbar. Lediglich eine funktionierende Netzwerkverbindung wird von einigen mobilen Applikationen vorausgesetzt.
- **Lokalisierbarkeit:** durch Technologien wie dem Global Positioning System (GPS) kann der Standort mobiler Geräte mittlerweile relativ genau bestimmt werden. Hierdurch werden mobile Applikationen ermöglicht, welche lokationsbezogene Dienste (engl. location based services, kurz LBS) anbieten. Beispiele für LBS sind Routenplaner oder ortsbezogene Werbung.

Mobile Applikationen im Geschäftsumfeld besitzen unterschiedliche Ziele und Zielgruppen. Daraus lassen sich unterschiedliche Klassen von mobilen Applikationen ableiten. In Bezug auf ihre Orientierung können mobile Applikationen in extern- und intern orientierte Applikationen unterschieden werden (Lehner 2003, 1 ff.). Extern orientiere mobile Applikationen

können wiederrum hinsichtlich ihrer Zielgruppe in mobile Applikationen für Kunden und mobile Applikationen für Geschäftspartner, wie beispielsweise Lieferanten, unterschieden werden. Im Falle von mobilen Applikationen für Geschäftspartner wird von „Business-to-Business“ (B2B) mobilen Applikationen gesprochen; im Falle von mobilen Applikationen für Kunden von „Business-to-Consumer“ (B2C) mobilen Applikationen (Lehner 2003, 2 f. und 14 ff.). ERP-Systeme wurden in Kapitel 2.1 als IT-Systeme zur Verwaltung von Unternehmensressourcen definiert. Diese Ressourcenverwaltung wird gewöhnlich von Mitarbeitern verantwortet. Demzufolge haben ERP-Systeme ebenso wie mobile ERP-Applikationen eine interne Orientierung.

2.2.6 Mobile Geschäftsprozesse

Nach Becker et al. (2008, 6) ist ein *Prozess* eine „inhaltlich abgeschlossene, zeitliche und sachlogische Folge von Aktivitäten, die zur Bearbeitung eines betriebswirtschaftlich relevanten Objektes notwendig sind“. Durch ihre Ausführung werden eine oder mehrere Inputfaktoren in eine oder mehrerer Outputfaktoren transformiert (Krcmar 2010, 141). Bei einem *Geschäftsprozess* handelt es sich um einen speziellen Prozess, „der der Erfüllung der obersten Ziele der Unternehmen (Geschäftsziele) dient und das zentrale Geschäftsfeld beschreibt“ (Becker et al. 2008, 6 f.). Wie bereits in Kapitel 2.1.1.1 beschrieben sind typische von ERP-Systemen unterstützte Geschäftsprozesse, die Materialdisposition, die Lieferantenangebotsbearbeitung, die Debitoren- und Kreditorenbuchhaltung oder die Lohn- und Gehaltsabrechnung (Hessler/Görtz 2008, 9).

In Bezug auf den Ausführungsort von Geschäftsprozessen kann zwischen *stationären*- und *mobilen Geschäftsprozessen* unterschieden werden (Gruhn et al. 2005; Mladenova et al. 2011). Dabei handelt es sich in Anlehnung an Gruhn et al. (2005) um einen mobilen Geschäftsprozess, wenn ein gewisser Grad an Unsicherheit bzgl. des Ausführungsortes einzelner Aktivitäten des Geschäftsprozesses gegeben ist. Demzufolge handelt es sich bei stationären Geschäftsprozessen um Geschäftsprozesse, deren Aktivitäten an einem vorher bekannten Ausführungsort durchgeführt werden (Gruhn et al. 2005). Eine Aktivität repräsentiert dabei ein Ergebnis einer Prozesshierarchisierung. Geschäftsprozesse können im Rahmen einer vertikalen Prozessauflösung bzw. Prozesshierarchisierung aufgegliedert werden (Krcmar 2010, 142 f.). Dabei entstehen verschiedene Subprozesse und Aktivitäten. *Aktivitäten* repräsentieren hierbei die letzte Auflösungsebene. Sie stellen Bestandteile eines Geschäftsprozesses dar, deren weitere Unterteilung nicht mehr zweckmäßig ist und welche durch eine einzelne Person durchgeführt werden (Gruhn et al. 2005).

Durch den technologischen Fortschritt und die große Akzeptanz mobiler Endgeräte und deren Applikationen versuchen auch Unternehmen die Potenziale von mobilen Technologien zu nutzen. Eine Möglichkeit dabei ist, ausgewählte Aktivitäten von Geschäftsprozessen durch mobile Applikationen zu unterstützten. In diesem Zusammenhang wird oftmals von der „Mobilisierung“ der jeweiligen Geschäftsprozesse gesprochen (siehe z.B. (Linnhoff-popien/Verclas 2012, 4)). Die Unternehmen erhoffen sich hierdurch einen Mehrwert, beispielsweise durch eine Erhöhung der Effizienz ihrer Mitarbeiter, geringere Fehlerraten oder eine Erhöhung der Mitarbeiterzufriedenheit (Niemann 2011, 12; Gronau et al. 2012, 25 f.).

Doch nicht alle Geschäftsprozesse eignen sich gleichermaßen für eine „Mobilisierung“ (Pousttchi/Becker 2012, 15). Es stellt sich jedoch die Frage, welche Geschäftsprozesse durch die Unterstützung durch mobile Applikationen einen Mehrwert stiften und somit für die Unterstützung durch mobile Applikationen geeignet sind (engl. mobile eligibility) (Mladenova et al. 2011).

Mladenova et al. (2011) schlagen ein zweistufiges Verfahren zur Auswahl geeigneter Aktivitäten für die „Mobilisierung“ vor. Im ersten Schritt werden die potenziellen Aktivitäten für die „Mobilisierung“ identifiziert. Hierbei werden die Aktivitäten der mobilen Mitarbeiter (engl. mobile workforce) analysiert. Beispiele für mobile Mitarbeiter, welche ihre geschäftlichen Tätigkeiten i.d.R. außerhalb des Firmengebäudes ausführen sind Vertriebsmitarbeiter oder Servicetechniker. Es werden jedoch auch explizit andere Tätigkeitsprofile mit einbezogen, beispielsweise Führungspositionen oder Tätigkeitsprofile ohne festen Büroarbeitsplatz, wie beispielsweise Lagermitarbeiter. Nach der Identifikation potenzieller Aktivitäten werden diese in einem zweiten Schritt mit Hilfe eines Kriterienkataloges bewertet. Als Ergebnis entsteht eine priorisierte Liste an geeigneten Aktivitäten für die „Mobilisierung“

Typische Aktivitäten, welche derzeit bereits von Unternehmen durch mobile Applikationen unterstützt werden sind u.a. (Lehner 2003, 1; Beckert et al. 2012, 147 ff.; Mall et al. 2012, 161 ff.):

- Suche im Produktkatalog des Unternehmens
- Erfassung von Kundenaufträgen und Überprüfung der Lieferfähigkeit vor Ort
- Abruf anstehender Termine und zugehöriger Unterlagen durch Wartungsteams
- Koordination von Kundenbesuchen
- Abruf von Kundeninformationen für ein Kundengespräch vor Ort
- Überwachung von Statusinformationen, z.B. von Maschinen oder Softwaresystemen
- Abfrage und grafische Anzeige wichtiger Geschäftskennzahlen und Berichte
- Genehmigung von Urlaubs- oder Dienstreiseanträgen

2.2.7 Zusammenfassung und Fazit

Moderne mobile Geräte wie Smartphones bieten durch ihre integrierten Technologien und ihre stetige Verfügbarkeit ein großes Potential im geschäftlichen Umfeld. Jedoch ist nicht jede geschäftliche Aktivität gleichermaßen geeignet, um durch eine Applikation auf einem Smartphone unterstützt zu werden. Auch die Gestaltung mobiler Applikationen unterscheidet sich in vielen Aspekten von klassischen Applikationen auf Desktop-Computern. Die wichtigsten Unterscheidungsmerkale sind der Funktionsumfang und die Benutzungsschnittstelle auf einem Multi-Touch Display mit vergleichsweise kleinem Formfaktor. Zudem wird die Entwicklung von mobilen Applikationen auf Smartphones durch fehlende Standards und die hohe Marktdynamik erschwert.

2.3 Gestaltungsempfehlungen für die Benutzungsschnittstelle von Smartphone-Applikationen

Ein wesentliches Erfolgskriterium von Computer Systemen im Allgemeinen ist die Gestaltung der Benutzungsschnittstelle (Nielsen 1993, 1 ff.). Dies gilt für Smartphone-Applikationen im besonderen Maße, da mobile Anwender i.d.R. kein aufwändiges Einarbeiten in eine Applikation akzeptieren und wenig Geduld mit Applikationen haben, welche nicht-intuitiv bedienbar sind (Nielsen/Budiu 2013, 44; Clark 2010, 8 ff.). Für den Erfolg einer Benutzungsschnittstelle für eine Smartphone Applikation sind in der Regel andere Aspekte entscheidend, als für den Erfolg einer Benutzungsschnittstelle für eine traditionelle Desktop Applikation (Ballard 2007, 5 ff.; Fling 2009, 109 ff.).

Bei der Gestaltung von Benutzungsschnittstellen müssen eine Reihe von Problemstellungen gelöst werden. Da manche Problemstellungen wiederholt auftreten, haben sich verschiedene Formen von wiederverwendbaren *Gestaltungsempfehlungen* etabliert. Deren Ziel ist das Explizieren von wiederverwendbarem Wissen für wiederkehrende Problemstellungen bei der Gestaltung von Benutzungsschnittstellen (van Welie 2001, 16). Es existieren sowohl Gestaltungsempfehlungen für Benutzungsschnittstellen von Computer Applikationen im Allgemeinen als auch für Benutzungsschnittstellen von Smartphone Applikationen im Speziellen. In Bezug auf ihre Gültigkeit können diese in systemübergreifende und systemspezifische Gestaltungsempfehlungen unterteilt werden. Unter einem System wird in diesem Zusammenhang der im vorherigen Kapitel eingeführte Begriff des mobilen Systems, als eine Kombination aus Gerätehardware und zugehöriger Software verstanden. Systemspezifische Gestaltungsempfehlungen werden oftmals als *Guidelines* bezeichnet. Ein weiteres Unterscheidungsmerkmal ist das Beschreibungsformat der Gestaltungsempfehlungen. Dabei kann zwischen *Gestaltungsprinzipien* und *Entwurfsmustern* (engl. patterns) unterschieden werden. Während Gestaltungsprinzipien kein festgelegtes Beschreibungsformat besitzen, wird die Beschreibung von Entwurfsmustern gemäß einer festgelegten Struktur vorgenommen. Abbildung 2-8 illustriert die unterschiedlichen Arten von Gestaltungsempfehlungen für die Benutzungsschnittstelle von Smartphone-Applikationen.

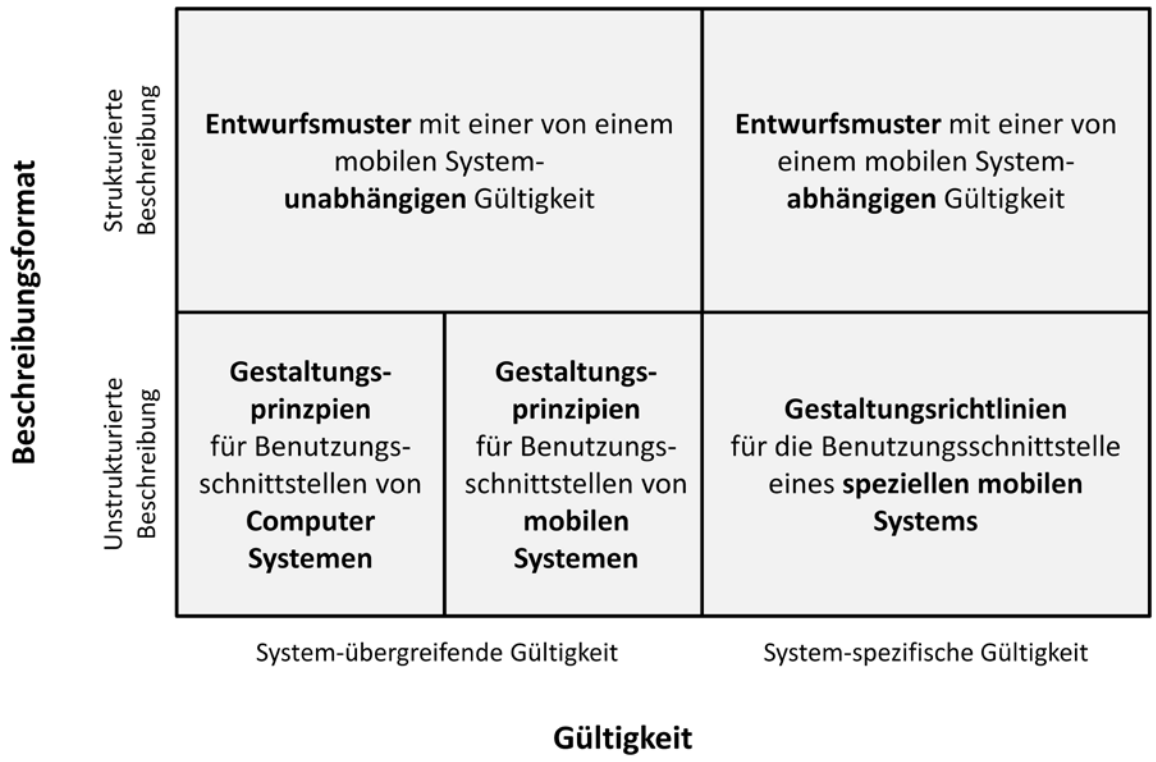

Abbildung 2-8: Gestaltungsempfehlungen für die Benutzungsschnittstelle von Smartphone-Applikationen
Quelle: Eigene Darstellung

2.3.1 Gestaltungsprinzipien

Gestaltungsprinzipien (engl. principles) (Shneiderman/Plaisant 2010, 80 ff.) bzw. abstrakte Gestaltungsrichtlinien (engl. abstract guidelines) (Borchers 2001, 6) stellen eine abstrakte Form von Gestaltungsempfehlungen zur Erreichung einer benutzungsfreundlichen Applikation dar. Beispiele für Gestaltungsprinzipien sind die „eight golden rules of interface design" von Shneiderman (z.B. in Shneiderman/Plaisant 2010, 88 f.), die „Usabillity Heuristiken" von Nielsen (1993, 115 ff.) oder die ISO 9241-110 Dialog Prinzipien (DIN 2006). Tabelle 2-1 führt die in den angeführten Quellen enthaltenen Gestaltungsprinzipien auf.

Tabelle 2-1: Ausgewählte Gestaltungsprinzipien
Quelle: In Anlehnung an (van Welie 2001, 16 f.)

Shneiderman	Nielsen	ISO 9241-10
Strebe nach Konsistenz	Konsistenz	Aufgabenangemessenheit
Stelle fortgeschrittenen Nutzern Shortcuts zur Verfügung	Shortcuts	Selbstbeschreibungsfähigkeit
Gebe informative Rückmeldungen	Rückmeldungen	Steuerbarkeit
Gestalte Dialoge so, dass ein bewusstes Beenden erkennbar ist	Vermeide Fehler / Gute Fehlermeldungen	Erwartungskonformität
Stelle eine Fehlervorbeugung und eine einfache Fehlerhandhabung zur Verfügung	Verzeihe dem Nutzer	Fehlertoleranz
Erlaube das Widerrufen von Aktionen	Reduziere die Belastungen für das Kurzzeitgedächtnis des Nutzers	Individualisierbarkeit
Gebe dem Nutzer die Kontrolle über das System	Einfache und natürliche Dialoge	Lernförderlichkeit
Reduziere Belastungen für das Kurzzeitgedächtnis des Nutzers	Hilfe und Dokumentation	
	Spreche die Sprache der Nutzer	
	Eindeutige Kennzeichnung des Endes einer Aktivität	

Gestaltungsprinzipien sind geeignet, um einen suboptimalen Gestaltungsentwurf zu identifizieren (Borchers 2001, 6). Aufgrund ihres relativ hohen Abstraktionsniveaus ist es jedoch oftmals nicht ersichtlich, welches konkrete Gestaltungsproblem sie adressieren (van Welie 2001, 94 f.). Zudem stellen sie keine Lösungsvorschläge für konkrete Gestaltungsprobleme dar (Borchers 2001, 6). Als Folge ist es für Designer oftmals nicht möglich, mit ihrer Hilfe ein konkretes Gestaltungsproblem zu lösen (Mahemoff/Johnston 1998, 132; van Welie 2001, 94 f.).

Die „Mobile Design Principles“ von Ballard (Ballard 2007, 69 ff.) stellen ein Beispiel für Gestaltungsprinzipien für Benutzungsschnittstellen von mobilen Systemen dar. Ein enthaltenes Gestaltungsprinzip trägt die Bezeichnung „User Context“. Es empfiehlt so viele Informationen über den aktuellen Nutzungskontext des Anwenders zu verwenden wie möglich. Beispielsweise sollten Sensordaten genutzt werden, ebenso wie die Daten aus anderen Applikationen wie der Kalender-Applikation - falls dies erlaubt und im konkreten Fall sinnvoll ist (Ballard 2007, 82 f.).

2.3.2 Gestaltungsrichtlinien

Gestaltungsrichtlinien (engl. concrete guidelines) stellen spezifische Gestaltungsvorschläge für ein spezielles Computer System dar (Borchers 2001, 6). Im Gegensatz zu Gestaltungsprinzipien beziehen sich Gestaltungsrichtlinien demzufolge auf ein konkretes Computer System und sind somit nur eingeschränkt auf andere Computer Systeme übertragbar. Gestaltungsrichtlinien werden oftmals vom Hersteller des Computer Systems erstellt. Ziel von Gestaltungsrichtlinien ist es möglichst einheitliche Benutzungsschnittstellen über möglichst viele Applikationen des entsprechenden Computer Systems zu gewährleisten. Dies ermöglicht es Nutzern sich schneller in neuen Applikationen zurechtzufinden. Beispielsweise wird in Gestaltungsrichtlinien festgelegt, welche Funktionalitäten mit welchen Bedienelementen abgebildet werden, wie Bedienelemente angeordnet werden sollen und teilweise auch welche Farbschemen genutzt werden sollen.

Ein Beispiel für Gestaltungsrichtlinien von Desktop-Applikationen sind die „OS X Human Interface Guidelines“ (Apple 2013b) für das Betriebssystem Mac OS X von Apple oder der „UX Guide“ (Microsoft 2013) für die Betriebssysteme Windows 7 und Windows Vista von Microsoft. Im Bereich mobiler Applikationen existieren beispielsweise die „Android User Interface Design Guidelines“ (Open-Handset-Alliance 2013) für das mobile Betriebssystem Android der Open-Handset-Alliance oder die „iOS Human Interface Guidelines“ (Apple 2013a) für das mobile Betriebssystem iOS von Apple. In den Gestaltungsrichtlinien für iOS Applikationen legt Apple beispielsweise u.a. fest, wie das Layout, die Navigationsstruktur und die verwendete Terminologie der Applikation sein soll. Zudem wird beschrieben, welche Farben zur Visualisierung bestimmter Ereignisse geeignet sind und welche Fingergesten zur Bedienung bestimmter Aktionen verwendet werden sollen. Apple gilt allgemein als recht restriktiv bei der Einhaltung dieser Gestaltungsempfehlungen. So kann ein Verstoß eines Softwareentwicklers gegen diese Gestaltungsempfehlungen die Bereitstellung der Applikation über den Apple App Store verhindern.

2.3.3 Entwurfsmuster

Das Konzept der *Entwurfsmuster* (engl. design patterns) wurde erstmals vom Architekten Christopher Alexander im Kontext der Städteplanung vorgestellt. Sein Werk „The Timeless Way of Building“ (Alexander 1979) stellt eine Menge von Entwurfsmustern vor, um unterschiedliche Aspekte in Städten und Gebäuden zu gestalten. Die Idee von Entwurfsmustern wurde von Beck und Cunningham (1987) erstmals auf den Bereich der Softwarearchitektur übertragen. Einen hohen Bekanntheitsgrad haben Entwurfsmuster schließlich durch das Buch „Design Patterns: Elements of Reusable Object-Oriented Software“ von Gamma et al. (1995) erlangt. Das erste Interesse an Entwurfsmustern im Bereich Human Computer Interaction (HCI), welcher sich mit der Gestaltung von Benutzungsschnittstellen von Computer Systemen beschäftigt, geht auf das Jahr 1994 zurück (siehe z.B. (Rijken 1994)).

Entwurfsmuster stellen im Allgemeinen eine strukturierte Beschreibung einer wiederverwendbaren Lösung für wiederkehrende Problemstellungen in einem bestimmten Kontext dar

(Dearden/Finlay 2006, 55; Borchers 2001, 7). Entwurfsmuster im HCI-Bereich adressieren Problemstellungen bei der Gestaltung von Benutzungsschnittstellen, mit dem Ziel die Benutzungsfreundlichkeit zu verbessern (van Welie 2001, 104 f.). Während die konkrete Umsetzung von Gestaltungsprinzipien aufgrund ihres hohen Abstraktionsniveaus oftmals schwierig ist, haben Entwurfsmuster einen expliziten Bezug zu einer konkreten Problemstellung in einem konkreten Kontext (Dearden/Finlay 2006, 52). Zudem handelt es sich bei Entwurfsmustern nicht um neue, sondern um bereits existierende, bewährte Lösungen (Tidwell 2011, xviii), welche empirisch identifiziert werden (van Welie 2001, 94). Eine wesentliche Charakteristik von Entwurfsmustern ist ihre strukturierte Beschreibung. Hierzu wird eine Schablone mit mehreren Rubriken verwendet, deren Inhalte bei der Beschreibung eines einzelnen Entwurfsmuster gefüllt werden. Die einzelnen Rubriken unterscheiden sich zwar zwischen unterschiedlichen Autoren, sind jedoch generell recht ähnlich. Beispielsweise verwendet Tidwell (2011) die folgenden Rubriken:

- **Name:** Der Name des Entwurfsmusters dient dazu, dass Entwurfsmuster zu finden. Er sollte daher sorgfältig gewählt werden und die zentrale Idee des Entwurfsmusters transportieren (Borchers 2001, 65 f.; van Welie 2001, 102).
- **Was:** Diese Rubrik beschreibt, für welches Problem ein Entwurfsmuster einen Lösungsvorschlag anbietet. Bei der Beschreibung der Lösung sollten auch Einschränkungen des Lösungsvorschlages erwähnt werden (Borchers 2001, 68 f.; van Welie 2001, 102 f.).
- **Nutzung wenn:** In dieser Rubrik wird der Kontext des Entwurfsmusters beschrieben. Der Kontext bezieht sich in diesem Zusammenhang auf eine Menge von Bedingungen, unter welchen die fokussierte Problemstellung auftritt. Zudem sollten an dieser Stelle Kriterien für den zweckmäßigen Einsatz des Entwurfsmusters genannt werden. Nach dem Lesen dieser Rubrik sollte einem UI (User Interface)-Designer klar sein, ob diese Entwurfsmuster für seine vorliegende Problemstellung geeignet ist (van Welie 2001, 103).
- **Warum:** In dieser Rubrik wird die Nutzung des Entwurfsmusters motiviert. Es beschreibt die bei der Nutzung des Entwurfsmusters erzielten Vorteile und begründet warum diese erreicht werden (van Welie 2001, 104 f.).
- **Wie:** In dieser Rubrik wird der eigentliche Lösungsvorschlag des Entwurfsmusters beschrieben. Die Beschreibung sollte verständlich und generisch genug sein, so dass sie auf unterschiedliche Situationen und Plattformen innerhalb des fokussierten Kontext anwendbar ist (Borchers 2001, 70 f.).
- **Beispiele:** Diese Rubrik enthält Beispiele zur Veranschaulichung des Entwurfsmusters. Ziel ist es, dass Verständnis des UI-Designers für das Entwurfsmuster zu verbessern. Für Entwurfsmuster zur Gestaltung von Benutzungsschnittstellen bieten sich Bildschirmaufnahmen von existierenden Applikationen an, da diese den Inhalt des beschriebenen Lösungsvorschlages in einer kompakten Form transportieren können (van Welie 2001, 104).

Mittlerweile existieren eine Reihe von Sammlungen von Entwurfsmustern zur Gestaltung von Benutzungsschnittstellen von Computer Systemen. Teilweise wird für diese Sammlungen auch der Begriff *Entwurfsmuster-Sprache* (engl. pattern language) verwendet (Borchers 2001, 26 ff.). Eine Entwurfsmuster-Sprache setzt allerdings voraus, dass die einzelnen Entwurfs-

muster miteinander verknüpft sind (Borchers 2001, 52 ff.; van Welie/van der Veer 2003, 528). Der größte Teil dieser Sammlungen ist auf die Gestaltung von Benutzungsschnittstellen von Webapplikationen oder Desktop-Applikationen fokussiert. Beispiele sind die Sammlungen von van Welie (2013) und van Duyne et al. (2006) oder die Design Pattern Library von Yahoo (2011).

Auch für mobile Applikationen existieren bereits Sammlungen von Entwurfsmustern, beispielsweise von Ballard (2007, 95 ff.) oder Neil (2012). Auch die Sammlung von Tidwell (2011, 448 ff.) enthält u.a. Entwurfsmuster für mobile Geräte. Dabei fällt jedoch auf, dass viele Entwurfsmuster der genannten Sammlungen in ihren Beispielen lediglich Bezug auf bestimmte mobile Geräte und mobile Betriebssysteme nehmen. Aktuell dominieren in den angeführten Sammlungen Beispiele für das mobile Gerät iPhone von Apple mit dem zugehörigen mobilen Betriebssystem iOS. Aufgrund der unterschiedlichen Formfaktoren mobiler Geräte und unterschiedlichen Interaktionstechniken ist die Übertragbarkeit der Entwurfsmuster eingeschränkt. Neben der Unterscheidung bzgl. unterschiedlicher Geräteklassen, existieren auch Sammlungen von Entwurfsmustern, welche sich neben einer bestimmten Geräteklasse auch auf eine bestimmte Domäne fokussieren. Ein Beispiel ist die Sammlung von Borchers (2001, 75 ff.) zur Gestaltung von interaktiven Musikanlagen.

2.3.4 Zusammenfassung und Fazit

Wiederverwendbare Gestaltungsempfehlungen können bei der Umsetzung oder Bewertung von Benutzungsschnittstellen für Smartphone-Applikationen eingesetzt werden. Während Gestaltungsprinzipen vergleichsweise abstrakt formuliert sind, sind Gestaltungsrichtlinien sehr konkret und spezifisch. Letztere sind jedoch in ihrer Anwendbarkeit i.d.R. auf ein bestimmtes mobiles System beschränkt. Entwurfsmuster stellen konkrete Lösungsvorschläge für eine bestimmte Problemstellung bereit. Aufgrund ihrer strukturierten Beschreibung werden zusätzlich das Verständnis sowie ihre Vergleichbarkeit gefördert. Aktuell existieren bereits mehrere Sammlungen von Entwurfsmustern für wiederkehrende Problemstellungen bei der Gestaltung von Benutzungsschnittstellen für Smartphone-Applikationen. Die dabei adressierten Problemstellungen haben jedoch i.d.R. keinen Bezug zu einer bestimmten Applikationsdomäne. Es stellt sich die Frage, ob innerhalb der Applikationsdomäne „mobile ERP-Applikationen“ wiederkehrende Problemstellungen auftreten, für welche sich die Bereitstellung domänenspezifischer Entwurfsmuster eignet.

2.4 Endbenutzer-Entwicklung

In diesem Kapitel werden für die vorliegende Arbeit wichtige Aspekte aus dem Forschungsgebiet „Endbenutzer-Entwicklung“ behandelt. Hierzu wird die Endbenutzer-Entwicklung zunächst motiviert und wichtige Begriffe erläutert. Anschließend werden Herausforderungen und Lösungsansätze der Endbenutzer-Entwicklung behandelt. Schließlich werden verwandte Forschungsarbeiten und deren zentralen Ergebnisse vorgestellt. Ziel bei Letzterem ist es, erste Lösungsideen für die vorliegenden Fragestellungen zu erhalten.

2.4.1 Motivation und begriffliche Klärung

Da Computer und Softwareapplikationen in unserem Alltag mittlerweile eine wichtige Rolle spielen, sind auch die meisten Menschen mit ihrer generellen Bedienung vertraut. Die Entwicklung und Anpassung von Softwareapplikationen erfordert jedoch nach wie vor eine professionelle Ausbildung in der Softwareentwicklung (Lieberman et al. 2006a, 1), in welcher u.a. Methoden der Softwarekonzeption und der Einsatz von Programmiersprachen erlernt werden.

Unter *Endbenutzern* werden die Nutzer von Softwareapplikationen verstanden (Poswig 1996, 2). Als Folge des arbeitsteiligen Berufslebens besitzen Mitarbeiter in der Regel einen hohen Spezialisierungsgrad, d.h. sie sind Fachexperten in einem bestimmten Arbeitsgebiet. Endbenutzer haben i.d.R. keine professionelle Ausbildung in der Softwareentwicklung und haben oftmals auch keine Motivation und/oder keine Zeit sich Kenntnisse in diesem Bereich anzueignen (Lieberman et al. 2006b, ix). Stattdessen sind sie Fachexperten in anderen Domänen, wie beispielsweise in der Medizin oder im Ingenieurwesen (Beringer 2004, 39 f.; Lieberman et al. 2006b, viii). In einem betriebswirtschaftlichen Umfeld sind Endbenutzer beispielsweise Fachexperten in den Bereichen Personalwesen, Finanzwesen oder Logistik. Demzufolge ist eine Softwareapplikation aus der Perspektive eines Endbenutzers im Wesentlichen ein Werkzeug zur Aufgabenerfüllung (Poswig 1996, 1).

Gute Softwareapplikationen sollten die Aufgabenerfüllung für Endbenutzer erleichtern und/oder verbessern. Jedoch empfinden Endbenutzer existierende Softwareapplikationen teilweise als nicht flexibel genug und als schwer erlernbar (Lieberman et al. 2006b, vii). Beispielsweise werden Aufgaben i.d.R. durch eine von der Softwareapplikation festgelegte Sequenz von Aktivitäten bewältigt (Lieberman et al. 2006b, vii). Diese muss von Endbenutzern häufig zunächst mit Hilfe der Dokumentation der Softwareapplikation herausgefunden werden. Insbesondere im Bereich von umfangreicheren betriebswirtschaftlichen Applikationssystemen, wie ERP-Systemen, sind zudem häufig Schulungen notwendig, um deren Bedienung zu erlernen (Oja/Lucas 2010, 2). Zudem haben Endbenutzer in der Regel Anforderungen, welche von den existierenden Softwareapplikationen nicht abgedeckt werden (Lieberman et al. 2006a, 2). Dies liegt zum einen daran, dass keine Softwareapplikation die Anforderungen sämtlicher Endbenutzer in allen möglichen Situationen vollständig erfüllen kann (MacLean et al. 1990, 175). Zum anderen können sich die Anforderungen auch im Zeitverlauf ändern (Lieberman et al. 2006b, viii). Hierdurch entsteht der Bedarf, die neuen oder geänderten Anforderungen in der existierenden Softwareapplikation umzusetzen. Außerdem können Endbenutzer Softwareapplikationen derzeit nur eingeschränkt an ihre Bedürfnisse anpassen. Anpassungen können gewöhnlich nur über spezielle, von den Softwareentwicklern vorgesehene Parameter vorgenommen werden (Lieberman et al. 2006a, 3).

Die Konzeption und Implementierung von Softwareapplikationen wird in klassischen Vorgehensweisen von professionellen Softwareentwicklern durchgeführt (Lieberman et al. 2006a, 2). Hierdurch entsteht ein Engpass bzgl. der Anzahl an verfügbaren professionellen Softwareentwicklern. Zudem ist in den besagten, klassischen Vorgehensweisen ein Abstimmungsaufwand zwischen Endbenutzern und Softwareentwicklern notwendig. Zum einen müssen die

Endbenutzer ihre Anforderungen spezifizieren und an die Softwareentwickler kommunizieren. Die Aufgabe der Softwareentwickler ist es hierbei, die Anforderungen hinsichtlich ihrer technischen Umsetzbarkeit zu prüfen. Zum anderen sollten die Endbenutzer die implementierte Applikation evaluieren, um zu prüfen, ob ihre Anforderungen richtig umgesetzt wurden. Aufgrund des unterschiedlichen Hintergrundwissens von Endbenutzern und Softwareentwicklern besteht hierbei jedoch das Risiko von Missverständnissen. Dieses Risiko wird in der wissenschaftlichen Literatur als „Kommunikationslücke“ bezeichnet (vgl. z.B. (Wulf/Jarke 2004, 42)). Beringer spricht in diesem Zusammenhang davon, dass die Herausforderung bei der Entwicklung erfolgreicher Softwareapplikationen darin besteht, die Spannungsverhältnisses zwischen den beteiligten Wissens-Domänen abzubauen (Beringer 2004, 39 f.).

Alternativ könnten Endbenutzer eine Ausbildung in der Konzeption und Implementierung von Softwareapplikationen erhalten. Jedoch ist die Einstiegshürde in die professionelle Softwareentwicklung recht hoch. Die Implementierung von Softwareapplikationen erfolgt i.d.R. mit konventionellen Programmiersprachen, wie beispielsweise Java, C, C++ oder C#. Deren Beherrschung ist mit einem hohen Lernaufwand und praktischer Übung verbunden. Endbenutzer sind jedoch in der Regel nicht bereit diesen Aufwand zu investieren (Lieberman et al. 2006b, ix). Demzufolge müssen Endbenutzer ihre Änderungsanforderungen oder Wünsche nach zusätzlichen Funktionalitäten an die Softwareentwickler kommunizieren und darauf warten, dass diese in der nächsten Version der Softwareapplikation berücksichtigt werden. Nach Norman (1986, 33 ff.) ist das Kernproblem von Endbenutzern bei der Programmierung von Softwareapplikationen die Diskrepanz zwischen der Repräsentation eines Problems in den Köpfen der Endbenutzer und der von Computern akzeptieren Repräsentationen. Er argumentiert, dass dieses Problem entweder durch eine Annäherung der Endbenutzer an die Computer oder durch eine Annäherung der Computer an die Endbenutzer gelöst werden kann. Während klassische Programmiersprachen die erstgenannte Alternative verfolgen, fokussiert sich die Endbenutzer-Entwicklung auf die zweite Alternative (Smith et al. 2001, 77 f.).

Das Forschungsgebiet *Endbenutzer-Entwicklung* (engl. End-User Development (EUD)) beschäftigt sich mit der Fragestellung, wie Softwareapplikationen von Endbenutzern selbst entwickelt, angepasst oder erweitert werden können (Lieberman et al. 2006a, 2). Es geht also um Tätigkeiten in der Endbenutzer per Definition keine Experten sind (Poswig 1996, 1; Beringer 2004, 39). Dabei stehen insbesondere Methoden, Techniken und Werkzeuge im Fokus, welche Endbenutzer zu dieser Aufgabe befähigen (Lieberman et al. 2006b, viii). Das Forschungsgebiet Endbenutzer-Entwicklung ist kein gänzlich neues wissenschaftliches Gebiet und existiert bereits seit den frühen 80er Jahren (Sutcliffe 2005, 1). Es vereint verschiedene Stränge aus anderen wissenschaftlichen Disziplinen wie beispielsweise der Human Computer Interaction (HCI), dem Software Engineering (SE), der Computer Supported Cooperative Work (CSCW) oder der Artificial Intelligence (AI) (Lieberman et al. 2006a, 3). Andere Bezeichnungen bzw. verwandte Forschungsgebiete sind: End-User Programming (Nardi 1993, 5), End-User Computing (Rockart/Flannery 1983, 776 ff.) oder End-User Software Engineering (Burnett 2009, 16 ff.). Das generelle Ziel jeder Softwareimplementierung ist es, eine möglichst hohe Übereinstimmung zwischen den Anforderungen seiner Benutzer und den Funktionalitäten der Softwareapplikation zu erreichen (Sutcliffe 2005, 1). Der EUD-Ansatz

versucht dies zur erreichen, indem er die Gestaltungsaufgabe auf die Endbenutzer selbst überträgt.

EUD besitzt unterschiedliche Ausprägungsstufen. Mögliche EUD-Aktivitäten reichen von Anpassungen existierender Softwareapplikationen über Parameter, Stylesheets und Benutzerprofilen über angepasste Berichte bis hin zur vollständigen Entwicklung neuer Softwareapplikationen (Sutcliffe 2005, 2). In Anlehnung an Lieberman et al. (2006a, 3) können drei Typen von EUD-Aktivitäten unterschieden werden:

- **Parametrisierung:** bei diesem Typ kann der Endbenutzer durch die Wertzuweisung bereitgestellter Parameter das Verhalten der Softwareapplikation beeinflussen. Dies wird auch als *anpassungsfähige* (engl. adaptable) Softwareapplikation verstanden (Oppermann 1994, 2). Die alternativen Verhaltensweisen müssen in diesem Fall jedoch bereits vom Softwareentwickler implementiert sein. Ein Spezialfall der Parametrisierung sind sogenannte *adaptive* (engl. adaptive) Softwareapplikationen. Adaptive Softwareapplikationen analysieren das Verhalten des Endbenutzers und wählen auf Basis der gewonnen Erkenntnis automatisch eine entsprechende Verhaltensweise der Softwareapplikation aus (Oppermann 1994, 2).
- **Modifikation:** bei der Modifikation wird die existierende Funktionalität von Softwareapplikationen verändert oder neue Funktionalität ergänzt. Modifikationen sind aufwändiger als der Auswahlprozess in der Parametrisierung.
- **Erstellung:** bei der Erstellung handelt es sich um die Konzeption und Implementierung einer neuen Softwareapplikation.

Eine weitere Unterscheidung in EUD wird hinsichtlich des Zeitraumes der Endbenutzer Partizipation vorgenommen. In diesem Zuge wird zwischen einer Endbenutzer Beteiligung während der initialen Gestaltungsphase (engl. „design-before-use“) und einer Beteiligung während der Nutzung der Softwareapplikation (engl. „design-during-use“) unterschieden (Lieberman et al. 2006a, 4). Im ersten Fall ist das Ziel die Anforderungen der Endbenutzer besser zu verstehen und umzusetzen. Im Idealfall ist der Endbenutzer selbst fähig, seine Anforderungen umzusetzen. Der zweite Fall fokussiert Möglichkeiten, existierende Softwareapplikationen nachträglich anzupassen. Eine Bereitstellung aller denkbaren Alternativen und deren Auswahlmöglichkeit über eine Parametrisierung ist i.d.R. zu aufwändig und daher nicht zweckmäßig. Die Herausforderung in diesem Fall liegt der Konzeption einer Softwareapplikation, welche es dem Endbenutzer im Hinblick auf seine vorhandenen Kenntnisse mit vertretbarem Aufwand ermöglicht, seine gewünschten Modifikationen umzusetzen (Lieberman et al. 2006a, 4).

Geeignete Endbenutzer-Entwicklungswerkzeuge versuchen die Komplexität der Softwareentwicklung vor den Endbenutzern zu verbergen und stattdessen möglichst ihr „mentales Modell“ zu unterstützen (Rode et al. 2006, 161 ff.). Dies bedeutet u.a. dass die von Endbenutzern verwendete Terminologie verstanden und genutzt werden sollte und gleichzeitig auf Begrifflichkeiten aus dem Bereich der Softwareentwicklung verzichtet werden sollte. Das Werkzeug sollte auf Metaphern der fokussierten Domäne aufbauen, welche der Endbenutzer kennt (Beringer 2004, 40). Daher sollte der erste Schritt bei der Entwicklung eines Endbenut-

zer-Entwicklungswerkzeuges das Verständnis der Domäne sein, speziell der dort vorhandenen Aufgaben, Problemstellungen und zugehörigen Begrifflichkeiten (Spahn et al. 2008a, 484). Ein Endbenutzer-Entwicklungswerkzeug sollte nach Beringer (2004, 40) eine semantische Sprache bereitstellen, deren Bausteine die Entitäten und Aktionen der Ziel-Domäne abbilden. Zugehörige Entwicklungswerkzeuge werden von Fischer et al. (1992, 511 ff.) als *Domänen-orientierte Entwicklungsumgebungen* (engl. domain-oriented design environments, (DODE)) bezeichnet. Die Bausteine einer DOE werden von Beringer als „high-level semantic building blocks“ (Beringer 2004, 40) bezeichnet. Diese repräsentieren bekannte domänenspezifische Bausteine. Endbenutzer können durch die Kombination dieser Bausteine neue Applikationen erschaffen. Die Bereitstellung von Entwicklungswerkzeugen, welche Endbenutzer befähigen, ihre gewünschten Anforderungen an Softwareapplikationen selbst umzusetzen, führt zu einer verbesserte Übereinstimmung der angebotener Funktionalität von Softwareapplikationen und der benötigter Funktionalität von Endbenutzer. Als Folge sind Effizienz- und Effektivitäts-steigerungen des Unternehmens möglich (Wulf/Jarke 2004, 41 f.).

In einigen Anwendungsgebieten konnte sich die Endbenutzer-Entwicklung bereits erfolgreich etablieren. Ein Beispiel ist die Erstellung von Tabellenkalkulationen für eigene Berechnungen (Myers 1992, 15; Nardi 1993, vi f.; Lieberman et al. 2006a, 2). Tabellenkalkulationsapplikation ist ein Oberbegriff für Softwareapplikationen, mit welchen nummerische und alphanummerische Daten durch zweidimensionale Tabellen verarbeitet werden (Prevezanos 2012, 677). Die einzelnen Zellen der Tabelle enthalten Daten oder mathematische Formeln. Durch letztere können interaktive Berechnungen realisiert werden (Kolberg 2010, 258 ff.; Prevezanos 2012, 677 f.). Tabellenkalkulationsprogramme sind insbesondere in der Buchhaltung und Statistik weit verbreitet (Prevezanos 2012, 678). Bekannte Tabellenkalkulationsprogramme sind Microsoft Excel, Lotus 1-2-3 und LibreOffice Calc (Prevezanos 2012, 678). Das Beherrschen einer Programmiersprache ist zur Bedienung von Tabellenkalkulationsapplikationen nicht erforderlich. Hierdurch sind Endbenutzer in die Lage, eigene Berechnungsmodelle ohne professionelle Softwareausbildung umzusetzen. Für fortgeschrittene Anwendungsszenarien besitzen einige Tabellenkalkulationsapplikationen jedoch einfachere Programmiersprachen, um beispielsweise bestimmte Auswertungen zu automatisieren oder die Dateneingabe zu vereinfachen. Ein Vertreter einer solchen Programmiersprache ist „Visual Basic for Applications“ (VBA) aus dem Tabellenkalkulationsprogramm Microsoft Excel (Kolberg 2010, 747 ff.).

Ein weiteres, weit verbreitetes Beispiel für die Endbenutzer-Entwicklung ist die Definition von Filtern in E-Mail Applikationen (Lieberman et al. 2006a, 2). Derartige Filter können E-Mails nach festgelegten Kriterien analysieren und bestimmte Folgeaktionen ausführen. Ein häufiger Anwendungsfall ist das Löschen von unerwünschten E-Mails oder das Sortieren von E-Mails in eine festgelegte Ordnerstruktur.

Ein weiteres Anwendungsgebiet der Endbenutzer-Entwicklung ist die Erstellung von Webseiten. Aufgrund der weiten Verbreitung des Internets haben viele Internetnutzer das Bedürfnis eine eigene Webseite zu erstellen. Die Erstellung von Webseiten erfordert jedoch die Beherrschung verschiedener Technologien wie beispielsweise der Hypertext Markup Language (HTML) zur Festlegung der Struktur sowie der Inhalte der Webseite, Cascading Style Sheets

(CSS) zur Festlegung des Layouts der Webseite oder die Programmiersprache JavaScript zur Umsetzung interaktiver Elemente. Die Erlernung dieser Technologien stellt für Nutzer, ohne professionelle Kenntnisse in der Webentwicklung, ein Hindernis dar. Um es dieser Personengruppe dennoch zu ermöglichen eigene Webseiten zu erstellen, bieten Webhoster in vielen Fällen spezielle Entwicklungswerkzeuge für Endbenutzer an. Diese Werkzeuge arbeiten oftmals mit einer ausgewählten Menge häufig eingesetzter Bausteine von Webseiten, wie beispielsweise Inhaltsverzeichnisse, Kontaktformulare, Bildergalerien oder Gästebücher. Diese können vom Benutzer in der Regel über Drag & Drop in einen „What You See Is What You Get“ (WYSIWYG)-Editor-kombiniert werden. Ein WYSIWYG-Editor verfolgt das Ziel, Dokumente während der Bearbeitung möglichst genauso darzustellen, wie es bei der späteren Ausgabe aussehen werden (Shneiderman/Plaisant 2010, 194 ff.; Prevezanos 2012, 765). Neben der Kombination der Bausteine können diese i.d.R. auch über bereitgestellte Parameter an die eigenen Bedürfnisse angepasst werden. Neben kommerziellen Endbenutzer Applikationen in der Gestaltung von Webseiten gibt es auch Werkzeuge aus dem wissenschaftlichen Bereich. Ein Beispiel ist das Werkzeug „Click“ (Rode et al. 2006, 178 f.). Click verfolgt das Ziel der „direkten Manipulation“ (Shneiderman 1983, 57), bei welchem nicht zwischen Erstellungszeit und Laufzeit der Softwareapplikation unterschieden wird. Stattdessen gibt es nur eine Version der Webseite und sämtliche Änderungen werden direkt wirksam. Zusätzlich stellt Click verschiedene Komplexitätsebenen zur Gestaltung von Webseiten bereit. Beginner können Webseiten auf Basis einer Kombination vordefinierter Komponenten erstellen; erfahrenere Endbenutzer können sich auch den Quellcode der Komponenten anschauen und diesen direkt manipulieren. Damit wird das Designprinzip des „weichen Komplexitätsüberganges“ (engl. gentle slope of complexity) verfolgt (MacLean et al. 1990, 181; Lieberman et al. 2006a, 4). Dieses Designprinzip fordert, dass kleine Änderungen einfach durchzuführen sind. Bei umfangreichere Änderungen ist hingegen eine proportional zum Umfang der Änderung ansteigende Komplexität der Entwicklungsaktivität erlaubt (Lieberman et al. 2006a, 4).

2.4.2 Herausforderungen der Endbenutzer-Entwicklung

Ziel von EUD ist es, Endbenutzer zu befähigen Softwareapplikationen auf einer bestimmten Komplexitätsstufe zu entwickeln und anzupassen, welche ihren individuellen Fähigkeiten und Anforderungen entspricht (Lieberman et al. 2006a, 2). Die Herausforderung besteht jedoch darin, eine geeignete Komplexitätsstufe zu identifizieren. Dabei ist neben der eigentlichen Entwicklungsaktivität u.a. auch zu berücksichtigen, dass sich Endbenutzer in Ihren Fähigkeiten und in Ihrem Hintergrundwissen unterscheiden (Shneiderman/Plaisant 2010, 40 ff.). Zudem werden Endbenutzer durch die regelmäßige Nutzung von Softwareapplikationen geübter im Umgang und haben in den verschiedenen Entwicklungsstufen ihrer Nutzungserfahrung mit der Softwareapplikation unterschiedliche Anforderungen an die Interaktion und den Funktionsumfang (vgl. z.B. (Shneiderman/Plaisant 2010, 80 ff.)). Es kann kein universell einsetzbares EUD-Werkzeug für alle denkbaren Applikationsdomänen geben (Reppening/Ioannidou 2006, 82). Der entscheidende Erfolgsfaktor eines EUD-Werkzeuges ist laut Reppenning und Ionnidou (2006, 82) eine positive Lernerfahrung für die Endbenutzer. Dabei sollten die Herausforderung bei der Nutzung des Entwicklungswerkzeuges und die erforderlichen Kenntnisse in einem angemessenes Verhältnis zu den durchzuführenden Aufgaben stehen (Reppening/Ioannidou 2006, 82).

Eine wesentliche Voraussetzung für EUD sind flexible Softwarearchitekturen. Die Ansätze, die geforderte Flexibilität zu unterstützen, reichen von bereitgestellten Parametern, Regeln, metadatenbasierte Spezifikation des Systemverhaltens bis hin zu komponentenbasierten Softwarearchitekturen (Lieberman et al. 2006a, 5). Ein anderer Ansatz EUD zu ermöglichen, ist die Bereitstellung domänenspezifischer Sprachen und zugehöriger Werkzeuge mit Hilfe derer Endbenutzer ihre Anforderungen spezifizieren können (vgl. z.B. (Pühler 2011, 144 ff.)). Idealerweise können auf deren Basis auch die zugehörigen Softwareapplikationen automatisiert generiert werden.

Die wesentliche Motivation hinter der Endbenutzer-Entwicklung ist eine höhere Abdeckung der Anforderungen der Endbenutzer (Sutcliffe 2005, 1). Dieser Vorteil geht jedoch einher mit einem zusätzlichen Aufwand, den Endbenutzer in die Entwicklungsaktivitäten investieren müssen (Klann et al. 2006, 476). Der Aufwand der Endbenutzer-Entwicklung setzt sich aus mehreren Einzelaufwänden zusammen. Diese umfassen u.a. den Aufwand zur Erlernung des Werkzeuges, den Aufwand zur Spezifikation der Anforderungen an die zu entwickelnde Softwareapplikation, den eigentlichen Entwicklungsaufwand und den Aufwand für das Testen und die anschließende Fehlerkorrektur (Sutcliffe 2005, 1). Der zu investierende Aufwand steht in Konkurrenz zu den Kernaufgaben der Endbenutzer (Klann et al. 2006, 476). Die Motivation der Endbenutzer zur Nutzung eines EUD-Werkzeuges ist das entscheidende Erfolgskriterium eines solchen Werkzeuges (vgl. z.B. (Sutcliffe 2005, 1)). Diese ergibt sich aus der persönlichen Gegenüberstellung vom erwarteten Nutzen und den zu erwartenden Aufwänden eines Endbenutzers (Sutcliffe 2005, 1). Demzufolge besteht die generelle Herausforderung bei der Gestaltung eines EUD-Werkzeuges darin, die Aufwände der Endbenutzer soweit wie möglich zu reduzieren und Vorteile für die Endbenutzer zu genieren. Neben der Motivation des einzelnen Endbenutzers sind hierzu auch geeignete Rahmenbedingungen im Unternehmen notwendig, welche EUD-Aktivitäten ermöglichen und fördern.

2.4.3 Lösungsansätze zur Endbenutzer-Entwicklung

Im Folgenden werden ausgewählte Lösungsansätze für die Endbenutzer-Entwicklung vorgestellt. Dabei wird prinzipiell zwischen Interaktionstechniken und Entwicklungstechniken unterschieden. Unter *Interaktionstechniken* wird die Art und Weise verstanden, wie der Endbenutzer mit dem Entwicklungswerkzeug interagiert. Die Interaktionstechnik hat somit einen wesentlichen Einfluss auf die Gestaltung der Benutzungsschnittstelle des Entwicklungswerkzeuges. Unter *Entwicklungstechniken* wird hingegen das Vorgehen bei der Entwicklung verstanden. Dieses hat Einfluss auf Softwarearchitektur des Entwicklungswerkzeuges, d.h. auf die Anordnung seiner Systemkomponenten sowie die Beschreibung deren Beziehungen (Reussner/Hasselbring 2009, 1). Dabei ist anzumerken, dass sich die vorgestellten Lösungsansätze nicht gegenseitig ausschließen. Stattdessen werden bei der Gestaltung eines EUD-Werkzeuges oftmals mehrere der vorgestellten Lösungsansätze miteinander kombiniert.

2.4.3.1 Interaktionstechniken für EUD-Werkzeuge

In Anlehnung an Nardi (1993, 61 ff.) werden die folgenden Interaktionstechniken erläutert: visuelle Programmierung, Programming by Example und formularbasierte Programmierung.

Aufgrund der wenigen, praktisch relevanten Beispiele wird die von Nardi zusätzlich angeführte Interaktionstechnik der „automatischen Programm-Generierung durch informale Programmspezifikation“ (Nardi 1993, 78 ff.) nicht behandelt.

2.4.3.1.1 Visuelle Programmierung

Die visuelle Programmierung bezeichnet allgemein die Nutzung von visuellen Programmiersprachen zur Implementierung einer Softwareapplikation. Dabei wird die Programmlogik durch grafische Elemente und deren zwei- oder mehrdimensionaler Anordnung definiert (Myers 1986, 60; Poswig 1996, 21 ff.). Die grafischen Elemente repräsentieren hierbei interaktiv manipulierbare Softwarekomponenten (Schiffer 1998, 1). Durch den Einsatz visueller Elemente bei der Programmierung soll die Verständlichkeit von Programmen erhöht werden sowie die Programmerstellung vereinfacht werden (Schiffer 1998, 1).

Schiffer (1998) setzt sich in seiner Arbeit intensiv mit den Begrifflichkeiten im Umfeld visueller Programmierung auseinander. Dabei kommt er zum Ergebnis, dass trotzt zahlreicher Veröffentlichungen im Bereich visueller Programmierung kein einheitliches Begriffsverständnis existiert (Schiffer 1998, 21). Nach seinem Verständnis ist eine *visuelle Sprache* „eine formale Sprache mit visueller Syntax oder visueller Semantik und dynamischer oder statischer Zeichengebung“ (Schiffer 1998, 64). Demzufolge besitzen visuelle Elemente eine syntaktische und/oder semantische Bedeutung. Die *visuelle Syntax* wird durch die grafische Notation der Grundsymbole repräsentiert. Die grafische Notation wird durch die verwendeten Grafiken und Texte, ihrer räumlichen Anordnung sowie ihrer Verbindungen untereinander gebildet. Ein Beispiel für ein visuell syntaktisches Element ist eine Verbindungslinie zwischen zwei visuellen Elementen, welche die beiden Elemente zueinander in Beziehung setzt. Die *visuelle Semantik* wird durch die grafische Darstellung des Laufzeitzustandes der Grundsymbole abgebildet. Die Semantik einer visuellen Programmiersprache basiert auf der Interpretation der syntaktischen Elemente sowie deren Anreicherung durch visuelle Attribute. Ein Beispiel hierfür ist eine rote Hervorhebung eines visuellen Programmelementes, welches gegen die Syntax der visuellen Programmiersprache verstößt. Unter dem Begriff *dynamische Zeichengebung* wird die grafische Darstellung von flüchtigen Vorgängen verstanden, beispielsweise das Aufblinken eines Objektes, welches sich verändert (Schiffer 1998, 15).

Visuelle Programmiersprachen werden oftmals durch zugehörige Entwicklungsumgebungen unterstützt. Diese Entwicklungsumgebungen unterstützten neben der visuellen Programmierung auch die anschließende Generierung der Programme auf Basis der visuell spezifizierten Programmlogik. Schiffer (1998, 46) unterscheidet hierbei zwischen visuellen Entwicklungsumgebungen mit visueller Programmiersprache und visuellen Entwicklungsumgebungen mit textueller Programmiersprache. Während in der ersten Gruppe die Programmlogik vorwiegend durch visuelle Elemente spezifiziert wird, beschränkt sich die visuelle Programmierung in der zweiten Gruppe in der Regel auf die Gestaltung der Benutzungsschnittstelle. Beispiele für die erste Gruppe sind LabView[2] (Laboratory Virtual Instrument Engineering Workbench)

[2] http://www.ni.com/labview, zugegriffen am 24.4.2014

von National Instruments oder VEE[3] (Visual Engineering Environment) von Agilent Technologies. Bei beiden Entwicklungsumgebungen handelt es um Entwicklungswerkzeuge zur Erstellung von Softwareapplikationen im Bereich Messtechnik. Ein Beispiel für eine visuelle Entwicklungsumgebung mit textueller Programmiersprache ist XCode[4] von Apple. Bei XCode handelt es sich um eine Entwicklungsumgebung zur Erstellung von Softwareapplikationen für die Apple Betriebssysteme iOS und Mac OS X. Dabei wird die Programmlogik mit Hilfe der Programmiersprache Objective-C spezifiziert. Die grafische Benutzungsschnittstelle der Applikation wird hingegen mit Hilfe eines WYSIWYG-Editors umgesetzt. Hierbei werden verfügbare Elemente der grafischen Benutzeroberfläche, wie beispielsweise Schaltflächen oder Textfelder per Drag & Drop, auf der Bildschirmmaske platziert und durch Eigenschaftswerte parametrisiert.

Neben visuellen Programmiersprachen existieren nach Schiffer (1998, 45 f.) sogenannte *visuelle Softwarebeschreibungssprachen*. Im Gegensatz zu visuellen Programmiersprachen können mit visuellen Softwarebeschreibungssprachen nicht alle Aspekte einer Softwareapplikation spezifiziert werden. Stattdessen beschränken sie sich auf bestimmte Aspekte einer Softwareapplikation, wie beispielsweise der Visualisierung der Vererbungsbeziehungen zwischen Klassen eines auf Basis einer objektorientierter Programmiersprache erstellten Softwareapplikation. Visuelle Softwarebeschreibungssprachen müssen dem Anspruch genügen, dass auf ihrer Basis Programmcode maschinell generierbar ist (Schiffer 1998, 45).

Die Befürworter visueller Programmierung führen u.a. die folgenden Vorteile an:

- Grafische Elemente sind mächtigere Kommunikationsmittel als Text, weil sie auf keine Sprachbarrieren stoßen. Dieses Argument bezieht sich insbesondere auf die oftmals auf englischer Sprache basierenden Schlüsselwörter in klassischen, textuellen Programmiersprachen (Suleiman/Wayne 1992, 141 ff.; Shu 1988, 7 f.).
- Visuelle Repräsentationen unterstützten das Verstehen und die Ideenübermittlung (Ichikawa et al. 1990, 3).
- Grafische Elemente besitzen eine höhere Informationsdichte als Text. Durch die bessere Informationsverarbeitung des Menschen bei Bildern im Vergleich zu textuellen Informationen können mehr Informationen gleichzeitig verarbeitet werden (Raeder 1985, 12 f.).
- Visuelle Programmiersprachen können zwei oder mehrere Dimensionen zur Strukturierung der Programmlogik nutzen. Darunter wird die Möglichkeit einer horizontalen- und vertikalen Anordnung der grafischen Elemente am Bildschirm verstanden. Dieses Argument bezieht sich speziell auf den Vorteil gegenüber dem eindimensionalen textuellen Quellcode in klassischen Programmiersprachen (Myers 1986, 60).

Hingegen führen Kritiker visueller Programmierung u.a. die folgenden Nachteile an:

[3] http://www.home.agilent.com/en/pd-1476554-pn-W4000D/vee-pro-932?nid=34095.806312.00&cc=DE&lc=ger&cmpid=20604, zugegriffen am 24.4.2014
[4] https://developer.apple.com/xcode, zugegriffen am 24.4.2014

- Es besteht ein Interpretationsspielraum bei der Nutzung grafischer Elemente. Die Bedeutung grafischer Elemente ist in vielen Fällen abhängig vom Kontext (Schiffer 1998, 60).
- Visuelle Programmiersprachen benötigen im Gegensatz zu textuellen Programmiersprachen einen hoher Platzbedarf (Nardi 1993, 64 f.; Leibs/Goldberg 1993, 45). Aus diesem Grund sind sie insbesondere für die Konstruktion umfangreicher Softwareapplikationen nicht geeignet (Nickerson 1994, 232 ff.).
- Visuelle Programme erfordern einen höheren Interpretationsaufwand als textuelle Programme. Insbesondere ist die Leserichtung oft unklar (Wang/Lee 1993, 330 f.; Schiffer 1998, 60 f.).
- Visuelle Programmiersprachen unterstützen Anwender nicht beim Verstehen von generellen Programmierkonzeption wie beispielsweise Bedingungen oder Iterationen (Myers 1992, 16).
- Es gibt kaum empirische Untersuchungen, welche die Vorteile visueller Programmiersprachen gegenüber textueller Programmiersprachen stützen (Nardi 1993, 62; Schiffer 1998, 253).

2.4.3.1.2 Programming By Example

Bei *Programming by Example*[5] (PbE) handelt es sich um eine Interaktionstechnik der Endbenutzer-Entwicklung, bei welcher ein Computersystem eine Programmlogik auf Basis von konkreten Beispieldatensätze erlernen soll (Myers 1986, 59). Dies unterscheidet PbE von klassischen Programmiersprachen, welche die Programmlogik i.d.R. abstrakt in Form einer von der Programmiersprache vorgegebenen Syntax beschreiben. PbE nutzt die Erkenntnis, dass Menschen besser mit konkreten Beispielen umgehen können, als mit abstrakten Ideen (Myers 1986, 60; Lieberman 2000, 73).

Die Grundidee bei PbE ist, dass ein Endbenutzer einzelne Schritte in seiner gewohnten Softwareapplikation durchführt, um eine Aufgabenstellung zu lösen. Die Softwareapplikation zeichnet diese Schritte auf und generiert daraus ein Programm, welches zu einem späteren Zeitpunkt ausgeführt werden kann (Schiffer 1998, 122). Der Vorteil bei PbE ist, dass der Endbenutzer im Gegensatz zu klassischen Programmiersprachen zur Erstellung von Programmen keine Syntax einer Programmiersprache erlernen muss (Lieberman 2000, 73 f.; Smith et al. 2001, 11 f). Zudem muss er den Umgang mit einer neuen Entwicklungsumgebung nicht einüben. Er führt die gewünschten Schritte zur Lösung einer Aufgabenstellung in seiner bereits bekannten Softwareapplikation aus, mit dem einzigen Unterschied, dass diese aufgezeichnet werden (Smith et al. 2001, 10).

Bei der Programm-Generierung werden in Anlehnung an Myers (1986, 60) zwei Ansätze unterschieden: „Mache, was ich gemacht habe“ (engl. „do what I did“) oder „Mache, was ich meine“ (engl. „do what I mean“). Im ersten Fall besteht die PbE-Aufgabe im Wesentlichen darin, die durchgeführten Schritte des Benutzers in Programmcode zu übersetzen. Bei diesem Ansatz ist jedoch die Einsetzbarkeit des generierten Programmcodes begrenzt, da bereits

[5] Teilweise auch als „Programming by Demonstration“ bezeichnet (vgl. z.B. (Lieberman et al. 2006a, 3))

kleine Abweichungen vom ursprünglichen Szenario zu Programmabbrüchen oder falschen Ergebnissen führen können (Schiffer 1998, 123). Diese simplen Aufzeichnungen und Übersetzungen werden häufig als *Makro* (Prevezanos 2012, 411) bezeichnet; das zugehörige Aufzeichnungswerkzeug als *Makrorecorder* (Schiffer 1998, 122 ff.). Im zweiten Fall versucht das System Schlussfolgerungen aus den gezeigten Schritten zu ziehen, um so Rückschlüsse auf die Absicht des Endbenutzers zu erhalten. Die zentrale Herausforderung liegt hierbei darin, die aufgezeichneten Schritte so zu Generalisieren, dass diese auch in ähnlichen Situationen anwendbar sind (Poswig 1996, 31; Schiffer 1998, 122; Lieberman 2001, 2). Während die generierten Programme der zweiten Gruppe flexibler einsetzbar sind, ist ihr Generierungsvorgang jedoch deutlich komplexer. Zu lösende Probleme sind hierbei u.a. Mehrdeutigkeiten in den demonstrierten Schritten des Endbenutzers sowie die Erkennung von Schleifen und Verzweigungen (Schiffer 1998, 122). Ein Lösungsansatz zur Beherrschung des Generalisierungsproblems ist die Aufbereitung und Präsentation der aufgezeichneten Schritte für den Endbenutzer, so dass dieser seine gewünschte Generalisierung selbst vornehmen kann. Hierbei ist jedoch zu klären, in welcher Art und Weise die aufgezeichneten Schritte präsentiert werden, als auch wie diese geeignet vom Endbenutzer modifiziert werden können (Smith et al. 2001, 10). Das zu lösenden Problem liegt hierbei in der Diskrepanz zwischen der Programmaufzeichnung auf Interaktionsebene und der Programmrepräsentation auf Systemebene (Schiffer 1998, 123). Es besteht Einigkeit darüber, dass das Erlernen einer komplizierten Syntax als Voraussetzung zum Verstehen und Modifizieren der Programmaufzeichnungen nicht geeignet ist (Smith et al. 2001, 10).

Ein bekanntes Beispiel für PbE sind Makros in den Bürosoftwareapplikationen der Firma Microsoft. Hierbei werden die vom Endbenutzer durchgeführten Schritte aufgezeichnet und in die Programmiersprache VBA übersetzt (Kolberg 2010, 744 ff.). Die Syntax von VBA ist an die Syntax der Microsoft Programmiersprache Visual Basic angelehnt, hat jedoch einen eingeschränkteren Sprachumfang (Kolberg 2010, 747). Ein Beispiel für PbE aus dem Bereich SAP-ERP ist eCATT (extended Computer Aided Test Tool). Mit eCATT kann die Durchführung von SAP-Transaktionen aufgezeichnet werden. Dieses wird in einen Programmcode in der SAP-Programmiersprache ABAP-Skript überführt. Ähnlich wie bei VBA-Makros kann der generierte Programmcode eingesehen und angepasst werden. Neben dem Testskript können auch Testdaten zur Parametrisierung des Skripts bereitgestellt werden. Die Hauptanwendungsfälle von eCATT sind das Testen von neuen SAP-Transaktionen und die automatisierte Eingabe großer Datenmengen (Naumann 2009, 17 ff.).

2.4.3.1.3 Formularbasierte Programmierung

Bei der formularbasierten Programmierung werden Programme durch das Ausfüllen von Formularen erstellt. In Anlehnung an Jeffries und Rosenberg (1987, 261) ist ein Formular eine Schablone mit einer Menge von Feldern zur Repräsentation von Variablen. Der Endbenutzer hat die Aufgabe in diese Felder Werte einzutragen (Nardi 1993, 67; Frank/Szekely 1998, 153). Dabei hat der Endbenutzer teilweise auch die Möglichkeit selbst Felder vom sichtbaren Formular zu entfernen oder neue hinzuzufügen (Jeffries/Rosenberg 1987, 261). In umfangreicheren formularbasierten Programmierumgebungen kann der Endbenutzer neben der Werteingabe auch Formeln zur Berechnung von Werten hinterlegen. Dabei erfolgt die

Programmierung deklarativ, d.h. der der Endbenutzer definiert keine Algorithmen, sondern gibt in Formeln an, welche Daten zur Berechnung anderer Daten benötigt werden und mit welchen Methoden die Daten zu verknüpfen sind. (Schiffer 1998, 125).

Der Vorteil formularbasierter Programmierung ist, dass sie generell leichter zu erlernen sind. Dies wird damit begründet, dass Endbenutzer bereits mit papierbasierten Formularen vertraut sind. Zudem muss im Gegensatz zu klassischen Programmiersprachen keine Syntax erlernt werden (Nardi 1993, 66 f.). Zudem wird die Wahrscheinlichkeit von Programmierfehlern durch die Bereitstellung von Optionen und Menüpunkten reduziert (Jeffries/Rosenberg 1987, 261; Nardi 1993, 66). Der Endbenutzer muss vordefinierte Felder in Formularen ausfüllen anstatt syntaktisch korrekte Programmieranweisungen in einen Quellcode-Editor zu schreiben (Jeffries/Rosenberg 1987, 261; Nardi 1993, 66). Da bei formularbasierten Ansätzen reduziert, beschleunigt sich der Programmiervorgang (Jeffries/Rosenberg 1987, 261).

Nachteilig ist beim formularbasierten Programmieransatz, dass sie i.d.R. nur für Problemklassen geeignet sind, welche eine kleine Menge möglicher Variablen besitzen (Nardi 1993, 66 f.). Zudem sollte die zu unterstützende Aufgabe bereits als eine strukturierte Abfolge von Aktionen vorliegen und die Menge der notwendigen Variablen und deren mögliche Wertausprägung bekannt sein (Nardi 1993, 67). Die formularbasierte Programmierung ist zudem nicht geeignet um komplexere, prozedurale Abläufe umzusetzen (Nardi 1993, 67).

Generell hat sich die formularbasierte Programmierung bewährt, wenn die Programmerstellung auf eine bestimmte Problemklasse begrenzt ist (Nardi 1993, 67). Ein Beispiel sind die in vielen Softwareapplikationen vorhandenen *Assistenten* (engl. wizards) zur Lösung bestimmter Aufgabenstellungen. Beispielsweise besitzt die Entwicklungsumgebung SAP Mobile Workspace (siehe Kapitel 6.1.2) verschiedene formularbasierte Assistenten zur Erstellung mobiler SAP Applikationen. Beispiele für unterstützte Aktivitäten durch die genannten Assistenten sind das Anlegen eines neuen Entwicklungsprojektes oder das Installieren einer mobilen Applikationen in die Laufzeitumgebung der SAP Mobile Platform (vgl. z.B. (Homann et al. 2013b, 188 ff.)). Das Tabellenkalkulationsprogramm Excel der Firma Microsoft ist ein Beispiel für eine umfangreichere formularbasierte Programmierumgebung. Neben der Werteingabe wird in Excel auch die Erstellung eigener Berechnungsformeln unterstützt (vgl. z.B. (Kolberg 2010, 229 ff.)).

2.4.3.1.4 Fazit und Diskussion

Insgesamt zeigt die Analyse der vorgestellten Interaktionstechniken, dass jede Interaktionstechnik ihre spezifischen Vor- und Nachteile besitzt. Die höchste Flexibilität bietet die visuelle Programmierung. Damit können neue Softwareapplikationen unabhängig von einer bereits existierenden Softwareapplikation entwickelt werden. Häufig werden visuelle Programmiersprachen nach dem Datenflussprinzip konzipiert. Bei der Konzeption einer visuellen Programmiersprache für mobile ERP-Applikation müssten die verfügbaren visuellen Elemente sowie deren Verknüpfungsregeln definiert werden. Bei der Umsetzung einer zugehörigen Entwicklungsumgebung müsste ein grafischer WYSIWYG-Editor zur visuellen Programmgestaltung bereitgestellt werden. Zusätzlich wäre ein Transformationswerkzeug zur Überführung

des visuellen Programmes in den ausführbaren Programmcode der mobilen ERP-Applikation notwendig. Zur Bewertung der Zweckmäßigkeit einer visuellen Programmiersprache im Kontext von mobilen ERP-Applikationen wäre vorab abzuwägen, ob wiederverwendbare Bausteine zur Definition der visuellen Elemente identifiziert werden können, wie diese parametrisiert werden können und ob der Datenfluss mobiler ERP-Applikationen die Flexibilität einer visuellen Programmiersprache erfordert.

Programming by Example eignet sich hingegen hauptsächlich für die Automatisierung häufig durchgeführter Aktivitäten in einer existierenden Softwareapplikation. Da hiermit keine neuen Softwareapplikationen erstellt werden können, sondern nur bestehende Funktionalitäten in einer festgelegten Reihenfolge aufgerufen werden, erscheint PbE ungeeignet für die Entwicklung neuer mobiler ERP-Applikationen zu sein.

Die formularbasierte Programmierung eignet sich zur Erstellung neuer Softwareapplikationen, bei denen im Vorfeld möglichst viel über deren Struktur bekannt ist. Die Programmierung beschränkt sich in diesem Fall auf die Auswahl bestimmter Optionen und die Wertzuweisung von bereitgestellten Parametern. Um die Zweckmäßigkeit der formularbasierten Programmierung im Kontext von mobilen ERP-Applikationen zu bewerten, müsste vorab untersucht werden, ob die möglichen Ausprägungen mobiler ERP-Applikationen in einer handhabbaren Auswahl von Optionen und Parametern abbildbar ist.

2.4.3.2 Entwicklungstechniken für EUD-Werkzeuge

Im Folgenden werden ausgewählte Entwicklungstechniken für EUD-Werkzeuge vorgestellt. Dabei werden folgende Entwicklungstechniken behandelt: die modellgetriebene Entwicklung, die komponentenorientierte Entwicklung sowie domänenspezifische Sprachen.

2.4.3.2.1 Modellgetriebene Entwicklung

Die Grundidee der modellgetriebene Softwareentwicklung (engl. model-driven software development (MDSD)) ist, dass die Spezifikation einer Softwareapplikation unabhängig von ihrer technischen Umsetzung erfolgen sollte (Gruhn et al. 2006, 21). MDSD ist ein Oberbegriff für Techniken, die automatisiert aus formalen Modellen lauffähige Software erzeugen (Stahl/Völter 2006, 14 ff.). Gegenüber der klassischen Softwareentwicklung wird bei MDSD nicht direkt Quellcode entwickelt. Stattdessen wird versucht die unterschiedlichen Aspekte der zu entwickelnden Softwareapplikation durch formale Modelle abzubilden (Stahl/Völter 2006, 14). Diese Modelle stellen anschließend die Grundlage dar, um mit Hilfe eines Transformationswerkzeuges sowie zugehöriger Transformationsregeln Quellcode zu erzeugen (Gruhn et al. 2006, 21). Grundlegendes Konzept von MDSD ist die *Abstraktion*. Darunter wird in MDSD das Weglassen von irrelevanten Details verstanden (Petrasch/Meimberg 2006, 45). Modelle können in MDSD auf unterschiedlichen Abstraktionsebenen existieren. Ziel ist eine vorzugsweise vollautomatisierte Transformation von einem abstrakteren hin zu einem technologiespezifischeren Modell (Gruhn et al. 2006, 21). Zusätzlich ist der *Formalisierungsgrad* des Modells eine wichtige Voraussetzung (Stahl/Völter 2006, 14). Formalismus stellt ein Regelwerk für ein System dar, das mathematisch exakt, widerspruchsfrei und vollständig

definiert werden kann (Sander et al. 1995, 100 f.). Formale Modelle erfordern die Existenz einer eindeutigen Syntax und einer eindeutigen Semantik (Gruhn et al. 2006, 65). Eine konkrete Spezifikation eines MDSD-Ansatzes ist die Model-Driven Architecture (MDA) der Object Management Group (OMG). Diese umfasst u.a. die notwendige Infrastruktur, den Software-Prozess bis hin zu den verwendeten Hilfsmitteln und Werkzeugen (Gruhn et al. 2006, 21).

Die Ziele von MDSD sind u.a.: eine Beschleunigung des Entwicklungsprozesses, die Verbesserung der Softwarequalität, die verbesserte Portierbarkeit sowie eine verbesserte Verwaltbarkeit (Gruhn et al. 2006, 21 f.; Stahl/Völter 2006, 13 f.). Die Beschleunigung des Entwicklungsprozesses soll durch ein hohes Maß an Werkzeugunterstützung und der damit verbundenen Automatisierung der Entwicklungsaktivitäten erreicht werden. Zudem soll der Aufwand durch die verbesserte Wiederverwendung von existierenden Softwareartefakten verringert werden. Dies wird u.a. dadurch bewerkstelligt, dass technische Details in wiederverwendbare Transformationsregeln gekapselt werden (Gruhn et al. 2006, 22). Diese Wiederverwendung führt auch dazu, dass Fehler in der Implementierung technischer Details vermieden werden und hierdurch die Qualität der Entwicklung bzw. der entwickelten Softwareartefakte erhöht wird (Stahl/Völter 2006, 13). Durch die systematische Abstraktion von technischen Details wird die Portierung der entwickelten Softwareapplikationen aus anderen Programmiersprachen oder Laufzeitumgebungen erleichtert (Gruhn et al. 2006, 22). Durch die Entwicklung auf einer höheren Abstraktionsstufe erfolgt ein Übergang von einer technikzentrierten Entwicklung zu einer stärker fachlich-orientierten Entwicklung. Hierdurch können wiederverwendbare, fachliche Lösungskomponenten gestaltet werden, welche besser verwaltet werden können (Gruhn et al. 2006, 21 f.; Stahl/Völter 2006, 13).

Neben der allgemeinen Forschung zur modellgetriebenen Softwareentwicklung existiert ein spezieller (Teil-)Forschungsbereich, welcher sich mit der Entwicklung von Benutzungsschnittstellen beschäftigt (engl. model-based user interface development (MBUID)). Die Motivation hinter MBUID ist, dass ein Großteil des Quellcodes von Softwareapplikationen sowie ein Großteil der Entwicklungszeit für die Gestaltung der Benutzungsschnittstelle verwendet wird (vgl. z.B. (Myers/Rosson 1992, 195)). Zudem wird die Entwicklung von Softwareapplikationen für unterschiedliche Geräte und Laufzeitumgebungen u.a. aufgrund der unterschiedlichen Displaygrößen, Grafikbibliotheken und Programmiersprachen erschwert (Meixner et al. 2011, 2). Analog zur modellgetriebenen Entwicklung im Allgemeinen versucht MBUID von diesen technischen Details zu abstrahieren und die Entwicklung auf einer abstrakteren Modellebene zu ermöglichen (Paterò 2000, 99 ff.; Meixner et al. 2011, 3 f.).

Nachteilig bei modellgetriebenen Entwicklungsansätzen ist vor allem der hohe Initialaufwand bei der Spezifikation der entsprechenden Modellebenen sowie die Entwicklung und Bereitstellung benötigter Werkzeuge. Hierbei stehen im Allgemeinen mehrere, sich gegenseitig beeinflussende Entwurfszielen in einem konfliktären Verhältnis zueinander (Behrens et al. 2006, 124 ff.). Zudem stellt die Sicherstellung der Konsistenz zwischen den unterschiedlichen Modellebenen eine Herausforderung dar. Dies ist insbesondere schwierig, wenn Änderungen auf unterschiedlichen Hierarchieebenen unterstützt werden sollen (Stahl/Völter 2006, 24).

2.4.3.2.2 Komponentenorientierte Entwicklung

Eine weitere potentielle Entwicklungstechnik der Endbenutzer-Entwicklung ist die komponentenorientierte Entwicklung (engl. component-based development, CBD) (Won et al. 2006, 115 ff.). Die zentrale Idee hinter der komponentenorientierten Entwicklung ist die Bereitstellung wiederverwendbarer Softwarekomponenten, welche für eigene Entwicklungsaktivitäten verwendet und angepasst werden können (Won et al. 2006, 117). In Anlehnung an Szyperski ist eine *Softwarekomponente* „a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" (Szyperski 2002, 41). Demzufolge kapseln Softwarekomponenten ihre Implementierung in einer modularen Einheit und stellen diese über definierte Schnittstellen zur Verfügung (Behrens et al. 2006, 48).

Durch die Nutzung wiederverwendbarer Softwarekomponenten kann die Entwicklungsgeschwindigkeit erhöht werden (Andresen 2003, 2). Durch die größere Nutzerzahl der Komponenten ist zudem häufig eine höhere Qualität gewährleistet (Andresen 2003, 2). Das Ziel von Softwarekomponenten ist die Kapselung von Geschäftslogik in eigenständige, wiederverwendbare Einheiten (Andresen 2003, 2). Aufgrund ihrer fachlichen Nähe sollten Softwarekomponenten demzufolge auch von Endbenutzern verwendet werden können (Won et al. 2006, 115 ff.).

Eine ähnliche Idee verfolgt das Konzept der *generativen Programmierung*. Diese orientiert sich an der industriellen Fertigung (Klar/Klar 2006, 2). Dabei sollen Softwareapplikationen einer bestimmten Domäne vollautomatisch aus der Anforderungsspezifikation und der durchgeführten Konfiguration von vorgefertigten Softwarekomponenten entwickelt werden können. (Czarnecki/Eisenecker 2000, 5; Gruhn et al. 2006, 45).

Ähnlich wie bei der modellgetriebenen Entwicklung ist auch bei der komponentenorientierten Entwicklung ein größerer initialer Aufwand notwendig, um geeignete Softwarekomponenten zu entwickeln. Hierbei muss ein geeigneter Modularisierungsgrad der Softwarekomponenten gefunden werden sowie die Abhängigkeiten zu anderen Softwarekomponenten berücksichtigt werden (Behrens et al. 2006, 89).

2.4.3.2.3 Domänenspezifische Sprachen

Domänenspezifische Sprachen (engl. domain-specific language (DSL)) sind eng mit der modellgetriebenen Entwicklung verbunden. Als *Domäne* wird ein bestimmter fachlicher Geltungsbereich bezeichnet (Stahl/Völter 2005, 4; Gruhn et al. 2006, 70; Petrasch/Meimberg 2006, 41) und besteht aus domänenspezifischen Abstraktionen, Konzepten und Regeln (Merrnik et al. 2005, 317; Stahl/Völter 2005, 250). Die dahinterliegende Idee ist es, Modelle auf einer fachlichen Ebene zu erstellen und technische Implementierungsdetails automatisiert zu ergänzen (Stahl/Völter 2005, 4). Die zugehörigen fachlichen Modelle werden i.d.R. mit Hilfe einer domänenspezifische Sprachen spezifiziert (Stahl/Völter 2005, 4). Durch die Fokussierung auf einen spezielle Domäne soll die bessere Ausdrucksmöglichkeit und ein besseres Verständnis erreicht werden (Gruhn et al. 2006, 70). Eine DSL wird verwendet, um

fachliche Modelle formal beschreiben zu können (Stahl/Völter 2005, 16). Domänenspezifische Sprachen besitzen ein Metamodell inkl. einer statischen Semantik, eine korrespondierende konkrete Syntax sowie eine dynamische Semantik (Stahl/Völter 2005, 68). Letztere weißt den Konstrukten der DSL eine Bedeutung zu (Stahl/Völter 2005, 68).

Bei der Gestaltung einer DSL ist es wichtig, dass diese Konstrukte der DSL die Sprachwelt der Domänenexperten verwendet. Dies erleichtert das Verständnis und die Nutzung der DSL für die Domänenexperten (Stahl/Völter 2005, 68). Zusätzlich sollten zugehörige Werkzeuge (Transformatoren bzw. Generatoren) existieren, um aus den DSL-Konstrukten Codebausteine der Zielsprache auf der technischen Ebene erzeugen zu können (Stahl/Völter 2005, 68). Die Aktivitäten zur Entwicklung einer DSL werden als Domain Engineering (DE) bezeichnet. DE beschäftigt sich damit, „die Wiederverwendung von Entwicklungswissen einer spezifischen Domäne auf eine systematische Basis zu stellen“ (Gruhn et al. 2006, 43). DE unterteilt sich die folgenden Aktivitäten: Domänen-Analyse, Domänen-Entwurf und Domänen-Implementierung (Gruhn et al. 2006, 43).

2.4.3.2.4 Fazit und Diskussion

Bei Betrachtung der vorgestellten Entwicklungstechniken zeigt sich, dass alle drei Techniken eine Entwicklung auf fachlicher Ebene verfolgen und von technischen Implementierungsdetails abstrahieren. Die vorgestellten Techniken schließen sich nicht gegenseitig aus. Vielmehr stellen domänenspezifischen Sprachen eine geeignete Basis zur Spezifikation von fachlichen Modellen im Rahmen der modellgetriebenen Entwicklung dar. Zudem kann die Transformation von einer abstrakteren Modellebene auf eine darunterliegende Modellebene unter Nutzung von bereitgestellten Softwarekomponenten erfolgen.

Aufgrund der hohen Heterogenität im Umfeld mobiler Endgeräte und zugehöriger mobiler Betriebssysteme erscheint es sinnvoll, mobile ERP-Applikationen auf einer abstrakteren Modellebene zu spezifizieren. Dies hat mehrere Vorteile. Zum einen könnten die spezifizierten Applikationen in unterschiedliche Implementierungsformate transformiert werden. Zudem müsste bei einem Versionswechsel des mobilen Betriebssystems nicht jede entwickelte Applikation angepasst werden, sondern lediglich das entsprechende Transformationswerkzeug. Um die Konstrukte der benötigten Modellebene formal zu definieren, erscheint eine DSL geeignet. Mit Hilfe der fachlichen Ausrichtung einer solchen DSL sollen Endbenutzer die Konstrukte der DSL nutzen können, um selbst mobile ERP-Applikationen zu entwickeln.

2.4.4 Verwandte Forschungsarbeiten

Im Folgenden werden verwandte Forschungsarbeiten aus dem Bereich Endbenutzer-Entwicklung mobiler ERP-Applikationen untersucht. Da keine Forschungsarbeiten gefunden werden konnten, welche sich direkt mit diesem Schwerpunkt „mobile ERP-Applikationen“ auseinandersetzen, werden stattdessen Arbeiten mit einem Fokus auf die Endbenutzer-Entwicklung von traditionellen, Desktop-basierten ERP-Applikationen und Arbeiten mit einem Fokus auf die Endbenutzer-Entwicklung von mobilen Applikationen untersucht.

2.4.4.1 Endbenutzer-Entwicklung von ERP-Applikationen

Trotz der weiten Verbreitung von ERP-Systemen sind kaum Forschungsarbeiten auf dem Gebiet der Endbenutzer-Entwicklung von ERP-Applikationen zu finden. Eine Ausnahme bilden die Forschungsarbeiten von Spahn et al. (vgl. z.B. (Spahn et al. 2008a; Spahn/Wulf 2009)). Die Forscher motivieren ihre Forschungsanstrengungen mit der Schwierigkeit von Unternehmen ihre ERP-Applikationen an Ihre spezifischen Bedürfnisse anzupassen (Spahn et al. 2008a, 483). In solchen Fällen müssen Unternehmen häufig auf professionelle IT-Experten zurückgreifen. Dies führt oftmals zu langen und kostspieligen Projekten (Brehm et al. 2001).

Spahn et al. fokussieren sich in ihren Forschungsarbeiten auf die Datenzugriff von Endbenutzern auf ERP-Systeme. Sie erläutern, dass Endbenutzer die bereitgestellten Benutzungsschnittstellen von ERP-Applikationen größtenteils verwenden, um auf die für ihre jeweilige Aufgabenerfüllung relevanten Daten zuzugreifen (Spahn et al. 2008a, 484). Im Rahmen einer empirischen Untersuchung stellen die Forscher fest, dass der Zugriff auf vordefinierte Datenmengen für Endbenutzer jedoch oftmals nicht ausreichend ist (Spahn et al. 2008a, 484). Sie schlussfolgern, dass Endbenutzer fähig sein sollten, spezifische Datenabfrage selbst zu definieren, um ihre Arbeitsaufgaben optimal zu unterstützen (Spahn et al. 2008a, 486). Im Rahmen ihrer empirischen Untersuchung stellen die Forscher jedoch fest, dass Endbenutzer mit einem geringen technischen Hintergrundwissen große Probleme mit der Anpassung der existierenden Applikationen an ihre eigenen Bedürfnisse haben (Spahn et al. 2008a, 483). Die Forscher begründen dies hauptsächlich mit der hohen Komplexität der ERP-Datenmodelle sowie der verfügbaren Entwicklungswerkzeuge (Spahn et al. 2008a, 483).

Als Lösungsvorschlag entwickelt Michael Spahn im Rahmen seiner Dissertation (Spahn 2010) ein Werkzeug auf Basis eines Mashup-Paradigmas. Das Werkzeug basiert auf einer webbasierten Entwicklungsumgebung. Diese ermöglicht es über einen WYSISYG-Editor, Softwareapplikationen durch die Komposition einzelner Informationsartefakte, sogenannte Widgets, zu erstellen. Zusätzlich ist es möglich Kommunikationsbeziehung zwischen den einzelnen Widgets herzustellen. Zur Abfrage von Daten aus einem ERP-System werden über ein zweites Werkzeug sogenannte Services entwickelt. Hierzu stellt das Werkzeug eine ontologiebasierte, fachliche Abstraktionsschicht zur Verfügung. Dadurch wird es auch Endbenutzern ohne Kenntnisse von technischen Datenabfragesprachen (wie bspw. SQL) ermöglicht, eigene Services zu erstellen.

2.4.4.2 Endbenutzer-Entwicklung von mobilen Applikationen

Im Gegensatz zu EUD-Forschungsarbeiten im Umfeld von ERP-Systemen gibt es bereits mehrere Forschungsbeiträge zu EUD von mobilen Applikationen. Im Folgenden werden exemplarisch zwei Forschungsarbeiten vorgestellt. Beide Forschungsarbeiten fokussieren sich auf die mobile Endgeräteklasse Smartphones; verfolgen jedoch unterschiedliche Entwicklungsansätze.

Cappiello et al. (2013) stellen in ihrem Forschungsbeitrag eine EUD-Entwicklungsumgebung für native Smartphone-Applikationen vor. Die Entwicklungsumgebung ist als webbasierte Applikation für Desktop-Computer realisiert. Der Entwicklungsansatz basiert ähnlich wie bei

Spahn (2010) auf einem Mashup-Paradigma (siehe Kapitel 2.4.4.1). Ein Endbenutzer kann aus existierenden Dataservices im sogenannten Data Panel der Entwicklungsumgebung seine gewünschten Datenquellen selektieren. Diese kann er anschließend mit einer visuellen Schablone verknüpfen, um das Aussehen der mobilen Applikation festzulegen. Diesen Vorgang kann ein Endbenutzer für mehrere Datenservices wiederholen, um sich schließlich seine individuelle mobile Applikation zu erzeugen. Die finale Spezifikation der mobilen Applikation wird in Form eines XML-Beschreibungsformates exportiert. Dieses Beschreibungsformat kann schließlich innerhalb einer speziellen Laufzeitumgebung, welche als native Applikation auf dem Smartphone installiert ist, interpretiert und schließlich ausgeführt werden. Die Ausführungen der Autoren beschränken sich auf die technische Realisierung der Entwicklungsumgebung; eine Evaluation wird nicht beschrieben. Abbildung 2-9 illustriert die Entwicklungsumgebung mit den beiden Bereichen für die Dataservices (in der Abbildung links dargestellt) und die visuellen Schablonen (in der Abbildung rechts dargestellt).

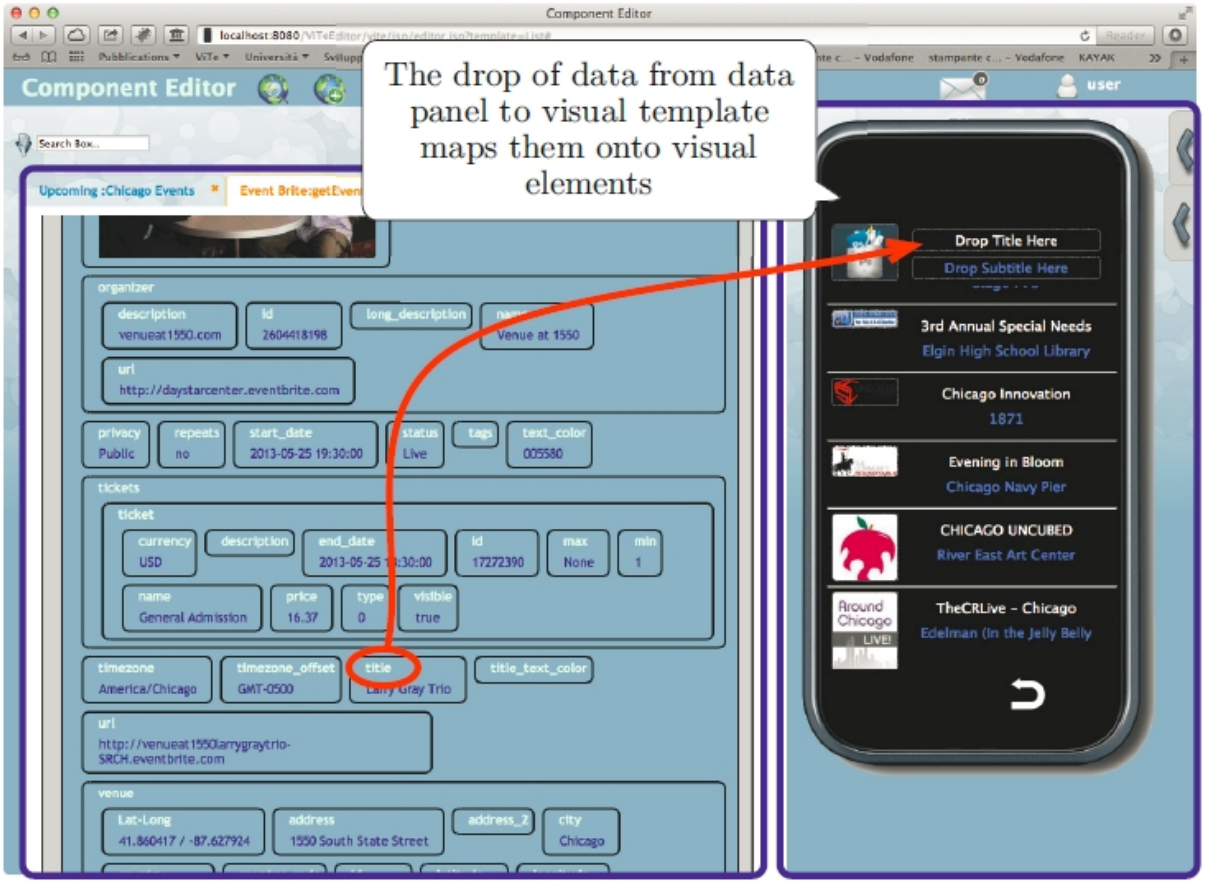

Abbildung 2-9: EUD Entwicklungsumgebung für mobile Mashup Applikationen
Quelle: (Cappiello et al. 2013, 645)

Danado and Paternó (2012) stellen in ihrem Forschungsbeitrag eine Entwicklungsumgebung für Smartphone-Applikationen auf berührungsempfindlichen Bildschirmen (engl. touchscreens) vor. Im Gegensatz zum zuvor vorgestellten Forschungsbeitrag von Cappiello et al.

(2013) ist ihre Entwicklungsumgebung selbst auf dem Smartphone realisiert. Hierdurch kann die Entwicklung und Ausführung einer mobilen Applikation auf dem gleichen Smartphone stattfinden. Die beiden Forscher leiten die Anforderungen an ihre Entwicklungsumgebung aus den Limitierungen von Smartphones im Vergleich zu Desktop-Computern ab. Dabei wird insbesondere die kleinere Bildschirmgröße von Smartphones fokussiert. Ihr Lösungsvorschlag orientiert sich an der Metapher eines Puzzles. Ihre Entwicklungsumgebung stellt wiederverwendbare Funktionalitäten in Form von Puzzleteilen zur Verfügung. Diese können über einen Drag & Drop Mechanismus zu einer neuen mobilen Applikation zusammengestellt werden. Die bereitgestellte mobile Applikation ist zugleich Entwicklungs- und Laufzeitumgebung der entwickelten mobilen Applikationen. Die Endbenutzer haben die Möglichkeit zwischen den verschiedenen Ansichten der Entwicklungsumgebung zu wechseln. Um die Entwicklung von mobilen Applikationen für unterschiedliche mobile Betriebssysteme zu unterstützten wurde der Prototyp des Entwicklungswerkzeuges mit Hilfe von Webtechnologien implementiert. Die Evaluation des Entwicklungswerkzeuges erfolgte in Form von Benutzertests. Hierzu wurden die Probanden gebeten eine Applikation mit beschriebener Funktionalität mit Hilfe des prototypisch implementierten Entwicklungswerkzeuges umzusetzen. Hierbei wurden die Probanden beobachtet und wurden nach dem Test gebeten einen Fragebogen auszufüllen. Insgesamt hat das Testergebnis gezeigt, dass die Puzzle-Metapher schnell verstanden wird. Es wurde jedoch teilweise kritisiert, dass die Kombination der einzelnen Puzzleteile im Prototypen an manchen Stellen nicht funktioniert hat. Zusätzlich wurden verbesserte Navigationsmöglichkeiten im Entwicklungswerkzeug gefordert, bspw. eine Übersichts- und Zoom-Funktionalität. Abbildung 2-10 illustriert exemplarische Ansichten der unterschiedlichen Modi der Entwicklungsumgebung. Links ist das Hauptmenü der Entwicklungsumgebung dargestellt; in der Mitte die Entwicklungsperspektive und rechts eine beispielhafte Applikation in der Laufzeitperspektive.

Abbildung 2-10: EUD-Entwicklungsumgebung "Puzzle"
Quelle: (Danado/Paternò 2012, 203)

2.4.4.3 Fazit und Diskussion

Die beschriebenen Forschungsarbeiten haben trotz ihrer unterschiedlichen Fokussierung eine Gemeinsamkeit. Alle drei Ansätze nutzen wiederverwendbare Bausteine, welche miteinander zu einer neuen Applikation kombiniert werden können. Grundlage für die Granularität der Bausteine sind jeweils Datenabfragen von einem Backendsystem oder Internetdienst. Alle drei Ansätze verwenden eine visuelle Programmierung als Interaktionstechnik. Bei der „Puzzle“ Applikation zeigt die Evaluation jedoch, dass diese bei einem Smartphone als Entwicklungsumgebung Probleme bereitet. Es stellt sich die Frage, ob die umgesetzten Ideen der vorgestellten Forschungsarbeiten miteinander kombiniert werden können, um eine Entwicklungsumgebung für mobile ERP-Applikationen auf Smartphones zu konzipieren.

3 Charakteristiken mobiler ERP-Applikationen

Im Folgenden werden die Charakteristiken von mobilen ERP-Applikationen untersucht. Hierzu wird zunächst erläutert, was in dieser Arbeit unter einer mobilen ERP-Applikation verstanden wird. Anschließend werden Charakteristiken mobiler ERP-Applikationen aus der Literatur abgeleitet. Das Ergebnis wird darauffolgend um zusätzliche Charakteristiken auf Basis einer Analyse von 20 existierenden mobilen ERP-Applikationen der SAP AG erweitert. Das übergreifende Ziel der Untersuchung ist die Nutzung der identifizierten Charakteristiken für die Anforderungsermittlung an das umzusetzende Entwicklungswerkzeug in der vorliegenden Forschungsarbeit. Insbesondere soll ein Teil der identifizierten Charakteristiken als Grundlage für die Gestaltungen geeigneter wiederwendbarer Bausteine zur Entwicklung mobiler ERP-Applikationen dienen.

3.1 Motivation und Begriffsverständnis

Durch das rasante Wachstum und die mittlerweile weite Verbreitung von Smartphones und mobilen Unternehmenslösungen wird auch im Bereich mobiler ERP-Lösungen ein großes Wachstumspotenzial gesehen (Dospinescu et al. 2008, 92 ff.). Es wird angenommen, dass durch den Einsatz mobiler ERP-Lösungen Geschäftsprozesse effizienter gestaltet werden können und die Mitarbeiterzufriedenheit gesteigert werden kann (Gronau et al. 2012, 25). Viele Hersteller von ERP-Systemen bieten aktuell entsprechende Erweiterungen für den mobilen Zugriff auf ihre ERP-Systeme an. Zudem existieren am Markt eigenständige Lösungen unabhängiger Softwareanbieter für den mobilen Zugriff auf ERP-Systeme.

Bei „mobile ERP“ handelt es sich um ein relativ junges Forschungsgebiet, bei dem sich noch kein einheitliches Begriffsverständnis etablieren konnte (Gronau et al. 2012, 24). Mobile ERP kann als Teilgebiet des „Mobile Business“ betrachtet werden, welches die Gesamtheit aller Aktivitäten, Prozesse und Applikationen im Unternehmen, welche mit mobilen Technologien durchgeführt oder unterstützt werden umfasst (Lehner 2003, 6 f.). Für zugehörige Applikationen werden teils unterschiedliche Begrifflichkeiten verwenden, wie bspw. „Business Apps“ (Lüerßen/Matschiner 2013, 16), „mobile Geschäftsapplikationen“ (Léopold/Bischel 2012,

127) oder „mobile Unternehmensapplikationen“ (Homann et al. 2013b, 23) verwendet. Derartige Applikationen haben den Zweck die Geschäftsprozesse in Unternehmen zu unterstützen bzw. zu verbessern. Demzufolge kann unter einer *mobilen ERP-Applikation* eine mobile Applikation verstanden werden, welche bestimmte Aktivitäten eines in einem ERP-System abgebildeten Geschäftsprozesses unterstützt. Nach Gronau et al. wird unter „mobile ERP-Software“ eine „ERP-Software verstanden, die auf einem mobilen Endgerät betrieben werden kann oder von der Informationen abgerufen werden können“ (Gronau et al. 2012, 24). Dieses Begriffsverständnis umfasst die beiden Kernaufgaben von mobiler ERP-Software: 1) das Abfragen von benötigten Daten aus dem ERP-System und 2) das Anzeigen der abgefragten Daten auf dem mobilen Gerät (Kurbel/Dabkowski 2003, 77). In der vorliegenden Arbeit wird unter einer Applikation immer eine Software verstanden, weshalb die Begriffe „mobile ERP Software“ und „mobile ERP-Applikation“ synonym betrachtet werden. Um das Begriffsverständnis für mobile ERP-Applikationen weiter zu schärfen und spezifische Charakteristiken für die Domäne „mobile ERP“ herauszuarbeiten, werden zunächst Charakteristiken mobiler ERP-Applikationen aus der Literatur identifiziert. Diese werden anschließend durch Charakteristiken existierender mobiler ERP-Applikationen erweitert.

Aktuell haben bereits mehrere ERP-Anbieter und Drittanbieter mobile ERP-Applikationen in ihrem Produktportfolio. Besonders hoch ist das aktuelle Angebot an mobilen ERP-Applikationen zur Unterstützung der Bereiche Vertrieb, Außendienst und Berichtswesen (Gronau et al. 2012, 23). Der Einsatz von mobilen ERP-Applikationen kann unterschiedliche Nutzenvorteile mit sich bringen. Beispielsweise kann die Effizienz der mobilen Mitarbeiter durch die Vermeidung von Doppelarbeiten verbessert werden und notwendige Abstimmungszeiten reduziert werden (Gronau et al. 2012, 25). Zudem kann die Verfügbarkeit von Echtzeitdaten aus dem ERP-System auf den mobilen Endgeräten die Entscheidungsfindung vor Ort bei Kunden erleichtern (Gronau et al. 2012, 25). Durch entsprechende mobile ERP-Applikationen können Mitarbeiter Reise- oder Wartezeiten für administrative Aufgaben, wie beispielsweise die Erfassung von Arbeitszeiten nutzen, wodurch die verfügbare Arbeitszeit für wertvollere Aktivitäten genutzt werden kann. Ein weiterer potenzieller Vorteil ist in der Mitarbeiterakzeptanz zu finden. ERP-Systeme gelten allgemein als recht komplex und nicht benutzungsfreundlich (siehe Kapitel 2.1.5.1). Die Nutzung von ERP-Funktionalitäten auf einem Smartphone ist mit der Hoffnung verbunden, diese Situation zu verbessern. Hierbei sollen die Mitarbeiterakzeptanz und Bedienung der entsprechenden ERP-Funktionalitäten durch die Reduktion der Funktionalitäten auf das Wesentliche und eine intuitive Bedienung über ein Multi-Touch-Display verbessert werden (Gronau et al. 2012, 26).

3.2 Charakteristiken aus der Literatur

Im Folgenden werden die aus der Literatur identifizierten Charakteristiken mobiler ERP-Applikationen erläutert. Diese sind entsprechend mit C1 bis C7 gekennzeichnet.

C1: Daten und Funktionalitäten stammen aus dem ERP-System

Mobile Unternehmensapplikationen wurden im vorherigen Kapitel als mobile Applikationen definiert, welche die Geschäftsprozesse eines Unternehmens unterstützten. Um die Geschäfts-

prozesse über mobile Geräte unterstützten zu können, müssen die zugehörigen Daten und Funktionalitäten bereitgestellt werden.

Im Fall von ERP-Systemen sind die Daten in einer Datenbank gespeichert, während die Funktionalitäten über einen speziellen Applikationsserver zur Verfügung gestellt werden (siehe Kapitel 2.1.5). Der Zugriff auf die benötigten Daten und Funktionalitäten in ERP-Systemen wird als eine der Herausforderungen und Kernaufgaben von mobilen ERP-Lösungen angesehen (Kurbel/Dabkowski 2003, 77). Die Herausforderung wird mit der hohen Komplexität der ERP-Datenschemata begründet. Als möglichen Lösungsvorschlag stellen Kurbel et al. (2003) eine mehrschichtige Architektur vor. Diese greift über eine spezielle Komponente auf die Daten des ERP-Systems zu und transformiert die abgefragten Daten in ein für mobile Endgeräte geeignetes Format. Hierbei greifen die Autoren direkt über SQL[6]-Befehle auf die Datenbank des ERP-Systems zu. Ein solcher, direkter Datenbankzugriff wird jedoch bei vielen etablierten ERP-Systemen nicht unterstützt. Stattdessen stellen die ERP-Systeme spezielle Programmierschnittstellen für den Zugriff auf bestimmte Daten und Funktionalitäten zur Verfügung (siehe Kapitel 2.1.5.2). Hierbei besteht die Herausforderung die bereitgestellten Programmierschnittstellen richtig zu parametrisieren.

C2: Stellen die Präsentationsschicht in einem mobilen ERP-Applikationsszenario dar

Als Folge der Charakteristik C1 stellen mobile ERP-Applikationen keine eigenen ERP-Daten oder -Funktionalitäten zur Verfügung. Eine Ausnahme sind lokal gespeicherte Daten bei offline-fähigen Applikationen (Beckert et al. 2012, 143; Homann et al. 2013b, 51). Hierbei werden ERP-Daten auf dem mobilen Gerät gespeichert. Dabei handelt es sich i.d.R. um bereits vom ERP-System abgefragte Daten. Hierdurch können die lokal gespeicherten Daten auch im Falle einer fehlenden Netzwerkverbindung genutzt werden. Jedoch handelt es sich in diesem Fall nur um eine Zwischenspeicherung der ERP-Daten; der primäre Datenspeicher bleibt das ERP-System.

Daraus resultiert die Fokussierung von mobilen ERP-Applikationen auf die Präsentationsschicht bzw. die Benutzungsschnittstelle. Traditionelle Desktop-ERP-Applikationen besitzen i.d.R. eine webbasierte Benutzungsschnittstelle und/oder eine speziell für eines oder mehrere Betriebssysteme entwickelte Benutzungsschnittstelle (z.B. SAP GUI für SAP-ERP-Systeme). Obwohl auch Smartphones i.d.R. einen Webbrowser besitzen, eignen sich die traditionellen ERP-Benutzungsschnittstellen nicht für mobile Geräte. Dies liegt vor allem an den umfangreichen Ein- und Ausgabemasken bei traditionellen ERP-Benutzungsschnittstellen auf der einen Seite und der kleinen Displaygröße von mobilen Endgeräte auf der anderen Seite (Gronau et al. 2012, 25). Um eine Adaption der traditionellen ERP-Ein- und Ausgabemasken zu ermöglichen stellen Kurbel et al. (2006, 147) drei verschiedene Adaptionsmechanismen vor:

- **Adaption des Inhaltes:** hierbei wird hauptsächlich das Weglassen unnötiger Inhalte fokussiert.

[6] Structured Query Language: Standard zur Abfrage von Daten aus relationalen Datenbanken

- **Adaption der Formatierung und der Anzeige- und Bedienelemente:** hierbei steht die Auswahl einer geeigneten Schriftgröße und die Auswahl geeigneter Anzeige- und Bedienelemente im Vordergrund.
- **Adaption der Navigationsstruktur:** hierbei sollen größere Inhaltsstücke in kleinere Teile zerlegt werden und die Navigationsstruktur entsprechend angepasst werden.

Aus diesen Adaptionsmechanismen wird ersichtlich, dass bei der Adaption von bestehenden Desktop-Benutzungsschnittstellen oder bei der Neuentwicklung von Benutzungsschnittstellen für mobile Endgeräte unterschiedliche Aspekte berücksichtigt werden müssen.

C3: Besitzen unterschiedliche Applikationsschwerpunkte

Mobile Unternehmensapplikationen können hinsichtlich ihres Schwerpunktes und ihrer Zielgruppe in unterschiedliche Applikationskategorien eingeteilt werden. Mall et al. (2012, 48 ff.) sowie Martino und Philipp (2012, 36 f.) unterscheiden zwischen den folgenden vier Applikationskategorien: Produktivitäts-, Prozesszentrierte-, Kunden- und Analytische Applikationen. Beckert et al. (2012, 188 ff.) führen sogenannte Industrie-Applikationen als fünfte Applikationskategorie an. Da Geschäftsprozesse jedoch häufig eine industriespezifische Ausprägung besitzen, wird diese Applikationskategorie der Kategorie Prozesszentrierte-Applikationen zugeordnet. Im Folgenden werden die verschiedenen Applikationskategorien erläutert.

1) Produktivitätsapplikationen

Produktivitätsapplikationen unterstützen Mitarbeiter i.d.R. bei der Durchführung einer speziellen, kurzweiligen Aktivität. Dies sind oftmals kurze Datenabfragen oder Genehmigungen. Ein wesentliches Merkmal von Produktivitäts-Applikationen ist ihre einfache und intuitive Bedienung. Ein Einarbeitungssaufwand zum Erlernen der Bedienung sollte bei dieser Applikationskategorie nicht notwendig sein (Mall et al. 2012, 207). Daher sollten sich zugehörige Applikationen durch einfache Navigationsstrukturen und die Nutzung gängiger Bedienelemente und Layoutstrukturen des jeweiligen mobilen Betriebssystems auszeichnen (Mall et al. 2012, 207). Ein Beispiel für eine Produktivitäts-Applikationen ist eine mobile Applikation zum Abfragen der Kontaktdaten eines Kollegen (Beckert et al. 2012, 149 f.). Ein weiteres Beispiel ist das Erfassen von Reisekostenbelegen oder das Genehmigen von Urlaubsanträgen von Mitarbeitern (Mall et al. 2012, 49 und 207 ff.; Beckert et al. 2012, 160 f.). Abbildung 3-1 illustriert eine Produktivitäts-Applikation am Beispiel der Applikation „Employee Lookup“ der SAP AG.

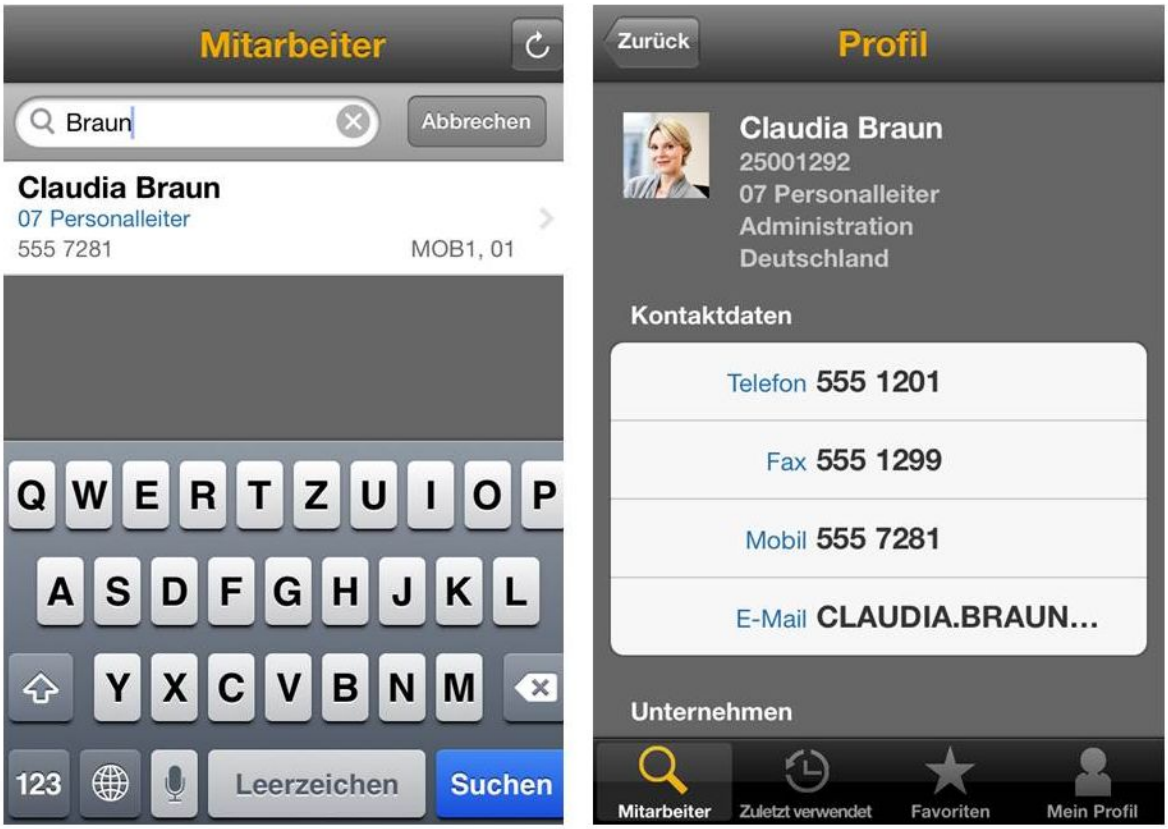

Abbildung 3-1: SAP „Employee Lookup“ Applikation als Beispiel für eine Produktivitäts-Applikation
Quelle: SAP Employee Lookup Applikation für das iPhone in Version 2.3.5

2) Prozesszentrierte Applikationen

Prozesszentrierte-Applikationen unterstützen das Kernaufgabengebiet eines mobilen Mitarbeiters. Infolge ihrer hohen Bedeutung für die Aufgabenerfüllung des Mitarbeiters werden zugehörige Applikationen auch teilweise als „mission critical“ bezeichnet (Mall et al. 2012, 49). Aufgrund des im Vergleich zu Produktivitätsapplikationen höheren Funktionsumfangs von Prozesszentrierten-Applikationen ist bei Prozesszentrierten-Applikationen i.d.R. ein gewisser Einarbeitungsaufwand erforderlich. Beispiele sind CRM Applikationen, welche u.a. Möglichkeiten zum Suchen und Pflegen von Kundendaten bereitstellen. Ein anderes Beispiel sind mobile Applikationen zur Unterstützung von Vertriebsmitarbeitern. Solche Applikationen besitzen beispielsweise einen Produktkatalog sowie die Möglichkeit direkt einen Verkaufsauftrag an das ERP-System zu senden und die voraussichtliche Verfügbarkeit und Lieferzeit des bestellten Produktes zu prüfen (Beckert et al. 2012, 182 f.). Ein anderes Beispiel ist eine mobile ERP-Applikation zur Unterstützung von Wartungsarbeiten an Industriemaschinen. Eine solche mobile Applikation kann u.a. Dokumente zur Behebung des Fehlers einer Maschinenkomponente laden und anzeigen (Beckert et al. 2012, 185 f.). Ein Beispiel aus dem medizinischen Bereich ist das Laden und Bearbeiten einer elektronischen Patienten-

akte auf dem mobilen Endgerät eines Arztes (Mall et al. 2012, 49 und 162 ff.; Beckert et al. 2012). Abbildung 3-2 illustriert ein Beispiel für eine Produktivitäts-Applikation am Beispiel der „CRM Sales“ Applikation von SAP.

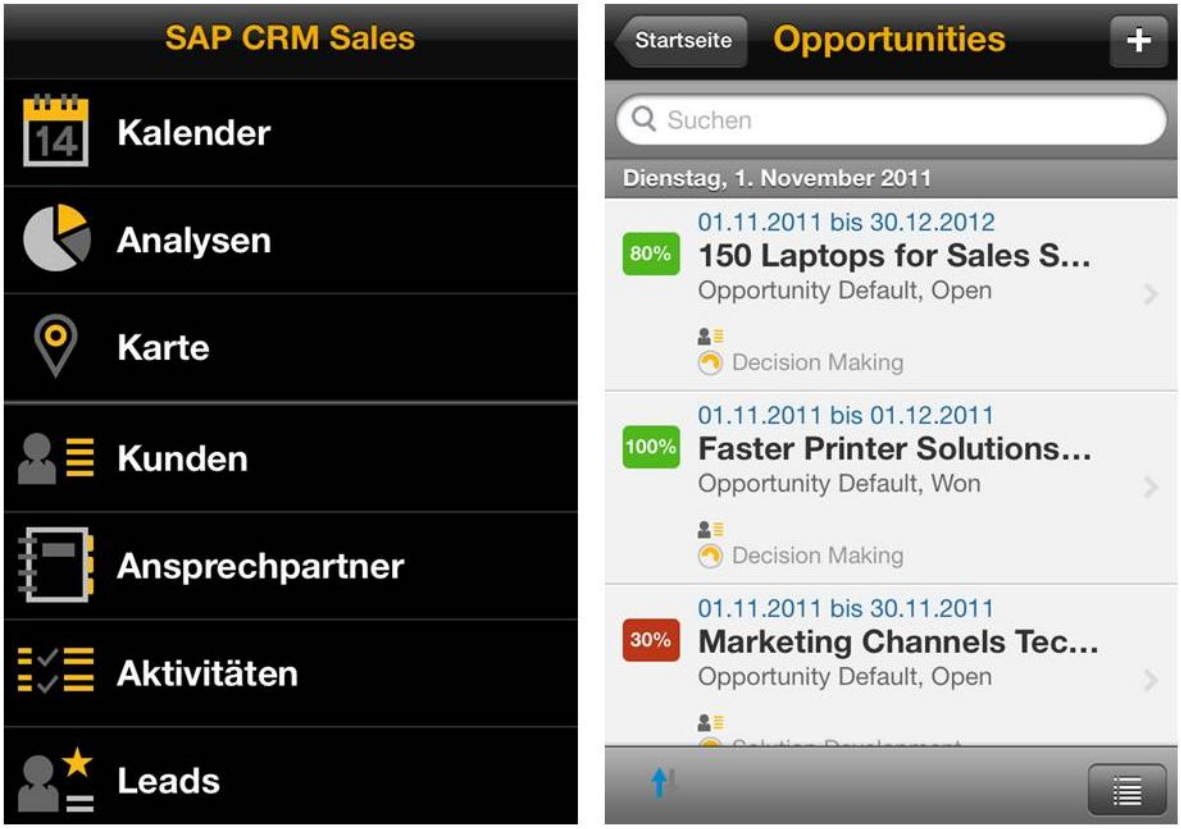

Abbildung 3-2: SAP CRM Sales Applikation als Beispiel für eine Prozesszentrierte Applikation

Quelle: SAP CRM Sales Applikation für das iPhone in Version 2.0.11

3) Kundenapplikationen

Kundenapplikationen erlauben die direkte Interaktion mit den Kunden des Unternehmens. Während bei mobilen Applikationen für Mitarbeiter häufig eine Steigerung der Mitarbeiterproduktivität erzielt werden soll, steht bei Kunden-Applikationen die Kundenbindung im Vordergrund. Ziel der Nutzung einer zugehörigen mobilen Applikation ist die Schaffung eines „positiven Erlebnisses“ für den Kunden, so dass er die mobile Applikation zukünftig gerne wieder nutzt. Ein Beispiel für eine Kunden-Applikation ist eine mobile Shopping Applikation, mit welcher der Kunde im Produktsortiment des Unternehmens stöbern kann und auch die Möglichkeit hat, direkt Bestellungen aufzugeben. Abbildung 3-3 illustriert die native iPhone Applikation von Zalando aus dem Apple App Store. Zalando ist ein größerer Online-Versandhändler für Schuhe und Mode. Zalando nutzt das ERP-System Comarch ERP Enterprise. Dabei werden u.a. die Geschäftsprozesse für die Abwicklung von Kundenaufträgen, Lieferungen und Retouren durch Comarch ERP Enterprise unterstützt (Böing 2011, 246 ff.).

Daher ist es wahrscheinlich, dass erstellte Kundenaufträge in der mobilen Applikation direkt an das ERP-System weitergereicht werden. In diesem Fall handelt es sich nach dem Begriffsverständnis dieser Arbeit um eine mobile ERP-Applikation vom Typ Kundenapplikation.

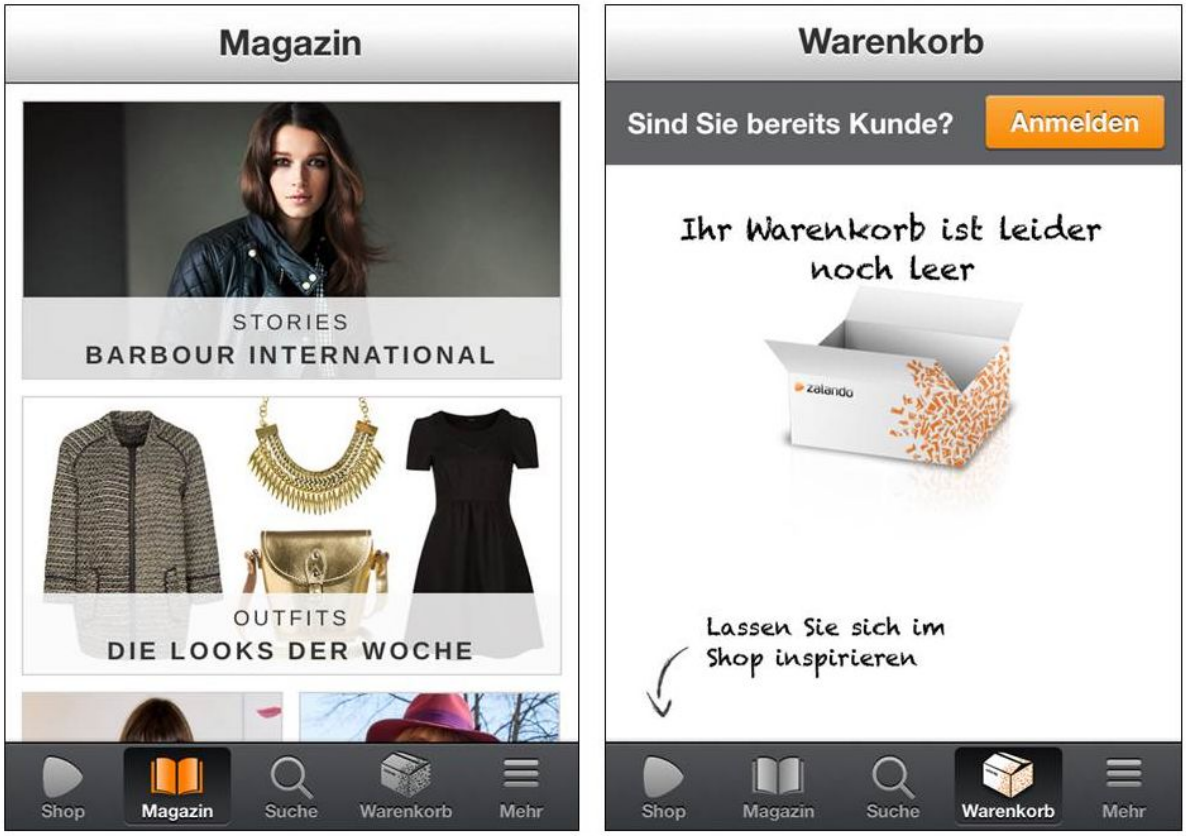

Abbildung 3-3: Zalando Applikation als Beispiel einer Kundenapplikation

Quelle: Eigene Darstellung unter Nutzung der Zalando Applikation für das iPhone; Aufruf am 18.11.2013

4) Analytische Applikationen

Analytische Applikationen verfolgen i.d.R. das Ziel wichtige Kennzahlen des Unternehmens grafisch zu visualisieren. Teilweise werden für diesen Applikationstyp auch die Begriffe mobile Business Intelligence (BI) Applikation (Bensberg 2009, 72 ff.; Beckert et al. 2012, 149) oder mobile Dashboard Applikation (Homann et al. 2013a, 20 ff-) verwendet. In diesen Applikationen kommen oftmals verschiedene Diagrammarten, wie beispielsweise Kreis- oder Balkendiagramme zum Einsatz. Neben einer statischen Darstellung der Kennzahlen erlauben es manche analytische-Applikationen auch mit den Daten zu interagieren, um beispielsweise einen höheren Detailierungsgrad der Daten anzuzeigen oder Daten eines ausgewählten Zeitfensters zu betrachten (Homann et al. 2013a). Oftmals greifen analytische-Applikationen auf BI-Systeme zu. Aber auch der Zugriff auf die transaktionalen Daten eines ERP-Systems ist denkbar. Ein Beispiel für eine analytische Applikation ist die Anzeige von Verkaufsaufträgen für bestimmte Produkte eines Unternehmens. Ein weiteres Beispiel ist die Visualisierung

offener Tickets in einem Team von Servicetechnikern. Abbildung 3-4 illustriert die Applikation Business Object Explorer von SAP als Beispiel für eine analytische Applikation.

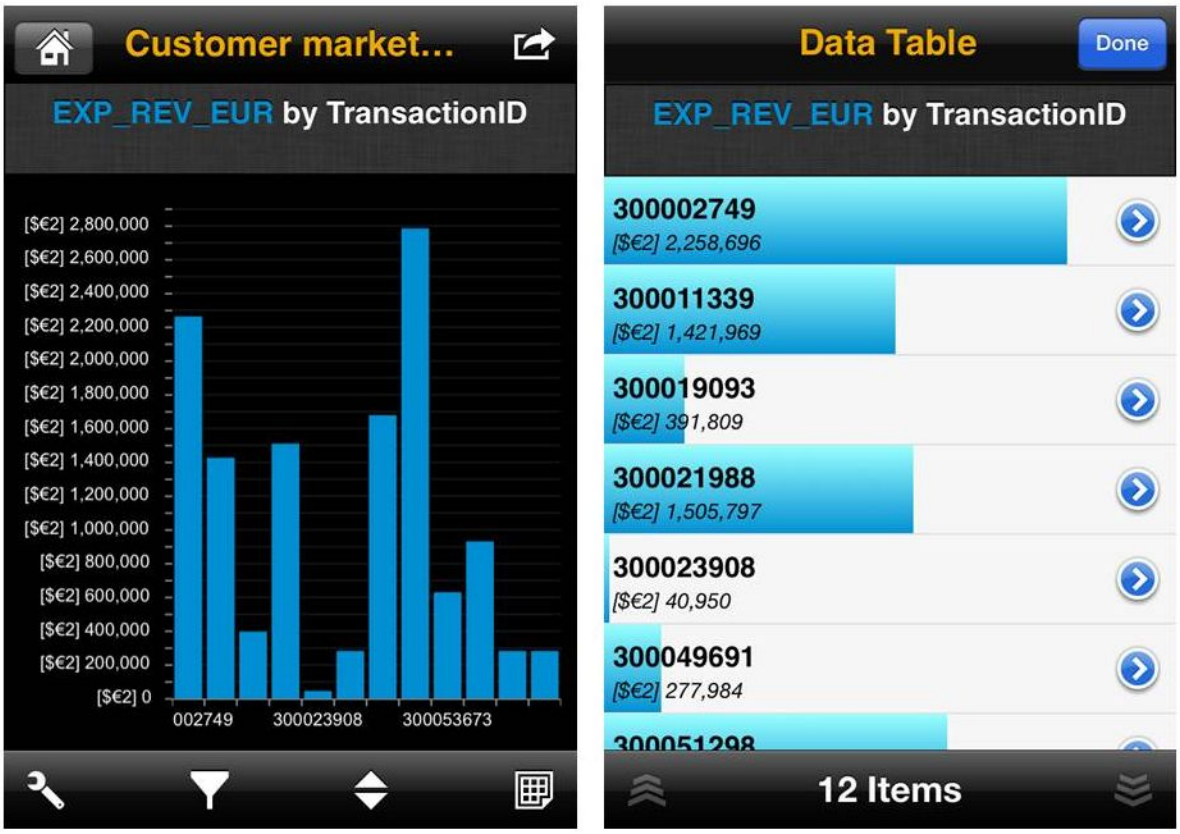

Abbildung 3-4: SAP Business Objects Explorer als Beispiel einer analytische Applikation
Quelle: SAP Business Object Explorer für das iPhone; Aufruf am 18.11.2013

C4: Nutzen unterschiedliche Implementierungsvarianten

Insbesondere in praxisorientierten Literaturquellen wird zwischen unterschiedlichen Implementierungsvarianten von mobilen ERP-Applikationen unterschieden. Dabei werden insbesondere die vier Varianten genannt: 1) Native Applikationen, 2) Hybride-Applikationen, 3) Container Applikationen und 4) Webapplikationen (Beckert et al. 2012, 138 ff.; Homann et al. 2013b, 51 ff.).

Native Applikationen

Native Applikationen werden speziell für ein mobiles Betriebssystem entwickelt. Dabei kommen i.d.R. Entwicklungswerkzeuge zum Einsatz, welche vom Hersteller des mobilen Betriebssystems bereitgestellt werden. Beispiele sind XCode zur Entwicklung von iOS-Applikationen oder das Android Development Kit (ADT) zur Entwicklung von Android Applikationen. Neben den Entwicklungswerkzeugen unterscheidet sich die Entwicklung nativer Applikationen oftmals auch in der eingesetzten Programmiersprache. Apple setzt für

die iOS-Entwicklung beispielsweise auf die Programmiersprache Objective-C, die Open-Handset-Alliance setzt bei der Android-Entwicklung auf die Programmiersprache Java (Homann et al. 2013b, 349 ff.).

Native Applikationen werden direkt in Maschinencode übersetzt. Dies resultiert in den besten Laufzeiteigenschaften unter den genannten Applikationstypen. Zudem besitzen sie die umfangreichsten Möglichkeiten zur Nutzung gerätespezifischer Funktionalitäten. Ein Nachteil nativer Applikationen ist, dass sie i.d.R. nur auf einem speziellen mobilen Betriebssystem lauffähig sind. Müssen mehrere mobile Betriebssysteme unterstützt werden, ist der Aufwand zur Bereitstellung mehrerer nativer Applikationen daher vergleichsweise hoch. Daher empfehlen praxisorientierte Literaturquellen diesen Applikationstyp „nur“ bei der Umsetzung von besonders wichtigen Applikationen bzw. zur Unterstützung von kritischen Geschäftsaktivitäten (Mall et al. 2012, 272 f.; Beckert et al. 2012, 139 f.; IBM 2012, 2 ff.).

Web-Applikationen

Web-Applikationen auf mobilen Endgeräten werden analog zu Web-Applikationen auf Desktop-Computern innerhalb eines Webbrowsers ausgeführt. Ihr großer Vorteil ist ihre Plattformunabhängigkeit. Jedoch muss hierbei bedacht werden, dass die Plattformunabhängigkeit durch unterschiedliche Webbrowser Implementierungen eingeschränkt wird.

Web-Applikationen werden üblicherweise mit Webtechnologien wie HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) und der Programmiersprache JavaScript implementiert. Dabei werden oftmals sogenannte HTML5-Frameworks eingesetzt, welche u.a. die Implementierung von typischen Bedienelementen auf mobilen Geräten vereinfachen. Beispiele für derartige HTML5-Frameworks sind jQT[7], JQuery-Mobile[8] oder Sencha Touch[9] (Bai 2011, 9 ff.). Nachteilig bei Web-Applikationen ist, dass sie im Gegensatz zu nativen-Applikationen einen eingeschränkteren Zugriff auf Gerätefunktionalitäten besitzen. Zudem müssen die Inhalte einer Web-Applikation zunächst vom Webbrowser übersetzt werden, bevor sie dargestellt werden können. Dies führt zu schlechteren Laufzeiteigenschaften als bei nativen-Applikationen. Der Aufwand zur Implementierung einer Web-Applikation ist oftmals geringer als bei nativen-Applikationen, da vielen Unternehmen bereits Kenntnisse in der Nutzung von Webtechnologien besitzen (Beckert et al. 2012, 140 f.; IBM 2012, 4 ff.; Homann et al. 2013b, 53 f.).

Hybrid-Applikationen

Hybrid-Applikationen stellen eine Kombination von nativen-Applikationen und Web-Applikationen dar. Sie werden analog zu Web-Applikationen auf der Basis von Webtechnologien entwickelt. Jedoch werden Hybrid-Applikationen mit einer zusätzlichen Programmierbibliothek ausgeliefert, welche oftmals als *Runtime-Wrapper* bezeichnet wird (Homann et al. 2014a, 33). Diese Programmierbibliothek ist mit nativen Technologien des jeweiligen mobi-

[7] http://jqtjs.com, zugegriffen am 4.11.2013
[8] http://jquerymobile.com, zugegriffen am 4.11.2013
[9] http://www.sencha.com/products/touch , zugegriffen am 4.11.2013

len Betriebssystems implementiert und erlaubt den Zugriff auf die Gerätefunktionalitäten über eine spezielle Programmierschnittstelle. Ein beispielhafter Ansatz zur Umsetzung von Hybrid-Applikationen ist PhoneGap[10] von Adobe (Beckert et al. 2012, 141 f.; Homann et al. 2013b, 54 ff.).

Container-Applikationen

Container-Applikationen sind ähnlich wie Hybrid-Applikationen eine Kombination aus nativen-Applikationen und Web-Applikationen. Die eigentlichen Applikationen werden ebenfalls mit Hilfe von Webtechnologien umgesetzt. Die Container-Applikationen werden jedoch innerhalb einer speziellen Laufzeitumgebung, dem *Container*, ausgeführt. Ein Container ist eine native Applikation, die analog zu einem Runtime-Wrapper bei Hybrid-Applikationen eine Programmierschnittstelle für den Zugriff auf Gerätefunktionalitäten bereitstellt. Im Gegensatz zu Hybrid-Applikationen werden Container Applikationen jedoch i.d.R. in Zusammenspiel mit einer sogenannten Mobile Enterprise Application Platform (MEAP) (siehe Charakteristik C7) eingesetzt. Container stellen neben der Laufzeitumgebung für Container-Applikationen eine Reihe von Mechanismen zur sicheren Kommunikation mit dem zugehörigen MEAP-Server zur Verfügung (Beckert et al. 2012, 142 f.; Homann et al. 2013b, 55 f.).

Abbildung 3-5illustriert die Unterschiede der genannten Applikationstypen für mobile ERP-Applikationen.

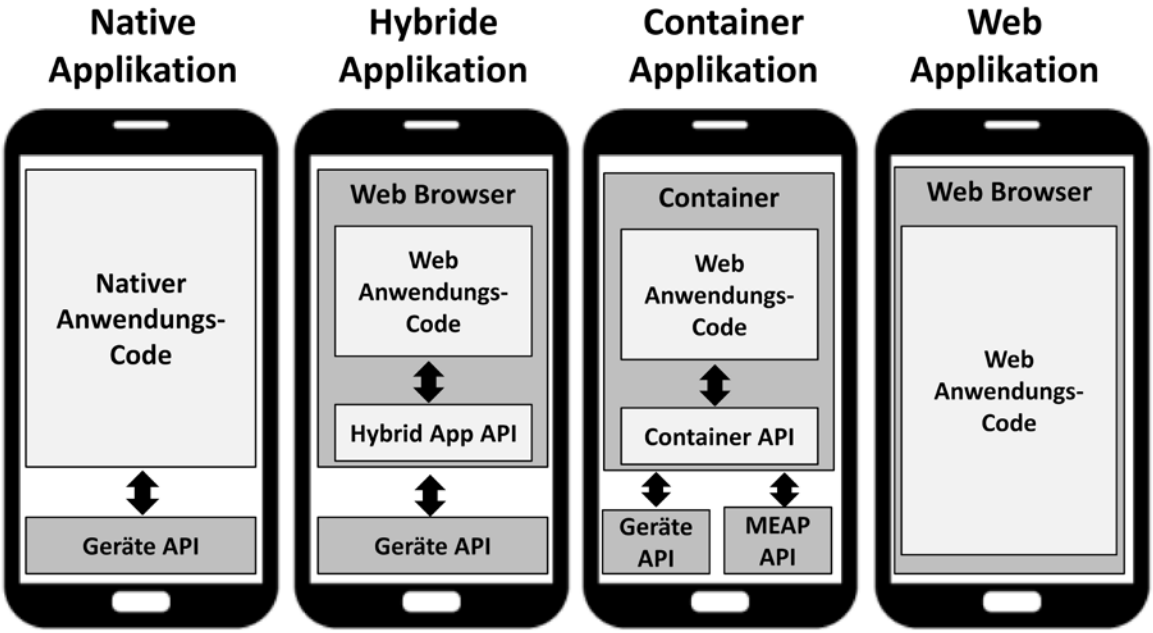

Abbildung 3-5: Implementierungsvarianten mobiler ERP-Applikationen

Quelle: (Homann et al. 2014a, 34)

[10] https://build.phonegap.com, zugegriffen am 4.11.2013

C5: Unterschiedliche Verfahren zur Datenspeicherung werden unterstützt

Bei mobilen ERP-Applikationen kann in Bezug auf die Datenspeicherung zwischen Online- und Offline-fähigen Applikationen unterschieden werden (Beckert et al. 2012, 143; Homann et al. 2013b, 51). *Online Applikationen* benötigen für jeden Datenzugriff eine Verbindung zum ERP-System. Sollte keine Verbindung zum ERP-System hergestellt werden können, beispielsweise falls das mobile Gerät keine Netzwerkverbindung besitzt, so ist kein Datenzugriff möglich. Online Applikationen eigenen sich vor allem für ERP-Datensätze, welche sich häufig ändern. Zusätzlich liegt ihr Vorteil darin, dass keine sicherheitskritischen Daten auf dem mobilen Gerät gespeichert werden (Beckert et al. 2012, 143; Homann et al. 2013b, 51).

Bei *Offline-fähigen Applikationen* werden die vom ERP-System abgefragten Daten hingegen in einer lokalen Datenbank auf dem mobilen Endgerät gespeichert. Somit kann auch ohne Netzwerkverbindung mit der zugehörigen Applikation gearbeitet werden. Offline-fähige Applikationen sind jedoch aufwändiger in der Implementierung. Dies liegt vor allem daran, dass ein spezieller Synchronisationsmechanismus bereitgestellt werden muss, welcher die Konsistenz zwischen den Daten im ERP-System und den lokal gespeicherten Daten sicherstellen muss (Beckert et al. 2012, 143; Homann et al. 2013b, 51).

C6: ERP-Daten und -Funktionalitäten werden reduziert

Eine weitere Eigenschaft von mobilen ERP-Applikationen ist, dass i.d.R. nur ein spezifischer ERP-Datensatz und/oder eine spezifische ERP-Funktionalität benötigt wird. Grund dafür ist, dass die Einschränkungen mobiler Endgeräte eine Reduktion der bereitgestellten ERP-Daten und -Funktionalitäten in einer mobilen ERP-Applikation erfordern (Gronau et al. 2012, 26). Ein Beispiel ist das Abfragen von Dienstreiseanträgen von Mitarbeitern und deren Genehmigung. In diesem Beispiel müssen in der Regel nicht alle im ERP-System verfügbaren Attribute für das Business Objekt „Mitarbeiter“ angezeigt werden. Stattdessen erfolgt eine Reduktion auf die für die mobile Applikation notwendigen Attribute. Dies verbessert gleichzeitig die Leistungsfähigkeit der Applikation, da die zu übertragende Datenmenge und die hierdurch verursachten Ladevorgänge reduziert werden (Mall et al. 2012, 237 ff.).

Die Hersteller von Entwicklungswerkzeugen für mobile ERP-Applikationen bieten unterschiedliche Möglichkeiten um ERP-Daten und -Funktionalitäten für mobile ERP-Applikationen bedarfsgerecht zu reduzieren. Das Ziel hierbei ist neben der Reduktion, die Bereitstellung einer wiederverwendbaren Einheit für den jeweiligen ERP-Datensatz oder die jeweilige ERP-Funktionalität (Beckert et al. 2012, 127). Ein Beispiel sind die sogenannten „Mobile Business Objects“ (MBOs) der SAP Mobile Platform (Mall et al. 2012, 237 ff.; Homann et al. 2013b, 183 ff.). Ein weiteres Beispiel sind die sogenannten Gateway Services der Komponente NetWeaver Gateway (Beckert et al. 2012, 239 ff.; Homann et al. 2013b, 44 ff.).

C7: Nutzung einer mobilen Unternehmensplattform

Eine weitere Charakteristik mobiler ERP-Applikationen ist, dass diese oftmals nicht direkt auf das ERP-System zugreifen. Stattdessen steuert eine spezielle Middleware-Plattform die Kommunikation zwischen den mobilen ERP-Applikationen und dem ERP-System (Mall et al.

2012, 89 ff.; Homann et al. 2013b, 75 ff.). Für eine solche Middleware-Plattform wird i.d.R. der vom Marktforschungsunternehmen Gartner geprägte Begriff *Mobile Enterprise Application Platform* (MEAP) verwendet (Gartner 2009). In der deutschsprachigen Literatur wird teilweise auch der Begriff mobile Unternehmensplattform verwendet (Martino/Philipp 2012, 34 ff.; Homann et al. 2014a, 32 ff.). Neben dem Begriff hat Gartner auch drei Regeln aufgestellt, welche die Zweckmäßigkeit einer MEAP beurteilen (Gartner 2009, 2; Beckert et al. 2012, 122 f.). Demzufolge lohnt sich der Einsatz einer MEAP falls:

- drei oder mehr mobile Applikationen im Unternehmen existieren oder geplant sind
- falls drei oder mehr mobile Betriebssysteme unterstützt werden sollen
- falls die Integration mit mindestens drei Backendsystemen (z.B. ERP-System, Datenbanken) durchgeführt werden soll

Eine MEAP stellt folgende Kernfunktionalitäten bereit (Beckert et al. 2012, 122; Homann et al. 2013b, 39 f.):

- ein Werkzeug für die Integration der Backendsysteme, wie z.B. einem ERP-System
- eine Verwaltungskonsole für mobile Endgeräte
- eine Programmierschnittstelle
- Programmierbibliotheken, Konnektoren für verschiedene Backendsysteme sowie wiederverwendbare Entwicklungskomponenten
- Eine Entwicklungsumgebung mit Modellierungswerkzeug

Durch den Einsatz einer MEAP kann der Zugriff der mobilen Applikationen auf das ERP-System zentral gesteuert und überwacht werden. Dies erleichtert die Konfiguration und Wartung der mobilen Infrastrukturkomponenten. Manche MEAPs unterstützen auch die Zwischenspeicherung von häufig abgefragten Backenddaten. Hierdurch kann die Anzahl der Anfragen an ein ERP-System deutlich reduziert werden und gleichzeitig die Geschwindigkeit der Datenabfrage der mobilen Applikationen erhöht werden (Homann et al. 2013b, 184 ff.). Am Markt sind aktuell mehrere MEAP Implementierungen erhältlich. Zu den am weitesten verbreiteten MEAP Implementierungen zählen die SAP Mobile Platform (vgl. z.B. Homann et al. 2013b, 121 ff.) und IBM Worklight (IBM 2013). Abbildung 3-6 illustriert das Konzept einer MEAP.

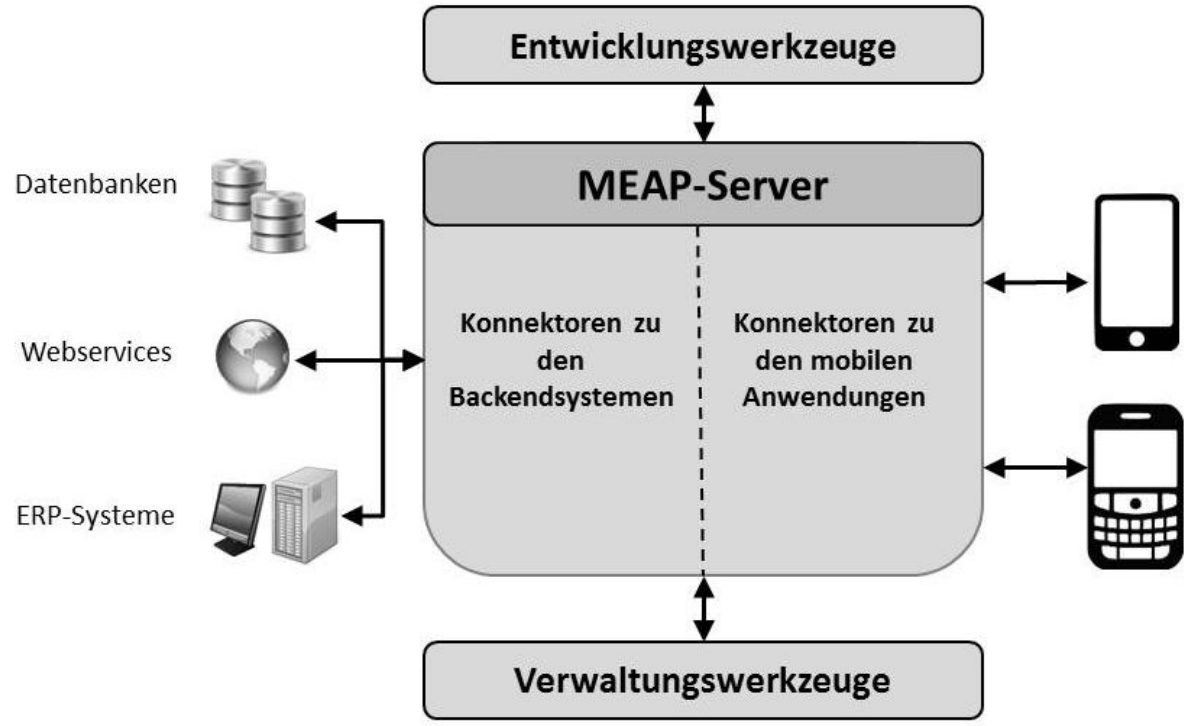

Abbildung 3-6: Konzept einer Mobilen Unternehmensplattform
Quelle: In Anlehnung an (Homann et al. 2013b, 76)

3.3 Charakteristiken existierender mobiler ERP-Applikationen

Um die aus der Literatur identifizierten Charakteristiken zu ergänzen wurden existierende mobile ERP-Applikationen analysiert. Ein größerer Teil der Ergebnisse wurde bereits im Rahmen des Veröffentlichungsprozesses dieser Dissertation in folgendem Beitrag publiziert: (Homann et al. 2013c). Das Vorgehen und die zentralen Ergebnisse werden im Folgenden beschrieben.

3.3.1 Untersuchungsgegenstand und -vorgehen

Die Literaturanalyse hat gezeigt, dass mobile ERP-Applikationen Daten und Funktionalitäten aus dem ERP-System nutzen (siehe Charakteristik C1 im Kapitel 3.2) und sich folglich auf die Repräsentation der Benutzungsschnittstelle fokussieren (siehe Charakteristik C2 in Kapitel 3.2). Aus diesem Grund erfolgte die Analyse der existierenden, mobilen ERP-Applikationen aus der Perspektive der Benutzungsschnittstelle. Hierzu wurde eine *Task Analyse* für jede Applikation durchgeführt und das Ergebnis in sogenannten *ConcurTaskTrees* (CTT) (Paterò 2000, 39 ff.) modelliert.

Als Untersuchungsgegenstand wurden 20 existierende mobile ERP-Applikationen der SAP AG für das iPhone von Apple ausgewählt. Während der Untersuchung musste eine hohe Änderungsdynamik der untersuchten Applikationen festgestellt werden, d.h. die Applikatio-

nen wurden regelmäßig vom Hersteller aktualisiert. Die Aktualisierung umfasste hierbei sowohl die Behebung von Fehlern als auch das Hinzufügen neuer Funktionalitäten. Um eine nachvollziehbares Ergebnis zu gewährleisten werden die jeweiligen Versionsnummern der untersuchten Applikationen bei der Beschreibung des Untersuchungsgegenstandes genannt.

Die Fokussierung auf mobile ERP-Applikationen der SAP AG erfolgte im Wesentlichen aus zwei Gründen. Zum einen war die SAP AG zum Zeitpunkt der Analyse das Unternehmen mit den meisten verfügbaren mobilen ERP-Applikationen im App Store von Apple. Zum anderen ist das SAP-ERP-System das in dieser Arbeit als Fallbeispiel verwendete ERP-System. Da sich die verwendeten Begrifflichkeiten und Technologien zwischen den einzelnen ERP-System-Herstellern unterscheiden, wird durch die Fokussierung auf Produkte der SAP AG eine konsistentere Untersuchung und Beschreibung der Untersuchungsergebnisse ermöglicht.

Die Fokussierung auf Applikationen aus dem App Store von Apple hatte im Wesentlichen drei Gründe. Zum einen waren im App Store zum Analysezeitpunkt mehr mobile SAP-ERP-Applikationen verfügbar als bspw. im Play Store von Google für Android Applikationen. Andere App Stores wurden aufgrund ihrer vergleichsweise geringen Bedeutung[11] nicht untersucht. Zum anderen erleichtert eine Fokussierung auf ein mobiles Betriebssystem die Vergleichbarkeit der Resultate, da Spezifika der einzelnen Betriebssysteme ausgeblendet werden können. Außerdem führt Apple eine vergleichsweise strikte Qualitätsprüfung der angebotenen mobilen Applikationen durch (Dudney/Adamson 2010, 573 f.), weshalb von der Funktionstüchtigkeit der untersuchten Applikationen ausgegangen werden kann.

Zum Startzeitpunkt der Analyse am 4. September 2012 waren 31 iPhone Applikationen der SAP AG im Apple App Store verfügbar. Um die Applikationen zu testen, stellten einige Applikationen einen Demo-Modus zur Verfügung. Mit Hilfe dieses Demo-Modus war es möglich die Applikationen ohne weitere Konfigurationsschritte zu testen. In der Untersuchung wurden nur Applikationen betrachtet, welche einen solchen Demo-Modus bereitgestellt haben. Ohne diesen Demo-Modus wäre die Installation zusätzlicher Komponenten und deren Konfiguration zum Testen der mobilen SAP-ERP-Applikationen notwendig gewesen. In vielen Fälle müsste beispielsweise zunächst der sogenannte SAP NetWeaver Gateway und die SAP Mobile Platform installiert und konfiguriert werden (vgl. z.B. (Beckert et al. 2012, 149 ff.)). Zudem wurden nur Applikationen ausgewählt, welche ihre Funktionalitäten aus einem SAP-ERP-System beziehen. Nicht betrachtet wurden daher Applikationen, welche beispielsweise einen SAP Business Objects Server benötigen oder ein SAP Business Intelligence System. Aus den 31 verfügbaren iPhone Applikationen im Apple App Store erfüllten 20 Applikationen diese Kriterien und konnten folglich für die Untersuchung verwendet werden. Tabelle 3-1 enthält die Liste der untersuchten Applikationen inkl. ihrer Versionsnummer sowie einer kurze Beschreibung[12].

[11] Siehe beispielsweise eine Statistik von Statista bzgl. der Anzahl verfügbare mobilen Applikationen in den einzelnen App Stores: http://de.statista.com/statistik/daten/studie/176828/umfrage/anzahl-der-apps-in-den-online-shops-der-betriebssystem-hersteller, zugegriffen am 20.08.2012

[12] Die Beschreibungen sind an die Beschreibungen der Applikationen im Apple App Store angelehnt.

Tabelle 3-1: Untersuchte mobile ERP-Applikationen der SAP AG

Quelle: In Anlehnung an(Homann et al. 2013c, 19)

Applikationsname	Version	Beschreibung
Business One	1.6.0	Prozesszentrierte Applikation für das ERP-System „Business One“ für kleine und mittlere Unternehmen (KMU). Beispielhafte Funktionalitäten sind das Prüfen der Lagerbestände oder der Zugriff auf den Produktkatalog.
Business ByDesign	3.5.1	Prozesszentrierte Applikation für das Software as a Service (SaaS) ERP-System „Business ByDesign“. Beispielhafte Funktionalitäten sind die Prüfung der Produktverfügbarkeit oder das Senden von Spesenabrechnungen.
Sales order notification	2.2.0	Produktivitäts-Applikation zum Zugriff auf die Daten von Kundenaufträgen.
Customer and contacts	2.2.0	Produktivitäts-Applikation für den Zugriff auf Kontaktdaten und weitere Details von Kunden.
CRM Sales	2.0.2	Prozesszentrierte Applikation für verschiedene Aufgaben im CRM-Bereich, beispielsweise den Zugriff auf Kunden- und Angebotsdaten oder die Planung von Kundenbesuchen.
Retail execution	2.1.1	Prozesszentrierte Applikation zur Verwaltung von Vertriebsaktivitäten für Konsumgüterhersteller. Beispielhafte Funktionalitäten sind der Zugriff auf die Historie der Kundenbesuche oder das Anlegen und Verwalten von Retouren bei Kundenbesuchen.
Timesheet	2.3.0	Produktivitäts-Applikation zur Verwaltung von Arbeitszeitblättern. Die Applikation kann neue Einträge in Arbeitszeitblättern anlegen und deren Genehmigungsstatus prüfen.
Employee lookup	2.2.0	Produktivitäts-Applikation für die Suche nach den Profildaten von Kollegen, insbesondere deren Kontaktdaten.
Material availability	2.2.0	Produktivitäts-Applikation für die Suche nach Materialien und deren Verfügbarkeitsprüfung.
Transport tendering	1.2.0	Produktivitäts-Applikation zum Empfangen von Frachtanfragen für Lieferungen von Logistikdienstleistern.
Travel expense approval	2.2.1	Produktivitäts-Applikation zur Genehmigung von Spesenabrechnungen von Mitarbeitern.
Travel expense report	1.0.1	Produktivitäts-Applikation zum Erstellen einer Spesenabrechnung.
Travel receipt capture	2.2.1	Produktivitäts-Applikation zum Aufnehmen von Spesenbelegen über die Kamera und Sicherung im SAP-ERP-System.
Payment approvals	2.2.0	Produktivitäts-Applikation zur Genehmigung von Zahlungsvorgängen.
Quality issue	1.1.0	Produktivitäts-Applikation zur Dokumentation von Qualitätsproblemen.
In-Store product lookup	1.0.1	Prozesszentrierte Applikation für das Verkaufspersonal

		auf den Ladenflächen zum Zugriff auf den Produktkatalog und die zugehörigen Lagerbestände.
ERP order status	2.2.0	Produktivitäts-Applikation zur Statusprüfung von Kundenaufträgen.
Customer financial fact sheet	3.0.2	Produktivitäts-Applikation zum Prüfen offener Zahlungsvorgänge von Kunden.
HR approvals	2.3.0	Produktivitäts-Applikation zum Genehmigen von Abwesenheitsanträgen oder eingereichter Arbeitszeitblätter.
Leave request	2.3.1	Produktivitäts-Applikation für das Senden und die Statusprüfung von Abwesenheitsanträgen.

3.3.1.1 Task Analyse und -Modellierung

Der Begriff *Task Analyse* stammt aus dem Bereich User-Centered Design (UCD), einem Teilbereich der HCI-Forschung. UCD verfolgt das Ziel das Verständnis zu verbessern, wie Anwender mit Benutzungsschnittstellen interagieren um bestimmte Aufgaben zu bewältigen (Limbourg/Vanderdonckt 2002, 135). Eine zugehörige Methode ist die sogenannte Task Analyse. Sie kann sowohl verwendet werden, um die Anforderungen an eine neue Benutzungsschnittstelle zu identifizieren, als auch um eine existierende Benutzungsschnittstelle zu untersuchen (Kirwan/Ainsworth 1992, 15 ff.). Das Ergebnis der Task Analyse wird gewöhnlich in einem *Task Modell* festgehalten. Ein solches Modell enthält die einzelnen (Teil-) Aufgaben, welche im Rahmen der Aufgabenbewältigung durchgeführt werden. Task Modelle sind oftmals hierarchisch strukturiert, wobei die unterschiedlichen Hierarchiestufen einen unterschiedlichen Detaillierungsgrad besitzen. Neben den Beziehungen zwischen einzelnen Hierarchiestufen, besitzen die Aufgaben einer Hierarchiestufe weitere Beziehungen, welche ihre Bearbeitungsreihenfolge spezifizieren (Limbourg/Vanderdonckt 2002, 135).

3.3.1.2 Task Modellierung mit der ConcurTaskTrees-Notation

Aktuell existieren unterschiedliche Notationen für die Task Modellierung mit teilweise unterschiedlichen Schwerpunkten. Beispiele sind: GOMS (Goals, Methods, Methods and Selection Rules), GTA (Groupware Task Analysis), UAN (User Action Notation) oder CTT (ConcurTaskTrees) (Limbourg/Vanderdonckt 2002, 137 ff.; Paterò 2000, 39 ff.). Für die vorliegende Aufgabenstellung erscheint die CTT-Notation geeignet. Durch seine grafische Visualisierung mit einfachen Symbolen und seiner hierarchischen Strukturierung sind CTT-Modelle leicht verständlich. Zudem existiert mit dem Softwarewerkzeug ConcurTaskTrees Environment[13] (CTTE) ein frei verfügbares Softwarewerkzeug zur Erstellung von CTT-Modellen.

[13] Verfügbar über: http://giove.isti.cnr.it/tools/CTTE/home, zugegriffen am 23.08.2013

Die Hauptmerkmale von CTT sind (Paterò 2000, 40 f.):

- **Fokussierung auf Aufgaben:** CTT erlaubt die Fokussierung auf die wichtigsten Aufgaben einer Interaktion zwischen einem Anwender und einem Computer-System. Implementierungsdetails werden dabei nicht berücksichtigt.
- **Hierarchische Strukturierung:** da Menschen komplexere Aufgabenstellungen häufig durch die Zerlegung in Teilaufgaben lösen, ist eine hierarchische Strukturierung i.d.R. intuitiv verständlich. Sie erlaubt zudem eine Aufgabenstellung in unterschiedlichen Detaillierungsstufen zu untersuchen.
- **Grafische Syntax:** eine grafische Syntax ist oftmals einfacher zu verstehen als eine textuelle Repräsentation. CTT nutzt eine Baumstruktur zur Visualisierung, wobei die einzelnen Aufgaben durch die „Blätter“ des Baumes dargestellt werden.
- **Modellierung der zeitlichen Abfolge:** neben der hierarchischen Zerlegung der Aufgabe unterstützt CTT auch die Modellierung der zeitlichen Abfolge der Aktivitäten. Neben sequentiellen Abfolgen können auch parallele Abfolgen modelliert werden.
- **Visualisierung des Aufgabenträgers:** durch entsprechende Symbole wird der Aufgabenträger explizit modelliert. Unter einem Aufgabenträger wird in diesem Zusammenhang die auszuführende Instanz, d.h. der menschliche Anwender oder das Computer-System, verstanden.
- **Visualisierung der Aufgabenobjekte:** neben den Aufgabenträgern können in CTT-Modellen auch die im Zuge der Aufgabendurchführung manipulierten Aufgabenobjekte modelliert werden.

CTT unterscheidet in Bezug auf den Aufgabenträger vier unterschiedliche Aufgabentypen: Anwender-Aufgaben, Applikations-Aufgaben, Interaktions-Aufgaben und Abstrakte-Aufgaben. Tabelle 3-2 enthält eine Beschreibung der jeweiligen Aufgabentypen ihrer CTT-Symbole:

Tabelle 3-2: Aufgabentypen in der ConcurTaskTree-Notation

Quelle: In Anlehnung an (Paterò 2000, 42)

Aufgabentyp	Beschreibung
Anwender-Aufgaben 	Aufgaben, welche von einem Anwender durchgeführt werden. Oftmals handelt es hierbei um kognitive Aufgaben, z.B. Überlegungen und Entscheidungen bzgl. einer Strategie zur Aufgabenbewältigung.
Applikations-Aufgaben 	Aufgaben, welche komplett durch das Computer-System bewältigt werden. Ein Beispiel ist die Abfrage von Daten aus einer Datenbank und deren anschließender Präsentation am Computerbildschirm.
Interaktions-Aufgaben 	Aufgaben, welche durch den Benutzer im Rahmen seiner Interaktion mit dem Computer-System durchgeführt werden. Beispiele sind das Drücken einer Schaltfläche oder das Editieren von Text in einem Dokument.
Abstrakte-Aufgaben 	Aufgaben, welche von mehreren Aufgabenträgern gemeinsam bewältigt werden. Dabei handelt es sich gewöhnlich um Aufgaben auf einer höheren Hierarchiestufe.

Neben der Visualisierung der Aufgaben stellt die CTT-Notation Symbole zur Spezifikation der zeitlichen Bearbeitungsreihenfolge zur Verfügung. Tabelle 3-3 enthält die für die spätere Ergebnisdarstellung dieser Arbeit relevanten zeitlichen Reihenfolgebeziehungen inkl. deren Symbole:

Tabelle 3-3: Ausgewählte Reihenfolgebeziehungen der ConcurTaskTree-Notation

Quelle: In Anlehnung an (Paterò 2000, 43 f.)

Reihenfolgebeziehung	Beschreibung
Auswahl **(A1 [ ] A2)**	Der Aufgabenträger kann zwischen unterschiedlichen Aufgaben wählen. Während der Bewältigung einer gewählten Aufgabe ist die Bewältigung der alternativen Aufgabe nicht möglich.
Aktivierung **(A1 >> A2)**	Durch die Bewältigung einer Aufgabe wird eine andere Aufgabe aktiviert. Beispielsweise ist bei einigen Applikationen zunächst eine Anmeldung möglich, bevor diese verwendet werden können.
Aktivierung mit Datenübergabe **(A1 [ ]>> A2)**	Dabei handelt es sich um einen Spezialfall der Aktivierung, bei welchem Daten von einer Vorgänger-Aufgabe an eine Nachfolge-Aufgabe übergeben werden. Dadurch ist das Ergebnis der Nachfolge-Aufgabe i.d.R. von den übergebenen Daten der Vorgänger-Aufgabe abhängig.

3.3.2 Untersuchungsergebnisse

Im Folgenden werden die Untersuchungsergebnisse der analysierten mobilen ERP erläutert. Zunächst erfolgt eine Beschreibung der identifizierten Gemeinsamkeiten. Anschließend werden die identifizierten Unterschiede der untersuchten mobilen ERP-Applikationen erläutert.

3.3.2.1 Gemeinsamkeiten der untersuchten mobiler ERP-Applikationen

Insgesamt konnten 5 Gemeinsamkeiten zwischen den untersuchten mobilen ERP-Applikationen identifiziert werden. Diese werden im Folgenden analog zu Kapitel 3.2 als Charakteristiken bezeichnet und in der gleichen Form beschrieben.

C8: Fokussierung auf ausgewählte Business Objekt Typen

Ein zentrales Merkmal der untersuchten Applikationen ist die Fokussierung auf ausgewählte Business Objekt Typen (BOTs) (engl. business object types). Dabei handelt es sich um die betriebswirtschaftlichen Objekttypen des SAP-ERP-Systems (siehe Kapitel 2.1.5.2.1). Produktivitäts-Applikationen fokussieren sich aufgrund ihres geringen Funktionsumfanges oftmals auf einen einzelnen BOT, während Prozesszentrierte-Applikationen i.d.R. mehrere BOTs umfassen. Ein Beispiel ist die Applikation „Employee lookup“, welche sich auf den BOT „Mitarbeiter“ fokussiert. Ein weiteres Beispiel ist die Applikation „Material availability“, welche sich auf das BOT „Material“ fokussiert. Die Prozesszentrierte Applikation „CRM Sales“ kann hingegen auf unterschiedliche BOTs zugreifen, beispielsweise auf Kunden, Kontakte, Aktivitäten oder Anfragen. In den meisten Fällen stehen die BOTs einer Prozesszentrierten Applikation miteinander in Beziehung. Im Beispiel der Applikation „CRM Sales“ sind beispielsweise Aufträge immer einem Kunden zugeordnet. Abbildung 3-7 skizziert einen Ausschnitt aus den beiden ersten Hierarchieebenen der Applikation „CRM Sales“. Hieraus ist ersichtlich, dass der Fokus des Einstiegsbildschirms der Applikation auf der Auswahl eines BOTs liegt. In der darauffolgenden Hierarchieebene können dessen Attribute angezeigt oder bearbeitet werden.

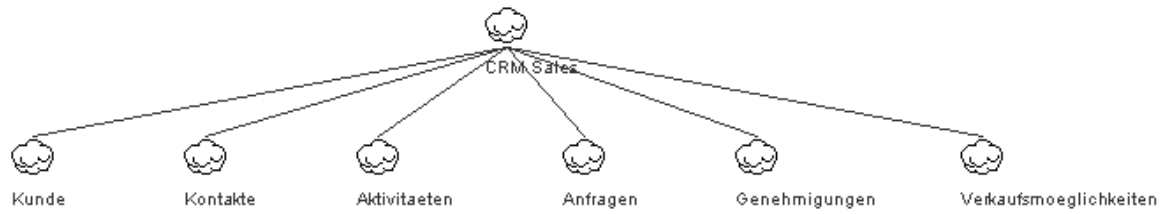

Abbildung 3-7: Ausschnitt aus dem CTT-Modell der "CRM Sales" Applikation
Quelle: Eigene Darstellung

C9: Bereitstellung ähnliche Funktionalitäten

Ein weiteres im Rahmen der Task Analyse identifiziertes Muster sind die ähnlichen Funktionalitäten, welche von den untersuchten Applikationen angeboten werden. Die meisten Applikationen erlauben die Auflistung und Sortierung von BOs oder die Suche nach einem bestimmten BO über ein ausgewähltes Attribut. Ein Beispiel ist die Applikation „Material availability“, welche Materialien wahlweise sortiert nach ihrer Warengruppe oder ihrer jeweiligen Basiseinheit auflisten kann. Zudem kann über die Materialnummer oder den Materialnamen nach einem bestimmten Material gesucht werden. Ferner können weitere Details für ein ausgewähltes Material, wie beispielsweise dessen Brutto- und Nettogewicht sowie dessen Größe angezeigt werden. Abbildung 3-8 illustriert die ersten beiden Hierarchieebenen des zugehörigen CTT-Modells.

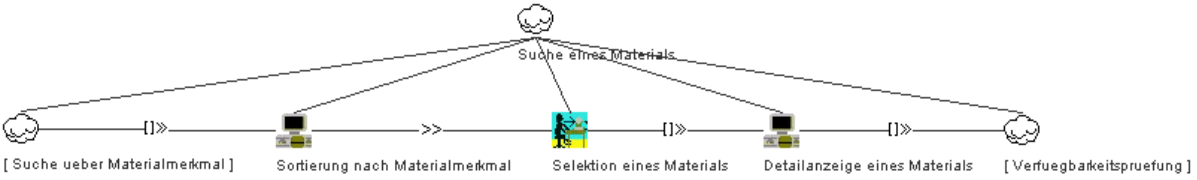

Abbildung 3-8: Ausschnitt aus dem CTT-Modell der "Material availability" Applikation
Quelle: Eigene Darstellung

Ein weiteres Beispiel ist die Applikation „HR approvals“. Hier hat der Anwender zunächst die Möglichkeit zwischen den beiden fokussierten BOTs „Abwesenheitsanträge“ und „Arbeitszeitblätter“ zu wählen. Anschließend werden die verfügbaren BOs des gewählten BOT nach dem Attribut „Eingangsdatum“ sortiert aufgelistet. Anschließend kann ein BO ausgewählt werden und dessen Details angezeigt werden. Im Beispiel von Abwesenheitsanträgen sind dies beispielsweise der Grund der Abwesenheit (z.B. Urlaub), das Start- und Enddatum sowie die Dauer in Arbeitstagen. In der Detailansicht kann ein offener Antrag schließlich genehmigt oder abgelehnt werden. Abbildung 3-9 illustriert die ersten beiden Hierarchieebenen des zugehörigen CTT-Modells inklusive der beiden darauffolgenden Hierarchieebenen für den Zweig „Abwesenheitsanträge“.

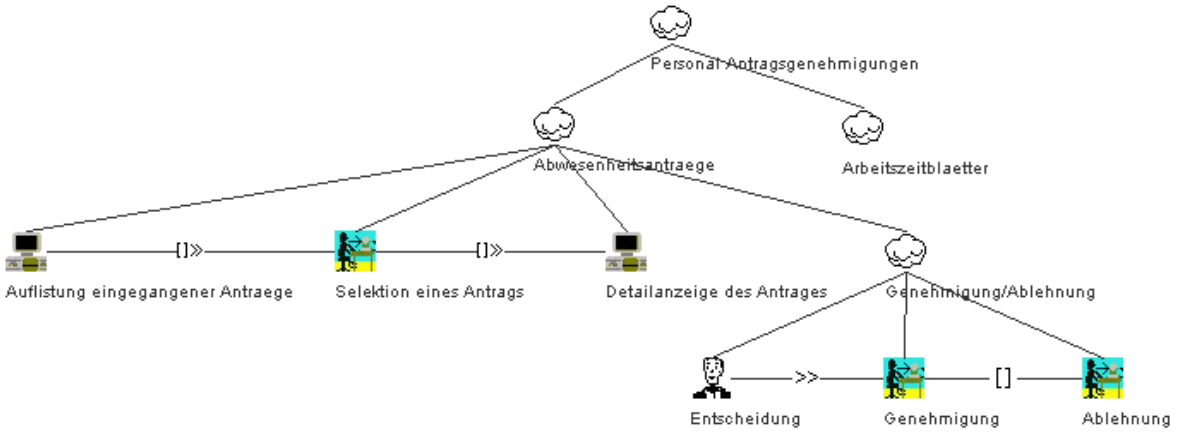

Abbildung 3-9: Ausschnitt aus dem CTT-Modell der "HR approvals" Applikation
Quelle: Eigene Darstellung

Insgesamt konnte festgestellt werden, das bestimmte Funktionalitäten in einem großen Teil der untersuchten Applikationen vorhanden sind. Tabelle 3-4 listet die identifizierten gemeinsamen Funktionalitäten inkl. einer kurzen Beschreibung auf. Die Beschreibung erfolgt hierbei durch die Nennung eines oder mehrerer Beispiele.

Tabelle 3-4: Gemeinsame Funktionalitäten der untersuchten mobilen ERP-Applikationen
Quelle: In Anlehung an (Homann et al. 2013c, 19)

Funktionalität	Beschreibung / Beispiele
Auswahl eine Business Objekt Typs	Beispielsweise wird in der Applikation einer der folgenden Menüpunkte gewählt: Kunden, Produkte oder Lieferanten.
Auflistung aller Business Objekte	Beispielsweise werden alle Lieferanten oder alle Mitarbeiter eines Unternehmens aufgelistet.
Sortierung einer Liste von Business Objekten nach einem bestimmten Attribut	Beispielsweise werden alle Mitarbeiter nach ihrem Nachnamen oder alle Produkte nach ihrem Preis sortiert aufgelistet.
Gruppierung einer Liste von Business Objekten nach einem bestimmten Attribut	Beispielsweise werden alle Produkte nach ihrer Produktgruppe oder ihrer Gewichtsklasse sortiert aufgelistet.
Suche nach einem Business Objekt über ein bestimmtes Attribut	Beispielsweise wird ein Mitarbeiter nach seinem Nachnamen oder seiner Personalnummer gesucht.
Filterung einer Liste von Business Objekten nach einem Attribut	Beispielsweise werden nur Mitarbeiter einer ausgewählten Abteilung aufgelistet oder nur Kunden aus einer ausgewählten Region.
Anzeige der Attribute und zugehörigen Werte eines ausgewählten Business Objekts	Beispielsweise wird eine bestimmte Produktinstanz ausgewählt. Daraufhin werden z.B. u.a. der Produktname, die Produktkategorie, seine Farbe, sein Gewicht und sein Preis angezeigt.
Navigation zu einem verknüpften Business Objekt eines anderen Business Objekt Typen	Beispielsweise kann von einer ausgewählten Kunden-Instanz zur Anzeige der Aufträge des Kunden navigiert werden.
Erzeugung eines neuen Business Objekts	Beispielsweise wird ein neuer Kundenauftrag angelegt oder eine neue Schadensmeldung erzeugt.
Änderung eines existierenden Business Objekts	Beispielsweise wird die Telefonnummer oder die Büronummer eines Mitarbeiters geändert.
Löschung eines existierenden Business Objekts	Beispielsweise verlässt ein Mitarbeiter das Unternehmens und wird demzufolge aus dem Mitarbeiterverzeichnis gelöscht.
Hinzufügen eines ausgewählten Business Objekts zu einer Favoritenliste	Beispielsweise wird ein häufig aufgerufenes Mitarbeiterprofil in die Favoritenliste aufgenommen, damit es zukünftig schneller zugreifbar ist.

C10: Nutzung ähnliche Interaktionsmuster

Das Untersuchungsergebnis hat gezeigt, dass die identifizierten gemeinsamen Funktionalitäten oftmals in einer ähnlichen Bearbeitungsreihenfolge ausgeführt werden. Diese ähnlichen Bearbeitungsreihenfolgen werden im Folgenden als *Interaktionsmuster* bezeichnet.

Das bestimmte Interaktionsmuster applikationsübergreifend vorzufinden sind, ist kein Zufall. SAP hat aktuell für die mobilen Betriebssysteme Android (SAP 2011c), Blackberry OS (SAP 2011a) und iOS (SAP 2011b) Gestaltungsrichtlinien erstellt. Alle angeführten Gestaltungsrichtlinien orientieren sich am sogenannten „VIA Work Zooming Model“. VIA steht hierbei für View, Inspect und Act und wird im Folgenden als VIA-Interaktionsmuster bezeichnet. Es besagt, dass sich der Anwender bei der Nutzung der Applikation zunächst einen Überblick über seine angestrebte Aktion verschaffen möchte (View). Anschließend macht er sich mit den Details der angestrebten Aktion vertraut (Inspect). In einem letzten Schritt führt er seine angestrebte Aktion durch (Act). Gemäß der angeführten SAP-Gestaltungsrichtlinien wird eine solche Bearbeitungsreihenfolge entweder aktiv oder passiv vom Benutzer initiiert (vgl. z.B.SAP 2011b, 4). In der aktiven Variante möchte der Anwender eine bestimmte Aktion durchführen. Beispielsweise möchte sich ein Vertriebsmitarbeiter über die Verfügbarkeit eines bestimmten Produktes informieren oder einen Kundenauftrag anlegen. In der passiven Variante wird der Anwender über eine bestimmte Situation informiert. Beispielsweise erreicht den Vorgesetzten eine Benachrichtigung über einen vorliegenden Dienstreiseantrag eines Mitarbeiters, welcher eine Genehmigung benötigt.

Das übergreifende VIA-Interaktionsmuster kann auch verwendet werden, um die unter Charakteristik C9 identifizierten gemeinsamen Funktionalitäten zu strukturieren. Prozesszentrierte Applikationen bieten i.d.R. im Einstiegsbildschirm eine Auswahl des gewünschten BOT an. Abbildung 3-7 illustriert beispielsweise die Auswahl zwischen den u.a. angebotenen BOTs Kunden, Kontakte und Aktivitäten im Einstiegsbildschirm der Applikation „CRM Sales“. Nach Auswahl eines BOTs oder bei Produktivitäts-Applikationen, welche oftmals nur einen BOT behandeln, erfolgt in den meisten Fällen eine Auflistung der zugehörigen BOs des BOTs. Diese werden teilweise bereits nach einem bestimmten Attribut sortiert aufgelistet oder gemäß ihrer Zugehörigkeit zu einer bestimmten Gruppierung. Zum Teil werden auch mehrere Sortierkriterien oder Gruppierungsoptionen angeboten, zwischen denen nach Bedarf gewechselt werden kann.

In der Übersichtsliste können teilweise auch neue BOs eines BOT erzeugt werden oder existierende BOs gelöscht werden. Zudem ist oftmals die Auswahl eines einzelnen BO möglich. Die Auswahl führt i.d.R. zu einer detaillierteren Ansicht auf das gewählte BO. So werden beispielsweise zusätzliche Attribute und deren Werte angezeigt.

Insgesamt zeigt sich, dass bei allen untersuchten Applikationen aus der Perspektive des BOs zunächst eine Selektion erfolgt, anschließend das ausgewählte BO präsentiert wird und schließlich auch in einigen Applikationen manipuliert werden kann. Aufgrund der unter Charakteristik C8 identifizierten Fokussierung auf Business Objekte wird im Folgenden die Bearbeitungsreihenfolge „Selektion → Präsentation → Manipulation“ als strukturierendes Interaktionsmuster dem von SAP vorgestellten VIA-Interaktionsmusters vorgezogen. Inhaltlich sind beide Interaktionsmuster identisch. Das VIA-Interaktionsmuster beschreibt die Interaktion aus der Perspektive des Anwenders; das Interaktionsmuster „Selektion → Präsentation → Manipulation“ aus der Perspektive des BOT bzw. des BO. Abbildung 3-10 ordnet die unter Charakteristik C9 identifizierten gemeinsamen Funktionalitäten den jeweiligen Bearbeitungsstufen des Interaktionsmusters zu.

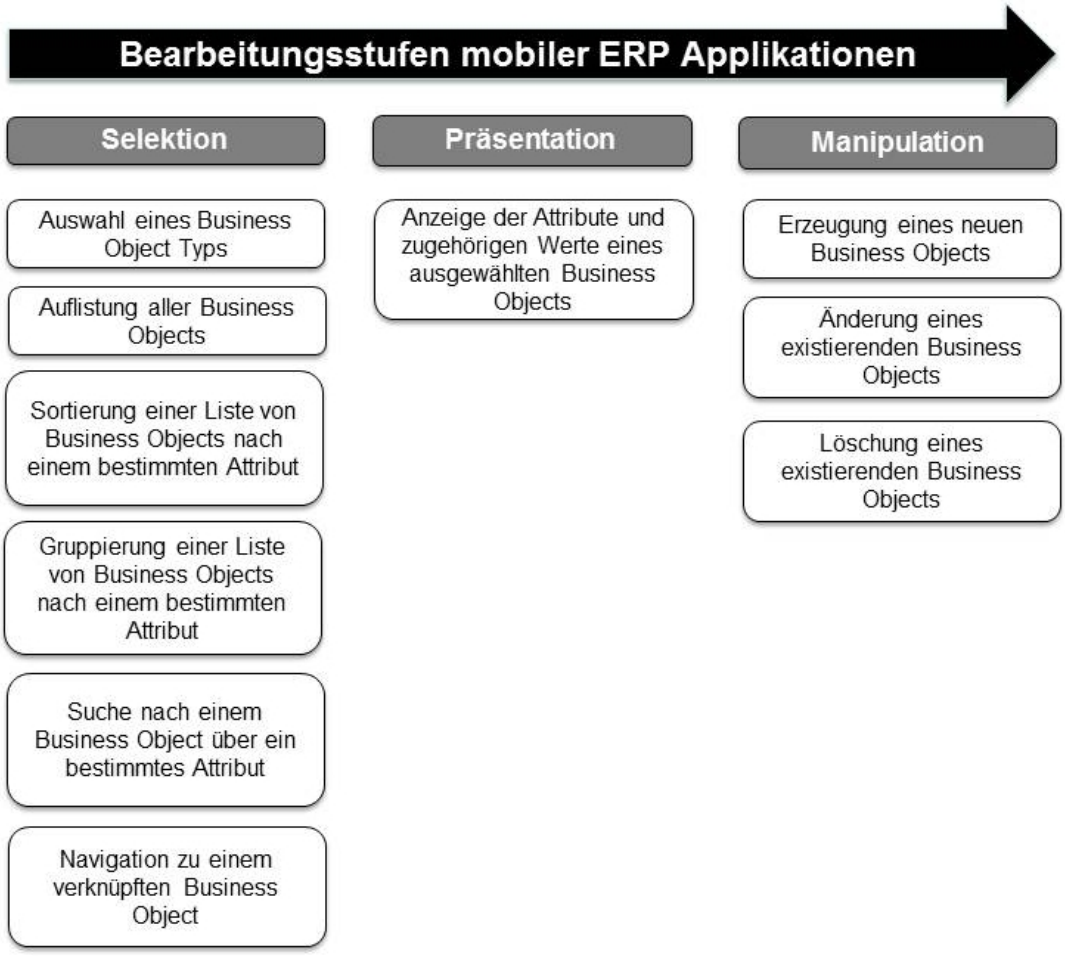

Abbildung 3-10: Interaktionsmuster der untersuchten mobilen ERP-Applikationen

Quelle: In Anlehnung an (Homann et al. 2013c, 21) und (SAP 2011b, 4)

Das Interaktionsmuster „Selektion → Präsentation → Manipulation“ ist auch in traditionellen SAP-ERP-Benutzungsschnittstellen, beispielsweise in der SAP GUI, vorzufinden. In der SAP GUI ist das Navigationsmenü, das sogenannte „Easy Access Menü“, für das jeweiligen Business Objekt ebenfalls i.d.R. in drei Transaktionen gegliedert: eine Transaktion zum Erzeugen eines BO, eine Transaktion zum Ändern oder zum Löschen eines existierenden BO und eine für das Anzeigen eines existierenden BO. Daher kann von einem Plattform- und geräteübergreifenden Interaktionsmuster für ERP-Applikationen ausgegangen werden. Abbildung 3-11 illustriert einen beispielhaften Ausschnitt aus dem „Easy Access“ Menü der SAP GUI für Windows. An dieser Bildschirmaufnahme ist erkennbar, dass für die drei BOTs „Handelswaren“, „Nicht-Lagermaterialen“ und „Dienstleistungen“ jeweils eine Transaktion zum Anlegen, zum Ändern oder zum Anzeigen von BOs angeboten werden.

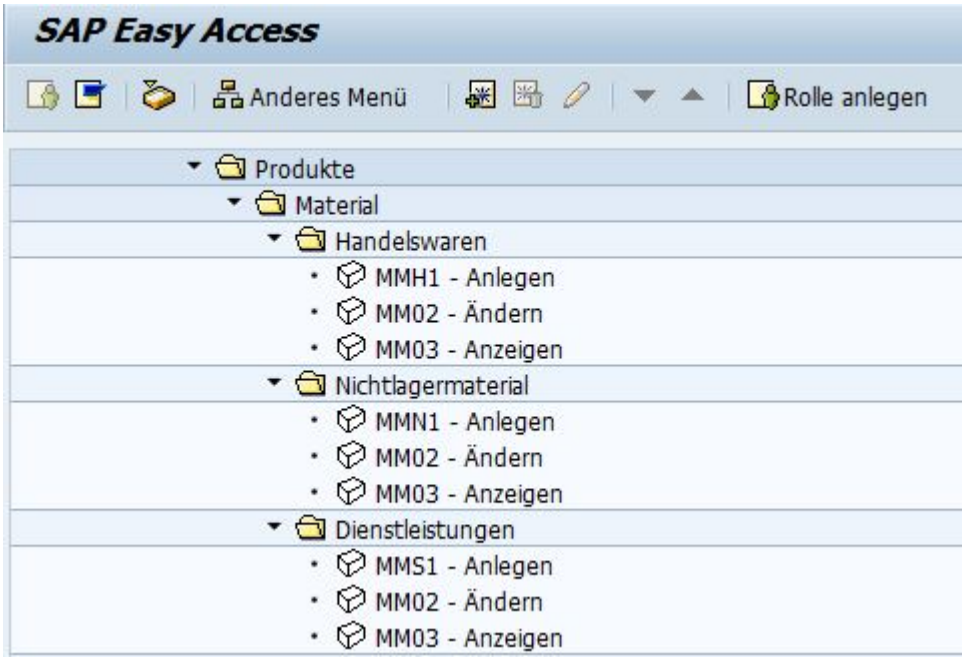

Abbildung 3-11: Ausschnitt aus dem SAP Easy Access Menü
Quelle: Eigene Darstellung

C11: Nutzung ähnlicher Benutzungsschnittstellen

Die Untersuchung der Benutzungsschnittstellen der mobilen ERP-Applikationen hat gezeigt, dass die identifizierten gemeinsamen Funktionalitäten (Charakteristik C9) i.d.R. auch durch ähnliche Kombinationen aus Anzeige- und Bedienelementen umgesetzt werden. Im Folgenden wird beschriebenen, welche Anzeige- und Bedienelemente für die einzelnen Funktionalitäten verwendet wurden.

Abbildung 3-12 illustriert die üblicherweise verwendeten Anzeige- und Bedienelemente am Beispiel der mobilen „Business One“ Applikation. Die linke Bildschirmmaske zeigt den Einstiegsbildschirm der Applikation, in welchem die fokussierten BOTs aufgelistet und zur Auswahl angeboten werden. Hierzu wird eine listenförmige Darstellung durch eine einspaltige Tabelle repräsentiert. Die verfügbaren BOs werden jeweils durch eine Zelle in der Tabelle repräsentiert. Die Möglichkeit zur Selektion einer Zelle wird durch eine spitze Klammer „>“ gekennzeichnet, welche auch als „disclosure indicator“ bezeichnet wird (SAP 2011b, 12; Apple 2013a, 170).

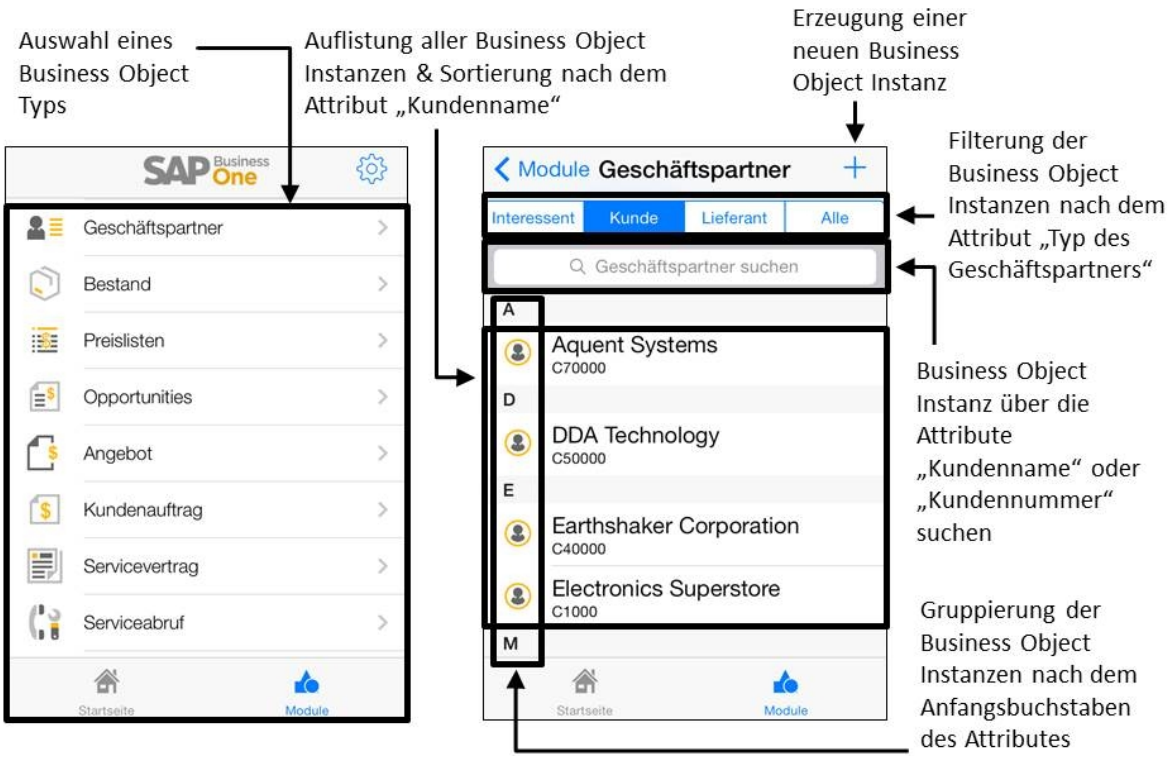

Abbildung 3-12: Ausschnitt gängiger Anzeige- und Bedienelemente mobiler ERP-Applikationen
Quelle: Eigene Darstellung

Die rechte Bildschirmaufnahme zeigt eine Bildschirmaufnahme nach Auswahl des BOTs „Geschäftspartner“ in der mobilen „Business One“ Applikation. In diesem Fall werden entweder alle BOs aufgelistet oder es erfolgt eine Filterung nach dem jeweiligen Typ des Geschäftspartners, z.B. „Interessent“, „Kunde“ oder „Lieferant“. Zur Auflistung der Instanzen wird wiederrum eine einspaltige Tabelle verwendet. In den einzelnen Zellen der Tabelle stehen die für die Identifikation eines bestimmten BOs wichtigen Attribute. Im Falle des Geschäftspartners sind dies der Name sowie die Identifikationsnummer des Geschäftspartners. Die Filterung nach den verfügbaren Geschäftspartner-Typen erfolgt durch eine Gruppierung von Schaltflächen oberhalb der Tabelle. Im illustrierten Bespiel kann zwischen den angebotenen Geschäftspartner-Typen „Interessent“, „Kunde“ und „Lieferant“ sowie einer Auflistung aller gespeicherten Geschäftspartner-Instanzen gewechselt werden. Die Suche einer bestimmten Instanz eines Geschäftspartners wird durch eine sogenannte Suchleiste bzw. Search Bar implementiert (SAP 2011b, 42 f.; Apple 2013a, 148 f.). Dabei handelt es sich um ein Textfeld, in welches der Wert eines Attributes eingegeben werden kann. Die Suche nach zugehörigen Instanzen erfolgt dabei unmittelbar während des Tippvorganges, d.h. ein manuelles Starten des Suchvorganges ist nicht notwendig. Das Erzeugen eines neuen BO erfolgt durch eine Schaltfläche in der oberen, rechten Ecke des Bildschirms, welche mit einem Plus-Symbol „+“ gekennzeichnet ist. Anschließend öffnet sich eine neue Bildschirmmaske, welche zur Eingabe der Werte für die jeweiligen Attribute des BO auffordert.

Abbildung 3-13 illustriert einen weiteren Ausschnitt der üblicherweise verwendeten Anzeige- und Bedienelemente.

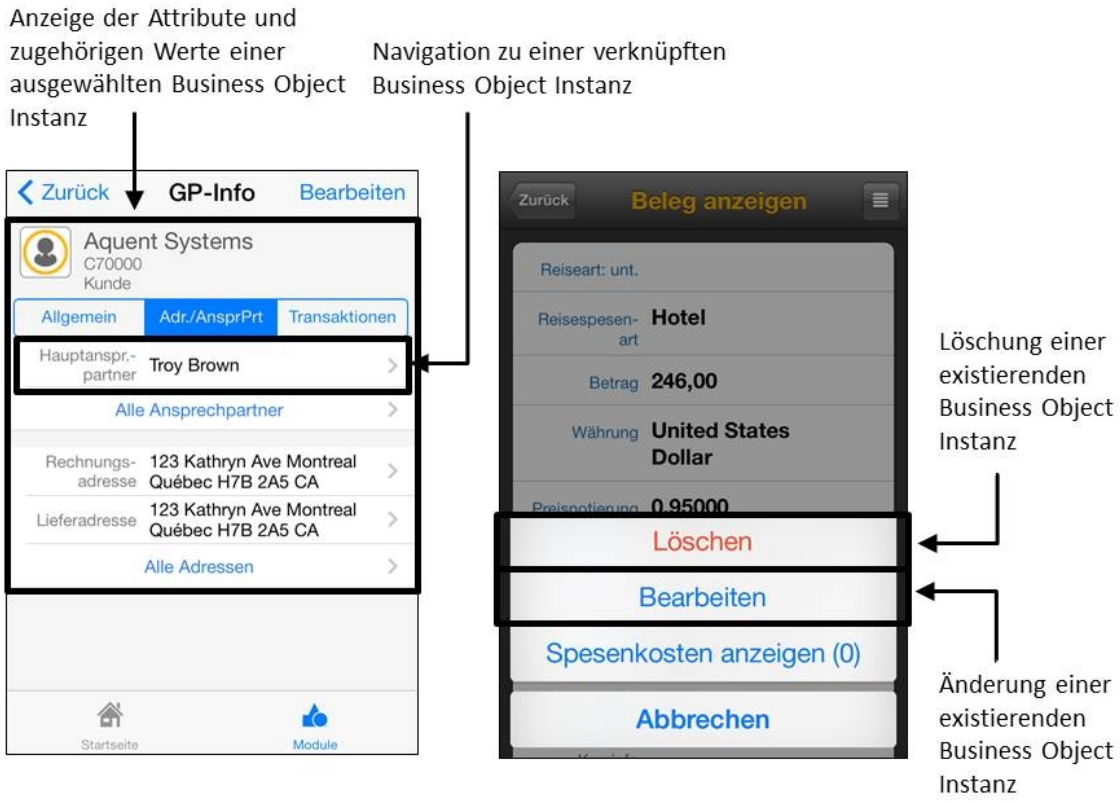

Abbildung 3-13: Ausschnitt gängiger Anzeige- und Bedienelemente mobiler ERP-Applikationen
Quelle: Eigene Darstellung

Beide Bildschirmaufnahmen veranschaulichen die Anzeige bzw. Bearbeitung der Attributwerte eines ausgewählten BO. Die linke Bildschirmaufnahme zeigt die mobile Applikation „Business One“ mit einer angezeigten Kunden-Instanz. Die rechte Bildschirmaufnahme zeigt die Applikation „Travel Expense Report“ am Beispiel eines selektierten Spesenbeleges. Es hat sich gezeigt, dass in den meisten Fällen wiederrum eine einspaltige Tabelle zur Anzeige der Attribute zum Einsatz kommt. Die enthalten Zellen beinhalten jeweils den Namen und den zugehörigen Wert des Attributes. Im Falle von verknüpften BOs werden wiederraum selektierbare Zellen verwendet, welche über den bereits genannten „disclosure indicator“ gekennzeichnet werden. In der linken Bildschirmaufnahme ist dies beispielsweise an den Zellen für den Hauptansprechpartner, die Rechnungsadresse oder die Lieferadresse zu sehen. Das Bearbeiten oder Löschen eines existierenden BO wird teilweise durch einfache Schaltflächen mit der Beschriftung „Löschen“ oder „Bearbeiten“ am unteren Bildschirmrand umgesetzt. Eine alternative Umsetzungsvariante ist die Verwendung eines „Action Sheets“ (SAP 2011b, 48 f.; Apple 2013a, 194 f.) mit zugehörigen Optionen zum „Löschen“ oder „Bearbeiten“, wie dies in der rechten Bildschirmaufnahme zu sehen ist. In beiden Fällen wird der Anwender zu einer Bildschirmmaske geführt, bei welcher die Attributwerte editierbar sind.

C12: Eingeschränkte Parametrisierungsmöglichkeiten

Bei den untersuchten mobilen ERP-Applikationen hat sich gezeigt, dass diese nur eingeschränkte Möglichkeiten zur Parametrisierung anbieten. In den meisten Fällen beschränken sich die Möglichkeiten zur Parametrisierung auf die Verbindungsparameter zum MEAP-Server, beispielsweise dessen IP-Adresse, zugehörigen Anmeldedaten oder maximale Zeitintervalle für die Verbindung zwischen der Applikation und dem MEAP-Server. Oftmals wird zusätzlich eine Möglichkeit angeboten, eine Protokollierungsfunktion (engl. trace) zu aktivieren. Die protokollierten Informationen erlauben im Fehlerfall eine bessere Analyse der Fehlerursache.

Die einzige unter den untersuchten Applikationen, welche zusätzliche, applikationsbezogene Parametrisierungsmöglichkeiten besitzt, ist die „Material availability“ Applikation. Diese erlaubt u.a. die Konfiguration der „Verkaufsorganisation“, des „Vertriebsweges“ und der „Sparte“- Mit Hilfe dieser Parameter kann die angezeigte Materialliste gemäß der Präferenzen des Anwenders eingeschränkt werden. Abbildung 3-14 illustriert die beschriebenen Parametrisierungsmöglichkeiten am Beispiel der beiden Applikationen „CRM Sales“ und „Materials availability“.

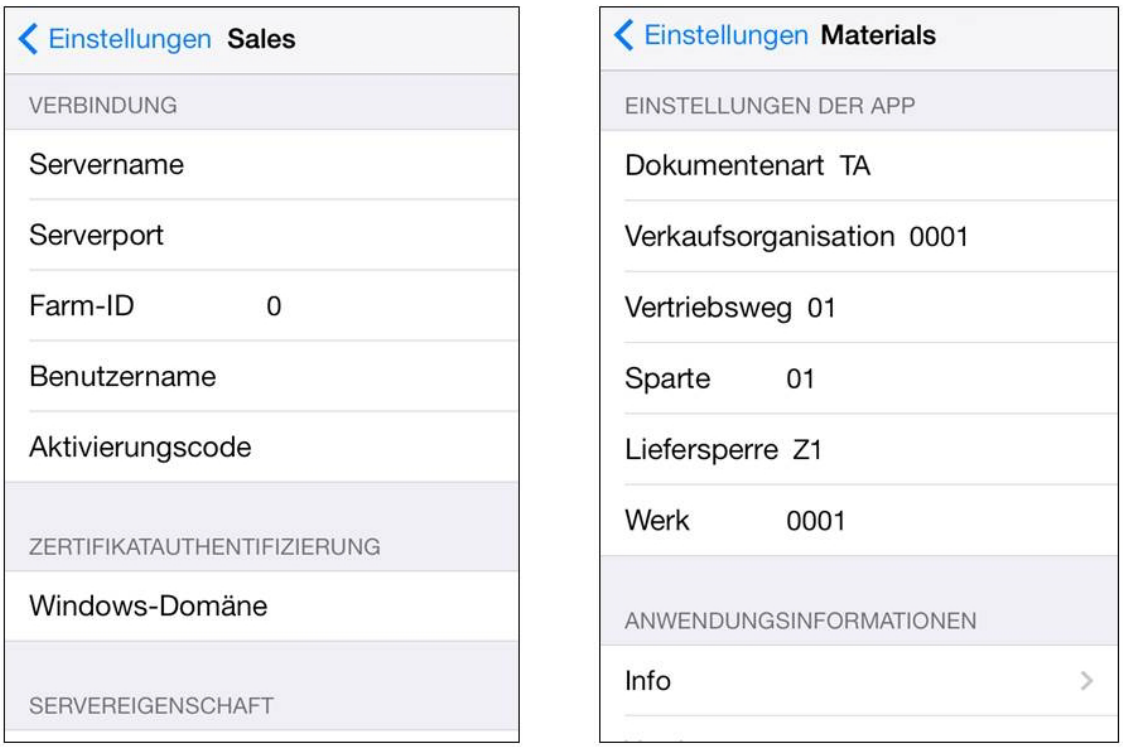

Abbildung 3-14: Parametrisierungsmöglichkeiten der untersuchten mobilen ERP-Applikationen
Quelle: Eigene Darstellung

Die linke Bildschirmaufnahme illustriert einen Ausschnitt aus den Parametrisierungsmöglichkeiten der „CRM Sales“ Applikation. In diesem Fall werden hauptsächlich die Verbindungsparameter zum verwendeten MEAP-Server konfiguriert, wobei der SAP Mobile Platform Server als MEAP-Server verwendet wird.

Die rechte Bildschirmaufnahme illustriert die Parametrisierungsmöglichkeiten der „Materials availability“ Applikation. In diesem Fall könnten unterschiedliche Informationen zum Material angegeben werden, um die angezeigte Materialliste einzuschränken. Beispielhafte Parameter sind die „Verkaufsorganisation“, der „Vertriebsweg“, die „Sparte“ und das „Werk“.

3.3.2.2 Unterschiede der untersuchten mobilen ERP-Applikationen

Es hat sich gezeigt, dass nur ein kleinerer Teil der untersuchten Applikationen von den eingebauten Sensoren des iPhones Gebrauch macht. In diesen Fällen wird oftmals die Kamera genutzt um Barcodes zu scannen oder Spesenbelege zu fotografieren. Ein Beispiel ist die „Retail Execution“ Applikation, welche den Barcode eines Materials über die Kamera scannen kann, um anschließend die Details des Materials anzuzeigen. Ein weiteres Beispiel ist die „Travel Receipt Capture“ Applikation, bei welcher die Reisespesen fotografiert und die zugehörige Sprachaufzeichnungen zur Beschreibung des Spesenbeleges archiviert werden können.

Bei den identifizierten ähnlichen Benutzungsschnittstellen wurden Anzeige- und Bedienelemente aufgeführt, die in nahezu allen untersuchten Applikationen vorzufinden sind. Es gibt jedoch auch Applikationen mit spezifischeren Anzeige- und Bedienelementen. Ein Beispiel ist die Applikation „Business One“. Diese hat u.a. auch die Möglichkeit auf verschiedene Berichte oder Dashboards zuzugreifen und besitzt somit auch Bestandteile einer analytischen Applikation (siehe Charakteristik C3). Ein anderes Beispiel ist die Applikation „Timesheet“, welche eine Kalenderansicht für die Navigation innerhalb der Applikation nutzt. Abbildung 3-15 illustriert in der linken Bildschirmaufnahme einen visualisierten Bericht in der Applikation „Business One“ und das Kalender-Bedienelement der Applikation „Timesheet“ in der rechten Bildschirmaufnahme.

Abbildung 3-15: Spezifischere Bedienelemente mobiler ERP-Applikationen
Quelle: Eigene Darstellung

3.4 Fazit und Diskussion

Bei der Identifizierung von gemeinsamen Charakteristiken wurde sowohl eine Literaturanalyse, als auch eine Analyse existierender mobiler ERP-Applikationen durchgeführt. Während durch die Literaturanalyse abstraktere Charakteristiken identifiziert wurden, konnten durch die die Analyse existierender Applikationen weitere Charakteristiken aus der Umsetzungsebene ergänzt werden. In stellt die identifizierten Charakteristiken zusammenfassend dar. Tabelle 3-5 stellt die identifizierten Charakteristiken nochmals zusammenfassend dar.

Tabelle 3-5: Charakteristiken mobiler ERP-Applikationen
Quelle: Eigene Darstellung

Charakteristik	Beschreibung
C1	Daten und Funktionalitäten stammen aus dem ERP-System
C2	Stellen die Präsentationsschicht in einem mobilen ERP-Applikationsszenario dar
C3	Besitzen unterschiedliche Applikationsschwerpunkte
C4	Nutzen unterschiedliche Implementierungsvarianten
C5	Unterschiedliche Verfahren zur Datenspeicherung werden unterstützt
C6	ERP-Daten und -Funktionalitäten werden reduziert
C7	Nutzung einer mobilen Unternehmensplattform
C8	Fokussierung auf ausgewählte Business Objekt Typen
C9	Bereitstellung ähnliche Funktionalitäten
C10	Nutzung ähnliche Interaktionsmuster
C11	Nutzung ähnlicher Benutzungsschnittstellen
C12	Eingeschränkte Parametrisierungsmöglichkeiten

Die Ergebnisse der Literaturanalyse haben gezeigt, dass mobile ERP-Applikationen ihre Daten und Funktionalitäten aus einem ERP-System beziehen und sich demzufolge auf die Benutzungsschnittstelle fokussieren. Die Daten und Funktionalitäten des ERP-Systems sind hierbei über entsprechende Programmierschnittstellen des ERP-Systems zugreifbar. Der Aufruf dieser Programmierschnittstellen erfolgt derzeit häufig nicht direkt durch die mobilen ERP-Applikationen, sondern durch eine mobile Unternehmensplattform. Eine solche Plattform steuert die Kommunikation zwischen den mobilen ERP-Applikationen und dem zugehörigen ERP-System. Durch diesen zentralen Kommunikationskanal wird die Umsetzung von Sicherheitsmechanismen oder die Überwachung der Kommunikationsverbindungen erleichtert. Derzeit werden unterschiedliche Technologien und zugehörige Entwicklungswerkzeuge zur Implementierung mobiler ERP-Applikationen eingesetzt. Diese Technologievielfalt in Kombination mit der derzeit hohen Dynamik im Markt für mobile Endgeräte erschwert derzeit die Entscheidung für eine bestimmte Implementierungsstrategie.

Die Analyse existierender mobiler ERP-Applikationen hat gezeigt, dass wiederkehrende Funktionalitäten in mobilen ERP-Applikationen identifiziert werden können. Diese Funktionalitäten können aus der Perspektive der Business Objekte in Selektions-, Präsentations- und Manipulationsfunktionalitäten eingeteilt werden. Die Untersuchung der zugehörigen Anzeige- und Bedienelemente der Benutzungsschnittstelle hat gezeigt, dass die identifizierten Funktionalitäten oftmals durch die gleichen Anzeige- und Bedienelemente umgesetzt werden. Dies gilt insbesondere für Produktivitäts- und Prozesszentrierte-Applikationen. Aufbauend auf dieser Erkenntnis stellt sich die Frage, ob wiederverwendbare Bausteine für die identifizierten Funktionalitäten entwickelt und bereitgestellt werden können, welche zur Implementierung neuer mobiler ERP-Applikationen verwendet werden können.

4 Empirische Untersuchung zur Entwicklung mobiler Unternehmensapplikationen in der Praxis

In diesem Kapitel soll die aktuelle Situation bei der Entwicklung mobiler Unternehmensapplikationen in der Praxis eruiert werden. Hierzu werden mit Hilfe semi-strukturierte Interviews zwei voneinander unabhängige Befragungsrunden mit Experten aus der Unternehmenspraxis durchgeführt. Bei der ersten Befragungsrunde liegt der Schwerpunkt auf der Ermittlung des Ist-Zustandes und der Anwendung modellbasierter Entwicklungsansätze bei der Entwicklung mobiler Unternehmensapplikationen. Schwerpunkt der zweiten Befragungsrunde ist hingegen die Einschätzung der Potentiale von Endbenutzer-Entwicklung im Bereich mobiler Unternehmensapplikationen. Übergeordnetes Ziel beider Befragungsrunden ist die Bekräftigung der Problemrelevanz der fokussierten Forschungslücke aus Perspektive der Unternehmenspraxis sowie die Ermittlung von Anforderungen für das angestrebte Endbenutzer-Entwicklungswerkzeug. Bei beiden Befragungsrunden wurde der Fokus der Befragung auf mobile Unternehmensapplikationen gelegt und nicht auf deren Teilmenge „mobile ERP-Applikationen“. Hierdurch konnte eine größere Anzahl von Interviewpartnern gefunden werden. Zudem besteht die Annahme, dass viele Problemstellungen bei der Entwicklung mobiler Unternehmensapplikationen im Allgemeinen auch für mobile ERP-Applikationen im Speziellen gelten.

Im Folgenden werden zunächst wesentliche Grundlagen des zugrundeliegenden Forschungsgebietes, der empirischen Sozialforschung, behandelt. In diesem Zusammenhang werden semi-strukturierte Interviews als gewählte Forschungsmethodik begründet und vergleichsweise detaillierter vorgestellt. Anschließend werden die Hintergründe, das Vorgehen und die zentralen Ergebnisse der beiden Befragungen vorgestellt.

4.1 Vorgehensweise in der Arbeit

Semi-strukturierte Experteninterviews stellen eine Forschungsmethode aus der empirischen Sozialforschung dar. Um die Entscheidung für semi-strukturierte Experteninterviews nachvollziehen zu können, wird zunächst ein kurzer Überblick über Methoden der empirischen Sozialforschung gegeben und die Entscheidung für die gewählte Forschungsmethode begründet. Anschließend werden unterschiedliche Ausprägungsarten von Befragungen erläutert und schließlich die eigene Vorgehensweise beschrieben.

4.1.1 Grundlagen empirischer Sozialforschung

Die *empirischen Sozialforschung* beschäftigt sich mit der „systematischen Erfassung und Deutung sozialer Erscheinungen“ (Atteslander 2010, 4). Der Begriff „empirisch“ bedeutet „erfahrungsgemäß“ (Atteslander 2010, 3) und weist darauf hin, dass „theoretisch formulierte Annahmen an spezifische Wirklichkeiten überprüft werden“ (Atteslander 2010, 4). „Systematisch“ verdeutlicht, dass dieser Vorgang regelbasiert erfolgen sollte (Atteslander 2010, 4 f.). Ziel des systematischen Vorgehens ist die *intersubjektive Nachprüfbarkeit* bzw. *Objektivität*

der Erkenntnisgewinne (Bortz/Döring 2006, 32). In Bezug auf das Vorgehen wird prinzipiell zwischen einem qualitativen- und einem quantitativen Forschungsansatz unterschieden (Hug 2001, 22; Atteslander 2010, 12 ff.).

Die *quantitative Forschung* orientiert sich am naturwissenschaftlichen Forschungsverständnis und basiert auf zähl- und messbare Faktoren (Hug 2001, 22; Bortz/Döring 2006, 139). Es wird eine Quantifizierung bzw. Messung von Ausschnitten der Beobachtungsrealität durchführt und die daraus resultierenden Messwerte anschließend statistisch verarbeitet (Bortz/Döring 2006, 138 ff.; Atteslander 2010, 245 ff.). Das Methodenspektrum umfasst standardisierte Befragungstechniken, schematisierte Beobachtungsformen, inhaltsanalytische und statistische Verfahren, experimentelle Vorgehensweise und Tests sowie Skalierungsverfahren (Hug 2001, 22). Quantitative Forschung hat oftmals ein *explanatives* bzw. *erklärendes* Erkenntnisziel (Bortz/Döring 2006, 299), d.h. es ist Ziel Hypothesen oder Theorien zu überprüfen (Gläser/Laudel 2009, 26).

Bei der *qualitativen Forschung* wird die Erfahrungswirklichkeit verbalisiert und anschließend interpretativ ausgewertet (Bortz/Döring 2006, 296). Im Gegensatz zu quantitativen Forschungsmethoden verzichten qualitative Forschungsmethoden auf Zahlenmaterial. Stattdessen werden qualitative Materialien wie bspw. Beobachtungsprotokolle, Interviewtexte, Fotografien, Zeichnungen oder Filme analysiert (Bortz/Döring 2006, 297). Das Methodenspektrum umfasst verschiedene Interviewformen, Gruppendiskussionsverfahren, verschiedene Beobachtungsvarianten, ethnographische Vorgehensweisen, inhaltsanalytische Verfahren und qualitative Experimente (Hug 2001, 22). Qualitative Sozialforschung orientiert sich am Prinzip der Offenheit, d.h. der Forscher soll offen für neue und unerwartete Erkenntnisse sein (Atteslander 2010, 77; Lamnek 2010, 19 f.). Aus diesem Grund hat die qualitative Forschung in Bezug auf ihr Erkenntnisziel einen *explorativen* bzw. *erkundenden* Charakter (Bortz/Döring 2006, 299), d.h. sie zielt darauf ab erste Einblicke in Bezug auf einen Untersuchungsgegenstand zu bekommen, um anschließend neue Hypothesen oder Theorien zu bilden (Bortz/Döring 2006, 356 ff.; Gläser/Laudel 2009, 26).

Bei der Untersuchung der Praxisrelevanz und Anforderungsermittlung an das umzusetzende Artefakt in der vorliegenden Arbeit sollen keine konkreten Hypothesen getestet werden, sondern ein möglichst gutes Verständnis über aktuelle Problemstellungen und Praxisanforderungen aufgebaut werden. Da es sich hierbei um eine explorative Untersuchung handelt, erscheint ein qualitativer Forschungsansatz als geeignet. Für die Untersuchung der vorliegenden explorativen Fragenstellungen sind eine offene Haltung des Forschers und eine geeignete Forschungsmethode, welche diese offene Haltung unterstützt, erforderlich. In der qualitativen Forschung sind prinzipiell vier Methodengruppen zu unterscheiden: Beobachtungen, Experimente, Befragungen und Inhaltsanalysen (Atteslander 2010, 54).

Unter einer *Beobachtung* wird das „systematische Erfassen, Festhalten und Deuten sinnlich wahrnehmbaren Verhaltens zum Zeitpunkt seines Geschehens“ (Atteslander 2010, 73) verstanden. Bei der Beobachtung versucht der Forscher in einem nicht-kommunikativen Prozess mit Hilfe sämtlicher Wahrnehmensmöglichkeiten Erfahrungen zu sammeln (Bortz/Döring 2006, 262). Dabei ist die wissenschaftliche Beobachtung im Vergleich zur Alltagsbeobach-

tung zielgerichteter und methodisch fundiert (Bortz/Döring 2006, 262). Dabei werden unterschiedliche Instrumente eingesetzt, um die Selbstreflektiertheit, Systematik und Kontrolliertheit der Beobachtung zu gewährleisten (Bortz/Döring 2006, 262).

Ein *Experiment* kann definiert werden als „Manipulation einer hypothetisch vermutetet Ursache oder Bedingung für eine Wirkung oder einen Effekt bei gleichzeitiger Kontrolle möglicher Störfaktoren bzw. experimenteller Randbedingungen. Das Ziel ist der exakte Beweis für eine kausale Beziehung“ (Scholl 2009, 87). In Bezug auf die Rahmenbedingungen können zwei Arten von Experimenten unterscheiden: Feld- und Laboratoriumsexperimente (Atteslander 2010, 181). Bei Laboratoriums- bzw. Laborexperimenten wird eine Experimentalgruppe unter bewusst konstruierten, „künstlichen“ Rahmenbedingungen untersucht (Atteslander 2010, 181; Lamnek 2010, 513). Hierdurch soll sichergestellt werden, dass lediglich die untersuchten Faktoren auf das Ergebnis einwirken und potenzielle Störfaktoren ausgeschlossen werden. Hingegen wird der zu untersuchende Gegenstand bei einem Feldexperiment nicht aus seiner natürlichen Umgebung herausgelöst (Atteslander 2010, 181; Lamnek 2010, 513).

Die *Befragung* nutzt die Kommunikation zwischen zwei oder mehreren Personen als Grundlage zur Gewinnung von Informationen über einen Untersuchungsgegenstand (Scholl 2009, 21; Atteslander 2010, 109). Bei der Befragung werden durch Fragen (verbale Stimuli), Antworten (verbale Reaktionen) hervorgerufen (Atteslander 2010, 109). Die Antworten stellen Meinungen und Bewertungen des Befragten über erlebte und erinnerte soziale Ereignisse dar (Atteslander 2010, 109). Die Befragung ist die in der empirischen Sozialforschung am häufigsten angewendete Methode zur Datenerhebung (Bortz/Döring 2006, 236). In Bezug auf die verwendete Kommunikationsart kann zwischen einer mündlichen Befragung in Form von Interviews und einer schriftlichen Befragung in Form von Fragebögen unterscheiden werden (Bortz/Döring 2006, 236).

Die *Inhaltsanalyse* dient der interpretierenden Auswertung des bereits erhobenen Datenmaterials (Lamnek 2010, 435). Mittels Inhaltsanalysen werden hauptsächlich Texte ausgewertet; jedoch ist auch die Analyse anderer Kommunikationsinhalte wie beispielsweise von Bildern, Videoaufnahmen oder Tonaufzeichnungen möglich (Atteslander 2010, 195; Lamnek 2010, 438). Die Inhaltsanalyse dient der primären Datengewinnung. Das Ziel der Inhaltsanalyse ist es, die zentralen Themen und Bedeutungen von Texten und anderen Objekten herauszuarbeiten und hieraus auf Zusammenhänge seiner Entstehung und Verwendung zu schließen (Atteslander 2010, 196). Hierbei lassen sich wiederrum quantitative- und qualitative Ansätze unterscheiden. Ansätze aus der quantitativen Inhaltsanalyse versuchen die Inhalte mit Ordinal-, Intervall- oder Ratio-Skalen zu versehen und mit Hilfe von statistischen Verfahren auszuwerten (Mayring 2010, 18). Ansätze aus der qualitativen Inhaltsanalyse arbeiten hingegen häufig mit Nominalskalen (Gläser/Laudel 2009, 201) und fokussieren sich auf die Suche nach „[…] Kausalmechanismen, die unter bestimmten Bedingungen bestimmte Effekte hervorbringen“ (Gläser/Laudel 2009, 26). Eine weitere Unterscheidung zwischen der qualitativen und quantitativen Inhaltsanalyse besteht in der Änderbarkeit des Kategoriensystems. Ein Kategoriensystem wird üblicherweise theoriebasiert abgeleitet, um die Kommunikationsinhalte systematisch auf Basis der erstellten Kategorien analysieren zu können. Während dieses bei

der quantitativen Inhaltsanalyse „geschlossen“ ist, d.h. nicht änderbar ist, kann das Kategoriensystem in der qualitativen Inhaltsanalyse angepasst werden. Ein Anpassung kann bspw. dann vorgenommen werden, falls während der Analyse Inhalte gefunden werden, die sich mit dem erstellten Kategoriensystem nicht auswerten lassen (Gläser/Laudel 2009, 197 ff.).

Experimente bieten ein hohes Maß an Kontrolle (Atteslander 2010, 177); erscheinen jedoch aufgrund der offenen Fragestellung der vorliegenden Arbeit als ungeeignet. Eine Beobachtung von Softwareentwicklern bei der Umsetzung mobiler Applikationen wäre prinzipiell möglich. Jedoch wäre dieses Vorgehen verhältnismäßig zeitaufwändig und ließe keine Rückschlüsse auf die Gründe für gewählte Vorgehensweise und Problemstellungen zu. Zudem wäre es hierbei nicht möglich die Einschätzungen der Untersuchungspersonen zu bestimmten Fragestellungen zu ergründen. Für die Untersuchung der aktuellen Praxissituation erscheint eine Befragung als geeignete Methode, da hierbei unterschiedliche Fragestellungen adressiert werden können und flexibel auf die Gesprächspartner eingegangen werden kann. Zur Auswertung der transkribierten Befragungen erscheint eine qualitative Inhaltsanalyse zweckdienlich. Aufgrund der kleinen Stichprobe und des explorativen Charakters der vorliegenden Fragestellung, erscheint ein streng formalisiertes Vorgehen als nicht geeignet. Stattdessen sollen die erhobenen Daten thematisiert und interpretiert werden.

4.1.2 Ausprägungsarten der Befragung

Hinsichtlich der verwendeten *Kommunikationsart* kann nach Scholl (2009, 29) zwischen persönlichen, telefonischen und schriftlichen Befragungen unterschieden werden. Zusätzlich ist noch die Online-Befragung als eine Variante der schriftlichen Befragung zu nennen (Scholl 2009, 29, 53 ff.). Für die vorliegende Untersuchung erscheint die persönliche Befragung als geeignet. Dabei entsteht der Vorteil den Befragten besser motivieren und durch gezielte Rückfragen unvollständige Antworten vervollständigen zu können (Scholl 2009, 37 f.). Zudem können sowohl der Interviewer als auch der Befragte bei Unklarheiten nachfragen und so die Qualität des Interviews erhöhen (Scholl 2009, 38). Außerdem hat der Interviewer bei einem persönlichen Interview die Möglichkeit, visuelle Unterstützungen zu nutzen (Scholl 2009, 38). Dies ist für die angestrebten Befragungen sehr nützlich, da hierdurch die Möglichkeit besteht, wichtige Konzepte bspw. über Powerpoint-Folien zu erläutern, falls diese den Befragten nicht bekannt sind. Zudem können ausgewählte Entwicklungswerkzeuge kurz vorgeführt werden, um die Meinung der Experten zu diesen Werkzeugen zu erfragen.

Nachteilig bei der persönlichen Befragung ist der hohe Aufwand sowie die vergleichsweise hohen Kosten, welche beispielsweise durch eine Anreise zu einem gemeinsamen Gesprächsort entstehen (Scholl 2009, 38). Zudem besteht eine verhältnismäßig geringe Anonymität des Befragten, was als Folge zu einer Einschüchterung und ausweichenden oder unehrlichen Antworten des Befragten führen kann (Scholl 2009, 41). Aufgrund der geringen Anzahl an Befragungen werden die angeführten Nachteile akzeptiert und falls möglich persönliche Interviews durchgeführt. In Fällen, in denen ein persönliches Gespräch mit potenziellen Interviewpartnern nicht möglich ist, wird stattdessen eine telefonische Befragung durchgeführt. Aufgrund aktueller Möglichkeiten zum Teilen von Bildschirmansichten (engl. screen

sharing) über Softwareapplikationen wie bspw. TeamViewer[14], kann auch bei Telefoninterviews von Folien-Präsentationen und Werkzeug-Vorführungen gebraucht gemacht werden.

Zusätzlich können Befragungen nach ihrem *Strukturierungsgrad* unterschieden werden. Dabei wird zwischen wenig strukturierten, teilstrukturierten und stark strukturierten Befragungen unterschieden (Atteslander 2010, 133 ff.). In mündlichen Befragungen werden in der Regel wenig- und teilstrukturierte Befragungen durchgeführt, wohingegen in schriftlichen Befragungen eine starke Strukturierung verwendet wird (Bortz/Döring 2006, 308). Wenig- und teilstrukturierte Befragungen ermöglichen eine flexiblere Gestaltung der Befragung und eigenen sich daher für offene Fragestellungen (Bortz/Döring 2006, 308 f.). Bei stark strukturierten Befragungen ist hingegen der Inhalt, die Anzahl und die Reihenfolge der Fragen festgelegt (Atteslander 2010, 134 f.). Durch die dabei erreichte Standardisierung der Interviewsituation (Scholl 2009, 77 f.) werden eine größere Unabhängigkeit vom Interviewer sowie eine bessere Vergleichbarkeit der Antworten ermöglicht.

Bei *wenig strukturierten Befragungen* arbeitet der Forscher ohne Fragebogen (Atteslander 2010, 134). Dies erlaubt eine flexible Gesprächsführung, da der Forscher seine Fragen individuell an die befragte Person und deren Antworten anpassen kann (Atteslander 2010, 134). Das Gespräch folgt nicht den Fragen des Forschers, sondern die jeweils nächste Frage ergibt sich aus der aktuellen Gesprächssituation. Der Forscher hat die Aufgabe das Gespräch in Gang zu halten und die Meinung des Befragten zu erfassen, ohne seine Reaktionsmöglichkeit einzuschränken (Atteslander 2010, 134). Da die Antworten des Befragten nicht durch festgelegte Antwortkategorien eingeschränkt werden, wird von einer offenen Fragestellung gesprochen (Bortz/Döring 2006, 315).

Bei *stark strukturierten Befragungen* wird ein Fragebogen konstruiert, welcher den Inhalt, die Anzahl sowie die Reihenfolge der gestellten Fragen festlegt (Atteslander 2010, 134). Zusätzlich wird bei der Konstruktion des Fragebogens bereits über die möglichen Antwortkategorien entschieden (Scholl 2009, 77; Atteslander 2010, 135). Aufgrund des fehlenden Freiheitsgrades bei der Beantwortung wird auch von *geschlossenen Fragestellungen* gesprochen (Bortz/Döring 2006, 314; Scholl 2009, 77). Auf der einen Seite wird hierdurch eine höhere Vergleichbarkeit der Befragungen erzielt; auf der anderen Seite sind hierdurch die Gestaltungsfreiheiten des Forschers als auch des Befragten sehr begrenzt (Scholl 2009, 78; Atteslander 2010, 134). Stark strukturierte Befragungen werden in der Regel schriftlich durchführt (Bortz/Döring 2006, 252 ff.). Sie sind unabhängig vom Forscher und hierdurch in einer großen Anzahl durchführbar. Aufgrund ihrer geringen Freiheitsgrade werden sie jedoch tendenziell eher in der quantitativen Forschung eingesetzt, bspw. um Hypothesen zu prüfen (Bortz/Döring 2006, 252 ff.).

Bei *teilstrukturierten Befragungen* wird das Gespräch durch vorbereitete und vorformulierte Fragen strukturiert (Atteslander 2010, 135). Dabei ist die Reihenfolge der Fragen nicht festgelegt und der Forscher hat jederzeit die Möglichkeit interessante Themen aufzugreifen und sie weiter zu vertiefen. Als Gedächtnisstütze für den Interviewer wird i.d.R. ein soge-

[14] http://www.teamviewer.com/de, zugegriffen am 30.12.2013

nannter *Leitfaden* eingesetzt, welcher die vorbereiteten Fragen enthält (Scholl 2009, 68; Atteslander 2010, 135). Hierdurch behält der Interviewer eine Gesprächsorientierung und reduziert so das Risiko wichtige Aspekte zu vergessen. Dennoch hat der Interviewer die Möglichkeit spontan auf die aktuelle Interviewsituation reagieren und beispielsweise klärend nachfragen oder tiefergehende Verständnisfragen stellen (Bortz/Döring 2006, 314; Scholl 2009, 68). Aufgrund der Nutzung eines Leitfadens werden teilstrukturierte Befragungen oftmals als *Leitfadeninterview* bezeichnet (Scholl 2009, 68). Leitfadeninterviews stellen die gängigste Form qualitativer Befragungen dar (Bortz/Döring 2006, 314).

In Bezug auf die befragte Personengruppe stellen die *leitfadengestützten Experteninterviews* eine spezielle Form von Leitfadeninterviews dar (Scholl 2009, 68 f.). Der Expertenstatus einer Person ergibt sich nach Scholl (2009, 69) aus seiner Position oder Funktion, welche die Person innerhalb einer Organisation begleitet. Nach Meuser und Nagel (1991, 442 ff., 466) zeichnet *Experten* aus, dass sie für eine bestimmte Aufgabe verantwortlich sind und hierdurch einen privilegierten Zugang zu den gewünschten Informationen besitzen.

4.1.3 Gewählte Vorgehensweise

Für die Beantwortung der vorliegenden Fragstellungen ist ein tiefergehendes Verständnis und Erfahrung im Bereich der Entwicklung mobiler Unternehmensapplikationen erforderlich. Daher sollen Personen mit mehrjähriger Erfahrung in der Entwicklung mobiler Unternehmensapplikationen identifiziert und befragt werden. Personen mit dieser Charakteristik werden in Bezug auf die Fragestellungen der Interviews als *Experten* bezeichnet. Aufgrund der offenen Fragestellung und des Expertenwissens dieser Personengruppe erscheint eine standardisierte Befragung mit vorgegebenen Antwortkategorien ungeeignet, um eine Expertenbefragung durchzuführen (Scholl 2009, 68 f.). Da bei einer *leidfadengestützte Expertenbefragung* die Befragung grob strukturiert werden kann und dennoch spontan auf die Befragungssituation reagieren werden, erscheint diese Umsetzungsvarianten für beide Befragungsrunden als geeignet. Durch die grobe Strukturierung kann gewährleistet werden, dass die fokussierten Fragestellungen adressiert werden. Durch die Freiheitsgrade während der Befragung kann auf das individuelle Antwortverhalten des jeweiligen Experten eingegangen werden.

Im Rahmen dieser Arbeit wurden zwei Befragungsrunden mit unterschiedlichen Fragestellungen und unterschiedlichen Experten in unterschiedlichen Zeiträumen durchgeführt. In der ersten Befragungsrunde wurden professionelle Softwareentwickler, mit mehrjähriger Erfahrung in der Entwicklung mobiler Unternehmensapplikationen befragt. Untersuchungsziel war zum einen die aktuelle Vorgehensweise sowie die aktuellen Herausforderungen der professionellen Softwareentwickler bei der Umsetzung mobiler Unternehmensapplikationen zu eruieren. Zum anderen wurden die Erfahrungen und Einschätzungen der professionellen Softwareentwickler hinsichtlich der modellgetriebenen Entwicklung, speziell in Bezug auf mobile Unternehmensapplikationen, erfragt. Grund hierfür ist die herausragende Stellung der modellgetriebenen Entwicklung als Entwicklungstechnik für die Endbenutzer-Entwicklung (siehe Kapitel 2.4.3.2.1).

Die Vorbereitung, Durchführung und Auswertung der Experteninterviews in Befragungsrunde I wurde gemeinsam mit dem damaligen Studenten Michael Schaub (2012) als Teil seiner Masterarbeit durchgeführt. In der zweiten Befragungsrunde wurden Mitarbeiter aus unterschiedlichen IT-nahen Bereichen zu Ihrer Einschätzung in Bezug auf die Endbenutzer-Entwicklung mobiler Unternehmensapplikationen befragt. Untersuchungsziel hierbei war ein möglichst breites Verständnis über die unterschiedlichen Potenziale und Herausforderungen der Endbenutzer-Entwicklung für mobile Unternehmensapplikationen zu erhalten. Die Vorbereitung, Durchführung und Auswertung der Experteninterviews in Befragungsrunde II wurde gemeinsam mit der damaligen Studentin Lisa Kolbe (2014) als Teil ihrer Masterarbeit durchgeführt. Die Ergebnisse beider Befragungsrunden dienen im weiteren Verlauf der Arbeit als Grundlage zur Ermittlung von Anforderungen an das in der vorliegenden Arbeit umzusetzende Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen.

Die Vorgehensweise der durchgeführten empirischen Untersuchung orientiert sich am beschriebenen Vorgehen von Gläser und Laudel (2009, 33 ff.). Die Vorgehensweise beginnt mit der Formulierung eines Untersuchungszieles. Hierbei werden insbesondere die zu beantwortenden Fragen formuliert. Diese Fragen werden auf Basis von theoretischen Vorüberlegungen getroffen. Die theoretischen Vorüberlegungen dieser Arbeit wurden bereits in Kapitel 2 der vorliegenden Arbeit behandelt und werden daher im Folgenden nicht wiederholt aufgeführt. Anschließend wird eine Untersuchungsstrategie zur Beantwortung der formulieren Fragestellungen entwickelt. Die Untersuchungsstrategie umfasst nach Gläser und Laudel (2009, 93 ff.) die Auswahl geeigneter Methoden für die Datenerhebung und -auswertung, die Auswahl geeigneter Interviewpartner sowie eine Zeitplanung für die Interviews. In diesem Kapitel wurde das leitfadengestützte Experteninterview als Datenerhebungsmethode sowie die qualitative Inhaltsanalyse als Auswertungsmethode bereits beschrieben und begründet. Daher beschränken sich die folgenden Ausführungen bzgl. der beiden Befragungsrunden auf die beiden verbleibenden Aspekte der Untersuchungsstrategie. Anschließend werden die Interviews durchgeführt und ihre gesammelten Daten ausgewertet. Schließlich werden die erzielten Untersuchungsergebnisse interpretiert, um die formulierten Fragestellungen zu beantworten. Die beschriebene Vorgehensweise ist in Abbildung 4-1 illustriert.

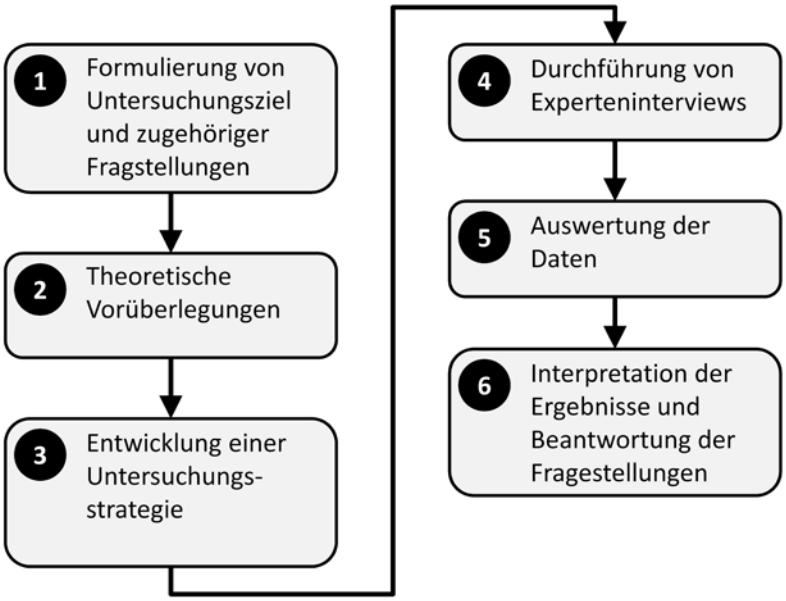

Abbildung 4-1: Vorgehen bei der empirischen Untersuchung
Quelle: Eigene Darstellung in Anlehnung an (Gläser/Laudel 2009, 35)

4.2 Befragung I: Potenziale und Herausforderungen bei der modellgetriebenen Entwicklung mobiler Unternehmensapplikationen

Nachfolgend werden das Untersuchungsziel, die zugehörige Untersuchungsstrategie, die Datenerhebung und –auswertung sowie die zentralen Untersuchungsergebnisse der ersten Befragung beschrieben. Schwerpunkt dieser Befragung war die Untersuchung der Potentiale und Herausforderungen bei der modellgetriebenen Entwickklung mobiler Unternehmensapplikationen.

4.2.1.1 Untersuchungsziel und zugehörige Fragestellungen

Untersuchungsziel der ersten Befragung ist die aktuelle Vorgehensweise in der Praxis bei der Entwicklung mobiler Unternehmensapplikationen. Hierbei sind insbesondere die aktuellen Projektanforderungen, Problemstellungen und eingesetzten Entwicklungswerkzeuge von Interesse. Neben der Erhebung des aktuellen Standes in der Unternehmenspraxis ist ein weiterer Bestandteil des Untersuchungsziels die Erfahrung der Experten mit modellgetriebenen Entwicklungsansätzen im Allgemeinen sowie mit modellgetriebenen Entwicklungsansätzen für Benutzungsschnittstellen im Besonderen. Hieraus ergeben sich folgende Fragestellungen:

(1) Wie werden mobile Unternehmensapplikationen aktuell entwickelt?

(2) Wie wird der Einsatz modellgetriebener Entwicklungsansätze für die Entwicklung mobiler Unternehmensapplikationen beurteilt?

(3) Wie wird der Einsatz modellgetriebener Entwicklungsansätze für Benutzungsschnittstellen für die Entwicklung mobiler Unternehmensapplikationen beurteilt?

4.2.1.2 Untersuchungsstrategie

Um die formulierten Fragestellungen zu beantworten mussten relevante Interviewpartner bzw. Experten gefunden werden. Diese sollten mehrjährige Erfahrung in der Entwicklung mobiler Applikationen im Unternehmenskontext besitzen. Um geeignete Experten zu identifizieren wurde einerseits das Karriereportal XING[15] genutzt als auch auf Industriekontakte des Lehrstuhls zurückgegriffen. Insgesamt konnten so sieben geeignete Experten identifiziert werden. Das Profil der interviewten Experten ist in Tabelle 4-1 dargestellt.

Tabelle 4-1: Übersicht über die Interviewpartner der Befragungsrunde I
Quelle: Eigene Darstellung

Interviewpartner	Position und Erfahrungshintergrund
Experte A	Presales Spezialist im Bereich mobile Unternehmensapplikationen mit mehrjähriger Erfahrung als Technologieberater im Bereich mobiler Unternehmensapplikationen.
Experte B	Softwareentwickler mit mehrjähriger Erfahrung in den Bereichen SAP und mobile Unternehmensapplikationen.
Experte C	Senior Software Architekt mit mehrjähriger Erfahrung in den Bereichen SAP und mobile Unternehmensapplikationen.
Experte D	Softwareentwickler mit mehrjähriger Erfahrung in der Entwicklung mobiler Unternehmensapplikationen.
Experte E	Projektleiter und Produktmanager mit mehrjähriger Erfahrung in den Bereichen SAP und mobile Unternehmensapplikationen.
Experte F	Projektleiter mit mehrjährigen Erfahrungen in der Entwicklung von Unternehmensapplikationen im Allgemeinen und mobiler Unternehmensapplikationen im Speziellen.
Experte G	Geschäftsführer eines auf Auftragsentwicklung mobiler Applikationen spezialisierten mittelständischen Unternehmens mit mehrjähriger Entwicklungs- und Projekterfahrung im Bereich mobiler Unternehmensapplikationen.

Ausgehend vom formulierten Untersuchungsziel sowie der dazugehörigen Fragestellungen wurde ein Interviewleitfaden konzipiert, welcher als grobes Strukturierungsinstrument für die Interviewdurchführung diente. Der Aufbau des Leitfadens orientiert sich am Leitfaden von Weßel (2010, 931). Übertragen auf die formulierten Fragestellungen gliedert sich der Interviewleitfaden in die folgenden Bestandteile:

[15] http://www.xing.com, zugegriffen am 5.10.2013

1. **Einleitung**

   Zu Beginn werden die Themen des Interviews und der Ablauf kurz dargestellt. Zudem besteht die Möglichkeit Fragen des Interviewpartners zu klären.

2. **Exploration der gegenwärtigen Situation**

   In diesem Kapitel wird nach der aktuellen Vorgehensweise bei der Entwicklung mobiler Unternehmensapplikationen gefragt. Hierbei wird u.a. danach gefragt, für welche mobilen Betriebssysteme aktuell entwickelt wird, welche Entwicklungswerkzeuge hierzu eingesetzt werden, inwiefern hierbei Guidelines, Modelle oder Patterns eingesetzt werden, in welcher Form Wiederverwendung stattfindet und wie die Dokumentation der entwickelten Applikationen erfolgt.

3. **Exploration der Erfahrungen und Einschätzungen zur modellgetriebenen Entwicklung im Allgemeinen**

   In diesem Kapitel wird die Einstellung der Experten gegenüber der modellgetriebenen Entwicklung im Allgemeinen abgefragt. Da nicht davon ausgegangen werden kann, dass jeder Experte bereits Erfahrungen in der modellgetriebenen Entwicklung im Bereich mobiler Unternehmensapplikationen hat, wird die Erfahrung des Experten mit diesem Entwicklungsansatz zu Beginn des Kapitels abgefragt. Sollte keine Erfahrung in diesem Themenbereich existieren, wird eine kurze Einführung durch den Interviewer gegeben. Anschließend wird die Interviewperson nach ihrer Einschätzung zum Vorgestellten befragt. Sollte hingegen bereits Erfahrung im Bereich modellgetriebener Entwicklung existieren, wird der Experte direkt nach seinen Erfahrungen und seiner Einschätzung gefragt.

4. **Exploration der Erfahrungen und Einschätzungen zur modellgetriebenen Entwicklung von Benutzungsschnittstellen**

   In diesem Kapitel wird die Einstellung des Experten gegenüber der modellgetriebenen Entwicklung von Benutzungsschnittstellen erfragt. Aufgrund der geringen Verbreitung modellgetriebener Entwicklungsansätze für Benutzungsschnittstellen in der Praxis (Meixner et al. 2011, 8) wird in diesem Kapitel generell davon ausgegangen, dass die Experten mit diesem Entwicklungsansatz keine oder kaum Erfahrungen besitzen. Daher wird analog zum Vorgehen im vorherigen Kapitel eine Einführung in diesen Entwicklungsansatz gegeben. Anschließend werden die Experten nach ihrer Einschätzung zu diesem Entwicklungsansatz im Bereich mobiler Unternehmensapplikationen befragt.

5. **Abschluss**

   Im Schlussteil des Interviews wird nochmals darauf hingewiesen, dass die Auswertung und Verwendung der erhobenen Interviewdaten in anonymisierter Form erfolgt, gefolgt von Dank und Abschied.

Der zugehörige Leitfaden der Interviews ist Anhang A zu finden.

4.2.1.3 Datenerhebung und -auswertung

Die Interviews fanden im Zeitraum vom 27. Juli 2012 bis zum 22. November 2012 statt. Alle Interviews wurden persönlich, entweder in den Räumen des Lehrstuhls oder Vorort bei den Interviewpartnern, durchgeführt. Die Interviews dauerten zwischen 49 und 77 Minuten. Von allen Interviews wurden Tonbandaufzeichnungen angefertigt. Diese wurden anschließend transkribiert, d.h. die Tonbandaufzeichnungen wurden in eine schriftliche Form überführt. Den Empfehlungen von Scholl (2009, 71) folgend, kommt es bei einem Experteninterview primär auf die Inhalte der Antworten an und weniger auf die Erzählweise und verwendete Sprache. Daher wurden bei der Transkription weder paraverbale Aspekte, wie bspw. Stimmlage oder Tonfall, noch nonverbalen Aspekte, wie bspw. Gestik oder Mimik, übernommen. Zudem wurden umgangssprachliche Redewendungen, „verschluckte“ Silben und unvollständig ausgesprochene Sätze bei der Transkription geglättet und Füllwörter entfernt (Bortz/Döring 2006, 311 f.).

Im Zuge der Auswertung wurden die transkribierten Interviews mehrfach gelesen. Aufgrund des explorativen Erkenntnisziels und der relativ geringen Stichprobengröße erschien eine Quantifizierung der Interviews nicht zweckdienlich. Stattdessen wurden die informationstragenden Aussagen der Interviews in Anlehnung an Scholl (2009, 72) schrittweise zusammengefasst, thematisch gruppiert und systematisiert. Zur strukturierten Darstellung des Untersuchungsergebnisses wurden die formulierten Fragenstellungen aus dem Interviewleitfaden als Auswertungskategorien verwendet. Zudem wurden die zugeordneten Ergebnisse weiterhin in geeignete Zwischenkategorien unterteilt (Scholl 2009, 72). Bei der qualitativen Inhaltsanalyse können unterschiedliche Analyseeinheiten verwendet werden. In Anlehnung an den Ratschlägen von Gläser und Laudel (2009, 210) wurde ein Absatz als Analyseeinheit verwendet, da einzelne Sätze oder Satzteile i.d.R. zu klein waren, um sinnvolle Interpretationen zu erlauben.

4.2.1.4 Untersuchungsergebnisse

Nachfolgend werden die zusammengefassten Aussagen und Erkenntnisse aus den sieben durchgeführten Interviews zu den drei formulierten Fragestellungen präsentiert.

4.2.1.4.1 Beschreibung der aktuelle Situation

Der Inhalt dieses Kapitels beschreibt die aktuelle Situation der Experten bei der Entwicklung mobiler Unternehmensapplikationen. Die Antworten der Interviewpartner wurden in die folgenden drei Kategorien zusammengefasst: 1) fokussierte Implementierungsvarianten und unterstützte mobile Betriebssysteme, 2) Entwicklungswerkzeuge, 3) Vorgehensweisen, 4) Dokumentationsformate, 5) Entwicklungsrichtlinien und Wiederverwendung, 6) Erfolgskriterien, 7) Problemstellungen und 8) Verbesserungsideen.

Fokussierte Implementierungsvarianten und unterstützte mobile Betriebssysteme

In Bezug auf die genutzten Implementierungsvarianten nennen die befragten Experten i.d.R. native Applikationen. Als Grund hierfür werden die Kunden- bzw. Projektanforderungen, die suboptimale Leistungsfähigkeit und die vergleichsweise eingeschränkten Entwicklungsmög-

lichkeiten von anderen Implementierungsvarianten (siehe Charakteristik C4, Kapitel 3.2) genannt. Lediglich die Experten B und G geben an, dass sie auch plattformunabhängige Applikationen auf Basis von HTML5-Technologien entwickeln.

In Bezug auf die fokussierten mobilen Betriebssysteme werden am häufigsten iOS und Android genannt. Dies wird durch die jeweiligen Kunden- oder Projektanforderungen, den Verbreitungsgrad der jeweiligen mobilen Betriebssysteme sowie deren Nutzungsstatistiken begründet. Experte F erläutert hierzu:

> *„Tatsächlich ist für uns heute primär iOS relevant, weil die Nutzungsstatistiken unserer Internetserver zeigen, das iOS in der tatsächlichen Nutzung massiv dominiert. Das heißt in einer Größenordnung von 70% haben unsere Nutzer iOS, dann hat noch ein Großteil von den verbleibenden Android, dann kommt noch ein bisschen Blackberry. Symbian ist bei der Webnutzung eigentlich nicht mehr relevant“.*

Experte D erläutert hierzu:

> *„[…] das sind Sachen, die von Kunden angefragt werden. Wenn die jetzt unbedingt auf Blackberry oder Windows Phone bestehen würden, dann würden wir sicherlich auch mehr in diese Richtung gehen, aber wir haben es bisher so erlebt, dass in der Regel Hersteller von Premiumprodukten auch immer erst einmal in Richtung iOS gehen und in einem weiteren Schritt Android nachziehen. […] Einige Firmen verfolgen schließlich eine „Bring-Your-Own-Device-Strategie“ und da ist iOS natürlich auch wieder vorne gefolgt von Android.“*

Lediglich Experte G gibt an, dass sein Unternehmen für nahezu alle mobilen Betriebssysteme native Applikationen entwickelt. Er erläutert hierzu:

> *„Technologie ist bei uns eigentlich Mittel zum Zweck. Wir sind um Unabhängigkeit bemüht. Wir versuchen konkret anhand der vorliegenden Projektanforderungen zu entscheiden, welches mobile Betriebssystem das geeignetste ist. Dabei spielen u.a Security-, Komplexitäts-und Animationskriterien sowie zu unterstützende Sensoren eine Rolle für die Entscheidung für oder gegen ein mobiles Betriebssystem.“*

Experte A nennt zusätzlich Windows CE und Windows Mobile, welche nach seiner Aussage häufig auf Spezialgeräten im industriellen Umfeld im Einsatz sind. Ähnlich äußert sich Experte C, welcher bereits mobile Applikationen für das mobile Betriebssystem von Motorola entwickelt hat. Nach seiner Aussage kommt das mobile Betriebssystem von Motorola hauptsächlich im industriellen Umfeld zum Einsatz.

Entwicklungswerkzeuge

Der Aspekt Entwicklungswerkzeuge wurde von den Experten vergleichsweise intensiv beantwortet, was die Bedeutung entsprechender Werkzeuge im Entwicklungsprozess hervorhebt. Alle Experten führten an, dass sie für die Entwicklung nativer Applikationen die von

den zugehörigen Herstellern bereitgestellten Entwicklungswerkzeuge verwenden. Häufig wurden hierbei die Entwicklungsumgebung XCode für die iOS-Entwicklung, sowie Eclipse für die Android-Entwicklung genannt. Experte E nannte zusätzlich die Entwicklungsumgebung „.NET Visual Studio“ von Microsoft zur Entwicklung von Windows Mobile Applikationen.

Neben den Entwicklungswerkzeugen für native Applikationen setzen einige Experten auch betriebssystemübergreifende Werkzeuge ein. Vergleichsweise häufig wird die MEAP „Sybase Unwired Platform“ (SUP) genannt, welche nach dem Interviewzeitraum durch den Hersteller SAP in „SAP Mobile Platform“ umbenannt wurde. Jedoch ist dieses Werkzeug nach Ansicht der Experten noch verbesserungswürdig. Experte B äußert sich in diesem Kontext bspw. folgendermaßen:

> *„In der SUP gibt es noch einiges zu tun. Beispielsweise ist die Integration in die bestehenden SAP-Produkte noch nicht wirklich gut und auch die SUP selber hat noch einige Macken die einen schon einmal viel Zeit kosten können“*.

Experte C setzt hingegen auf das „Metaframework“ Syclo, welches nach dem Interviewzeitraum ebenfalls in die SAP Mobile Platform (ab Version 3.0) integriert wurde. Experte C äußert sich hierzu wie folgt:

> *„Unser Steckenpferd ist eigentlich Syclo. Das ist ein Metaframework, mit dem man Applikationen komplett beschreiben und dann auf verschiedenen Geräten deployen kann. Da ist auch schon out-of-the-box eine SAP-Anbindung integriert, die man zwar noch ein bisschen anpassen muss, aber damit hat man bei der Entwicklung gegen das Backend eigentlich gar nicht soviel Arbeit“*.

Für Experte E war insbesondere der „Investitionsschutz“ der verwendeten Entwicklungswerkzeuge wichtig, d.h. das diese auch zukünftig von den Herstellern angeboten und weiterentwickelt werden. Neben Entwicklungswerkzeugen zur Erstellung der mobilen Applikationen setzt Experte A ein Werkzeug (balsamiq[16]) zur Erstellung von low-fidelty Prototypen ein. Dabei handelt es sich um rudimentäre Skizzen, um die Benutzungsschnittstelle der Applikation für Diskussionszwecke zu visualisieren.

Vorgehensweisen

In Bezug auf die Vorgehensweisen bei der Entwicklung mobiler Applikationen ist auffällig, dass alle Experten ein agiles Vorgehen nutzen. Als Grund wird die häufige Änderung der Anforderungen im Projektverlauf genannt. Zudem empfinden die Experten ein aufwändiges Vorgehensmodell aufgrund der verhältnismäßig geringen Komplexität von mobilen Applikationen nicht sinnvoll. Experte C äußert sich hierzu wie folgt:

[16] http://balsamiq.com, zugegriffen am 12.12.2012

„Es ist in der Regel so, dass wir zusammen mit unseren Kunden einen Prototypen definieren, der gewisse Anforderungen erfüllt, damit kann der Kunden dann beispielsweise schon einmal zehn Techniker in den Praxiseinsatz schicken, die dann damit arbeiten und Feedback liefern können und anhand dieses Feedbacks wird die Applikation dann in mehreren Iterationen erweitert bis hin zur finalen Version. In der Praxis fehlt uns dann häufig einfach die Zeit die Entwicklung mit einem solchen Vorgehensmodell aufzubauen. Das hat zwar jeder von uns schon einmal gehört, wird aber nicht gelebt."

Dieser Interviewauszug zeigt zudem, dass der zeitliche Aspekt ein kritischer Faktor in der Entwicklung ist. Die befragten Experten gaben an, Entwicklungsprojekte für mobile Applikationen häufig als Auftragnehmer durchzuführen, wobei die Projektlaufzeit und das Budget in der Regel knappe Ressourcen sind.

Experte A nannte *Scrum* als das in vielen seiner durchgeführten Projekte verwendete Vorgehensmodell. Wobei es sich bei Scrum ebenfalls um eine agile Vorgehensweise handelt (Rubin 2013, 1 ff.). Experte D nannte den umfangreicheren „Rational Unified Process" (RUP) (vgl. z.B. (Essigkrug/Mey 2007, 1 ff.; Sommerville 2011, 50 ff.)), welcher das offizielle Vorgehensmodell seines Unternehmens darstellt. Der Experte erwähnt jedoch auch, dass RUP in seinem vollen Umfang selten eingehalten wird.

Dokumentationsformate

In Bezug auf die verwendeten Dokumentationsformate wurden häufig Powerpoint-Folien und Word-Dokumente genannt. Die Experten A und E nannten zusätzlich Wikis als genutztes Dokumentationsformat. Jedoch erläuterte Experte C, dass Dokumentationen häufig nicht im Projektbudget berücksichtigt werden und daher oftmals nicht angefertigt werden. Experte C äußert sich hierzu wie folgt:

„Rein in Bezug auf die Dokumentation ist es oft so, dass der Kunde nicht gewillt ist uns das zu bezahlen. Wir dokumentieren das dann nur für uns intern soweit, dass wir technisch noch wissen, wo wir hin greifen müssen. Wir erstellen jedoch in der Regel keine Hochglanz Schulungsunterlagen anhand derer ein dritter, durch reines lesen, weiß wie er die Applikation warten bzw. weiterentwickeln kann. Also es kommt vor, dass ein solche Dokumentation auch verlangt ist, aber das ist eigentlich die Ausnahme".

Entwicklungsrichtlinien und Wiederverwendung

In Bezug auf die verwendeten Entwicklungsrichtlinien wurde häufig genannt, dass von den Herstellern der mobilen Betriebssysteme Gestaltungsrichtlinien verfügbar sind und diese bei der eigenen Entwicklung berücksichtigt werden. Zusätzlich wurde angeführt, dass häufig bestimmte „Corporate Identity"-Vorgaben einzuhalten sind. Insbesondere wurde hierbei oftmals das Unternehmenslogo und vorgegebene Farbschemata des Unternehmens genannt.

In Bezug auf die Wiederverwendung erwähnte Experte A, dass in seinen Projekten i.d.R. versucht wird die Datenobjekte wiederzuverwenden. In Bezug auf die Benutzungsschnittstelle sehen die befragten Experten die Wiederverwendung hingegen kritisch. Als Gründe hierfür

nennt Experte D beispielsweise individuelle Anforderungen des Auftraggebers sowie den Einsatz unterschiedlicher Programmiersprachen und Programmierschnittstellen. Experte A erläutert jedoch, dass Entwickler in seinem Unternehmen häufig mit Templates arbeiten, welche ein Grundgerüst für verschiedene Anwendungstypen bereitstellen. Experte A erklärt hierzu:

> *Zum Beispiel haben wir auch für Android ein eigenes Framework geschrieben, an das sich jeder Entwickler, der eine App bauen will, halten muss. Er bekommt dann Templates vorgefertigt, die kann er sich runterladen und in sein Projekt einbinden und hat dann im Prinzip die Standardfarben schon drin. Falls das eine Standard-App werden soll, dann sind Buttons und Kontextmenü schon enthalten und natürlich auch gewisse Methoden schon vorbereitet. Es sind also nicht nur die UI-Elemente, sondern auch Funktionalitäten, wie das bspw. das Kontextmenü überall in der App angesprochen werden kann.*

Erfolgskriterien

In Bezug auf die Erfolgskriterien einer mobilen Unternehmensapplikation wird von den Experten A, E und F genannt, dass die Benutzungsschnittstelle ansprechend sein sollte. Hierfür wird von den Experten der Begriff „fancy“ genutzt. Experte A erläutert in diesem Zusammenhang, dass der Benutzer Spaß bei der Verwendung der Applikation haben sollte. Als weiteres Erfolgskriterium nennt Experte A, dass die Qualität der Daten im Backendsystem wichtig ist und dass die in der mobilen Applikation benötigten Attribute häufig nur eine Teilmenge der im Backendsystem vorhandenen Attribute des Datenobjektes sind. Experte E erläutert hierzu:

> *„Die Apps müssen schnell, smooth und natürlich fancy sein. […] Es ist natürlich auch toll, wenn die App ein bisschen schicker aussieht und man schöne Gesten verwenden kann. Dann ist in der Regel die Bedienung auch intuitiver und einfacher“.*

Problemstellungen

In Bezug auf die aktuellen Problemstellungen der befragten Experten werden häufig Folgen der aktuellen Geräte- und Betriebssystemheterogenität gekannt. Beispiele hierfür sind die unterschiedlichen Auflösungen der mobilen Endgeräte und die unterschiedlichen Implementierungsvarianten. Der erste Aspekt bedingt die Bereitstellung unterschiedlicher Varianten der Benutzungsschnittstelle. Der zweite Aspekt erfordert unterschiedliche Entwicklerfertigkeiten. Experte E beschreibt, dass bei der Unterstützung mehrerer mobiler Betriebssysteme aktuell für jedes zu unterstützende mobile Betriebssystem eine komplette Neu-Entwicklung stattfindet und beklagt den hohen Ressourcenaufwand dieses Vorgehens. Zudem empfinden Experte A und B die kurzen Release-Zyklen der mobilen Betriebssysteme als herausfordernd, da diese einen stetigen Anpassungsaufwand der bereits entwickelten Applikationen mit sich bringen. Experte A nennt zusätzlich die Ungewissheit bzgl. der zukünftig relevanten Implementierungsvarianten als Problemstellung. Experte A erläutert hierzu:

„Die Frage, wo es da hingehen wird, ist auch noch nicht ganz geklärt; also wird es eher in Richtung HTML5 gehen, oder bleibt es bei nativen Applikationen. Da scheiden sich im Augenblick noch die Geister, auch bei den Analysten. Aber das größte Problem ist im Grunde, dass man den Prozessexperten hat, weil man den für jede Plattform braucht, weil dieser sowieso Plattform unabhängig ist, der kann aber das Design wenig beeinflussen, und man braucht unterschiedliche Skills“.

Bei Multi-Plattform Implementierungsvarianten bemängelt Experte E, dass diese aktuell hinsichtlich ihrer Leistungsfähigkeit und ihrer Entwicklungsmöglichkeiten derzeit häufig seine Anforderungen nicht erfüllen. Experte F erläutert hierzu, dass Benutzer auch bei einer Web-Applikation ein Aussehen und Bedienverhalten in Bezug auf die verwendeten Anzeige- und Bedienelemente als auch in Bezug auf die Geschwindigkeit wie bei einer nativen Applikation erwarten.

Verbesserungsideen

Die befragten Experten nannten mehrere Verbesserungsideen auf unterschiedlichen Abstraktionsebenen. Experte A nannte als Verbesserungsidee, dass Entwicklungswerkzeuge bereits rudimentäre Bildschirmmasken generieren sollten. Diese sollten seiner Meinung nach jedoch nicht zu viele Details besitzen, damit eine individuelle Anpassung möglich ist. Der jeweilige Detaillierungsgrad sollte jedoch einstellbar sein, da in manchen Fällen laut seiner Aussage auch eine Standard-Applikation ausreichend ist. Er erläutert hierzu:

„Bei nativen Apps, sollte meiner Meinung nach die Plattform schon gewisse Screens vorgenerieren können, so das man einen lockereren Einstieg für die Entwickler bietet. Diese Screens müssen auch gar nicht hübsch sein, sondern es wäre gut, wenn schon ein paar Felder da wären und ich könnte die dann einfach wiederverwenden. Das ist für mich als Entwickler so der Stolperstein Nummer eins im Moment. An der Stelle denke ich, könnten wir noch ein bisschen mehr Geschwindigkeit erreichen.“

Laut Experte E wäre ein strukturierteres Vorgehen bei der Entwicklung mobiler Unternehmensapplikationen hilfreich. Er erläutert hierzu:

„Häufig denkt man halt, dass das nur eine kleine App ist und man deswegen nicht mit schwergewichtigen Modellen etc. arbeiten muss, aber häufig ist es dann gar nicht so einfach und dann wünscht man sich doch eher strukturiert an solche Aufgaben heranzugehen.“

Experte F bezieht seine Verbesserungsideen auf die Werkzeugunterstützung. Als Beispiele nennt er verbesserte Debugging-Möglichkeiten von HTML5-Applikationen und eine automatisierte Testfall-Generierung.

4.2.1.4.2 Beurteilung modellgetriebener Entwicklungsansätze im Allgemeinen

In Bezug auf die modellgetriebene Entwicklung mobiler Applikationen war die Einstellung der befragten Experten tendenziell kritisch. Die Experten A und C erklärten, dass eine mo-

dellgetriebene Entwicklung im Bereich mobiler Applikationen in der Praxis kaum eingesetzt wird. Als möglichen Grund nannten sie die ältere Generation an Entwicklern, welche mit diesen Ansätzen nicht vertraut sind. Experte B nennt zusätzlich, dass der generierte Quellcode oftmals nicht nachvollziehbar oder zumindest ein hoher Aufwand investiert werden muss, um den generierten Quellcode zu verstehen. Die Experten B und C erläutern, dass durch eine modellgetriebene Entwicklung ein Zusatzaufwand entsteht, welcher vom Projekt bzw. vom Auftraggeber unterstützt werden müsste. Insbesondere die hohe Komplexität und der daraus resultierende hohe Einarbeitungsaufwand in zugehörige Werkzeuge werden hierbei bemängelt. Experte B erläutert hierzu:

> *„[...] In erster Linie natürlich der Zusatzaufwand. Es ist bei uns natürlich alles sehr eng getaktet, das Budget ist immer knapp und dann bringt uns das natürlich nichts, wenn wir erst einmal möglichst alles modellieren. […] Was ich bis jetzt immer gesehen habe ist, dass die Tools unglaublich komplex sind und man muss in aller Regel viel Zeit investieren, um diese dann auch richtig bedienen zu können, was in der Praxis nicht gern gesehen wird“.*

In Bezug auf die modellgetriebenen Entwicklungswerkzeuge wird zusätzlich die fehlende oder mangelhafte Round-Trip-Engineering Fähigkeit von den Experten D, E und F bemängelt. Unter *Round-Trip-Engineering* wird in der modellgetriebenen Softwareentwicklung die Sicherstellung der Konsistenz zwischen der Modellebene und dem Quellcode verstanden (Stahl/Völter 2005, 82). Dabei müssen zwei Anpassungsrichtungen berücksichtigt werden. Beim *Forward-Engineering* sollte die Anpassung der Modellebene automatisch zu einer entsprechenden Anpassung des Quellcodes führen (Stahl/Völter 2005, 28). Beim *Reverse-Engineering* sollte eine Änderung des Quellcodes automatisch zu einer entsprechenden Anpassung der Modellebene führen (Stahl/Völter 2005, 82).

Zusätzlich wird die Qualität des generierten Quellcodes bemängelt. Experte C bezweifelt die Wartbarkeit und Erweiterbarkeit des genierten Quellcodes:

> *„Eine Frage die sich mir stellt ist die Qualität der generierten Quelltexte. Wenn man beispielsweise ein Word-Dokument als HTML-Seite abspeichert und später im Quelltext etwas verändern möchte, dann wird man sich damit in aller Regel sehr schwer tun und so etwas dürfte bei einer solchen Lösung natürlich nicht passieren. Der Quelltext muss natürlich vernünftig zugänglich sein, wenn ich beispielsweise Felder oder Feldnamen verändern will, dann muss ich wissen wo ich in den Quelltext greifen kann und das muss mit vernünftigem Aufwand zu realisieren sein“.*

Zudem sieht Experte F nachteilig, dass sich mit modellgetriebene Entwicklungswerkzeugen häufig nur die rudimentären Elemente generieren lassen, wobei die anschließenden Anpassungen einen hohen manuellen Aufwand verursachen. Als Beispiel nannte er die fehlende Möglichkeit Validierungsfunktionen von Eingabefeldern und zugehörige Feedbackmechanismen zu modellieren. Experte F erläutert hierzu:

„Bei den Eingabevalidierungen bin ich bereits am Ende, d.h. das aller Elementarste, dass ich im Modell hinterlegen kann, welche Zeichen in einem Feld zulässig sind und eventuell eine automatische Validierungsfunktion fehlen schon. Das heißt der Benefit den ich durch die Verwendung des Modells habe ist, dass ich mir den Dialog generieren lassen kann, was ein Entwickler auch in zehn Minuten kodiert hat und der Teil der wirklich Arbeit bedeutet muss auch hier wieder von Hand gemacht werden. Zusätzlich habe ich die Komplexität noch erhöht, weil ich ein weiteres Tool in meiner Kette habe. Wenn die Tools in diesem Bereich stärker wären, dann wäre das vielleicht eine Option, aber so wie es im Moment aussieht hört die Funktionalität einfach zu früh auf“.

Experte E ist die Unterscheidung zwischen einfachen und komplexeren mobilen Applikationen wichtig. Nach seiner Einschätzung ist ein modellgetriebener Entwicklungsansatz bei einfachen Applikationen möglich; bei komplexeren Applikationen nicht.

Experte B erklärt, dass er eine modellgetriebene Entwicklung bei der Umsetzung der Datenschnittstelle bereits eingesetzt hat und damit gute Erfahrungen gemacht hat. Zur Entwicklung der Benutzungsschnittstelle hat er noch keinen modellgetriebenen Ansatz verwendet.

Als Vorteil eines modellgetriebenen Entwicklungsansatzes sieht Experte A vor allem die bessere Übersichtlichkeit sowie die Bewältigung von Komplexität. Zudem wird nach Ansicht des Experten E die Kommunikation und das gegenseitige Verständnis zwischen Softwareentwicklern und Fachexperten die gemeinsame Arbeit auf Modellebene gefördert.

4.2.1.4.3 Beurteilung modellgetriebener Entwicklungsansätze für Benutzungsschnittstellen

In Bezug auf die modellgetriebene Entwicklungsansätze für Benutzungsschnittstellen zeigte sich, dass die befragten Experten zunächst angaben mit der modellgetriebenen Entwicklung von Benutzungsschnittstellen keine Erfahrung zu haben. Nach einer kurzen Beschreibung konnten die befragten Experten die generellen Ideen dieses Entwicklungsansatzes jedoch zügig verstehen und auf ihr Aufgabengebiet übertragen. Zudem zeigte sich, dass sich ein Teil der Experten nach der Beschreibung an eigene Berührungspunkte mit der modellgetriebenen Entwicklung von Benutzungsschnittstellen erinnern konnten.

Als Nachteil sehen die Experte A, E und F, dass die generierten Benutzungsschnittstellen häufig nur sehr rudimentär sind und den Anforderungen an das Aussehen und die Bedienung mobiler Unternehmensapplikationen nicht genügen. Zudem sehen sie die Anpassung der generierten Bildschirmmasken als aufwändig an, da hierzu ihrer Ansicht nach der generierte Quellcode erst verstanden werden muss. Nach Ansicht des Experten B ist eine Generierung von Bildschirmmasken für unterschiedliche mobile Betriebssysteme aufgrund ihrer großen Unterschiede nicht möglich.

Aus der Perspektive der Experten A und D eignet sich ein modellgetriebenen Entwicklungsansatz für Benutzungsschnittstellen vor allem um low-fidelty Prototypen zügig zu entwickeln. Durch diese low-fidelty Prototypen wird eine Möglichkeit geschaffen, um das Datenmodell

zu validieren und die technische Umsetzbarkeit der Applikation zu demonstrieren. Experte A erläutert hierzu:

> *„Das heißt wir bewegen uns hier in einem sehr engen Rahmen, in dem wir allerdings um dem Kunden mal schnell was zu zeigen, also was zu prototypisieren, ganz schnell sind und das ist der große Vorteil. [...] Wir können somit sehr früh demonstrieren, dass die benötigten Daten schon auf dem Device vorhanden sind, weil das Datenmodell das Gleiche bleibt. Zusätzlich können wir auf diese Weise validieren, dass wir keine Fehler in der Datenstruktur haben. Entsprechend können wir sehr früh zeigen, dass eine Entwicklung der gewünschten Funktionalitäten technisch machbar ist, wenn auch noch nicht alle Feinheiten im Design vorhanden sind“*.

Nach Ansicht des Experten D sollten in einem modellgetriebenen Entwicklungswerkzeug für Benutzungsschnittstellen eigene Regeln hinterlegt werden können, welche den Generierungsprozess steuern. Experte E hält komplexere, parametrisierbare Komponenten für geeignet, um wiederkehrende Bestandteile von Benutzungsschnittstellen in mobilen Applikationen möglichst effizient zu entwickeln. Er äußert sich hierzu wie folgt:

> *„Dann sollte es ein Tool ermöglichen Standardfälle sehr einfach zu realisieren und gleichzeitig Möglichkeiten anbieten, um an dem generierten Code Änderungen vorzunehmen bzw. eingreifen zu können. Dann sollte das Tool ein vielfältiges Set an vorgefertigten Komponenten anbieten, die auch ein bisschen komplexer sind, als eine einfach Maske, gerade im Hinblick auf die Touch-Oberflächen auch mit irgendwelchen Wischgesten, die man sich da vorstellen kann“*.

Experte G findet bei der Beurteilung der modellgetriebenen Entwicklung von Benutzungsschnittstellen für mobile Unternehmensapplikationen die Unterscheidung zwischen Standard-Benutzungsschnittstellen und ausgefalleneren Benutzungsschnittstellen wichtig. Nach seiner Meinung gibt es viele „mobile“ Anwendungsfälle in Unternehmen, die mit einfachen Anzeige- und Bedienelementen, wie bspw. Listenanzeigen auskommen. In diesem Bereich sieht er Potential für eine modellgetriebene Entwicklung. Ausgefallenere Benutzungsschnittstellen verwenden nach seinem Verständnis sehr spezifische und neuartige Interaktionsmuster und zugehörige Anzeige- und Bedienelemente. Ziel dieses Anwendungstyps ist es nach seiner Auffassung vor allem positive Emotionen zu vermitteln und somit zur Nutzung der mobilen Unternehmensapplikation zu motivieren. Bei diesem Anwendungstyp sieht der Experte kein Potential für eine modellgetriebene Entwicklung.

4.2.2 Fazit und Diskussion

Die Befragung I hat gezeigt, dass insbesondere die aktuelle, hohe Heterogenität von mobilen Endgeräten mit unterschiedlichen Auflösungen, unterschiedlicher Hardware und unterschiedlichen mobilen Betriebssystemen, eine große Problemstellung für professionelle Softwareentwickler darstellt. In vielen Fällen werden die zugehörigen mobilen Unternehmensapplikationen mit unterschiedlichen Programmiersprachen, Programmierschnittstellen und Entwick-

lungswerkzeugen implementiert. Dies erfordert unterschiedliche Kenntnisse von den Entwicklern und wird generell als aufwändig empfunden.

Zudem hat sich gezeigt, dass die Zeit sowie das Budget zur Umsetzung einer mobilen Unternehmensapplikation knappe Ressourcen sind. Dies könnte die Ursache dafür sein, dass sich die Entwickler nicht mit der modellgetriebenen Entwicklung auseinandersetzen, da dies zunächst einen gewissen Einarbeitungsaufwand erfordert. Jedoch könnte dieser reduziert werden, wenn die zugehörigen Entwicklungswerkzeuge einfacher zu bedienen wären. Modellgetriebene Ansätze könnten die befragten Entwickler überzeugen, wenn sie ohne größeren Einarbeitungsaufwand schnelle Resultate erzielen würden. Für die Entwickler ist es jedoch auch notwendig, den generierten Quellcode zu verstehen und ggf. anpassen zu können.

Der zeitliche Faktor könnte zudem die Ursache für das gewählte Vorgehen in den Entwicklungsprojekten sein. Alle befragten Entwickler nutzen eine agile Vorgehensweise, die meistens mit der Gestaltung von rudimentären Skizzen der Benutzungsschnittstelle startet. Als Grund nennen die Entwickler häufig die Änderungen von Anforderungen während der Entwicklung sowie das schnelle Erzielen von ersten Resultaten für Demonstrationszwecke.

Als zentraler Erfolgsfaktor mobiler Unternehmensapplikationen konnte das Aussehen und die Bedienbarkeit einer mobilen Unternehmensapplikation aus der Befragung identifiziert werden. Demzufolge sollten auch die generierten Benutzungsschnittstellen des fokussierten Entwicklungswerkzeuges die Gestaltungsrichtlinien der jeweiligen mobilen Betriebssysteme erfüllen. Die Befragung hat zudem ergeben, dass gewisse wiederkehrende Bestandteile einer mobilen Unternehmensapplikation existieren, welche sich für einen modellgetriebenen Entwicklungsansatz eigenen. Zudem hat sich gezeigt, dass zwischen Standard-Anforderungen und spezifischen Anforderungen unterschieden werden sollte. Standard-Anforderungen eignen sich für einen modellgetriebenen Entwicklungsansatz; wohingegen spezifische Anforderungen i.d.R. eine Implementierung auf Ebene des Quellcodes erfordern. Jedoch sollte es auch bei einer Standard-Applikation möglich sein, die „Corporate Identity“-Vorgaben des Unternehmens zu berücksichtigen. Hierzu zählen u.a. die Platzierung des Unternehmenslogos sowie die Verwendung der vorgegebenen Farbschemata.

4.3 Befragung II: Potenziale und Herausforderungen bei der Endbenutzer-Entwicklung mobiler Unternehmensapplikationen

Nachfolgend werden das Untersuchungsziel, die zugehörige Untersuchungsstrategie, die Datenerhebung und –auswertung sowie die zentralen Untersuchungsergebnisse der zweiten Befragung beschrieben. Schwerpunkt dieser Befragung war die Untersuchung der Potentiale und Herausforderungen bei der Endbenutzer-Entwicklung mobiler Unternehmensapplikationen.

4.3.1.1 Untersuchungsziel und zugehörige Fragestellungen

Untersuchungsziel der zweiten Befragung ist die Einschätzung von Experten bzgl. der Endbenutzer-Entwicklung mobiler Unternehmensapplikationen in der Praxis sowie die Ermittlung

von Anforderungen an ein zugehöriges Entwicklungswerkzeug. Neben diesem Untersuchungsziel wird analog zur ersten Befragung die Ist-Situation beim Einsatz und der Entwicklung mobiler Unternehmensapplikationen erfragt. Hieraus ergeben sich folgende Fragestellungen:

(1) Welche mobilen Unternehmensapplikationen werden aktuell genutzt und wie wurden diese entwickelt?

(2) Wie wird der Einsatz eines Endbenutzer-Entwicklungsansatzes generell beurteilt?

(3) Wie wird der Einsatz eines Endbenutzer-Entwicklungsansatzes für die Entwicklung mobiler Unternehmensapplikationen beurteilt?

(4) Welche Anforderungen werden an ein Werkzeug zur Endbenutzer-Entwicklung mobiler Unternehmensapplikationen gestellt?

4.3.1.2 Untersuchungsstrategie

Um die formulierten Fragestellungen zu beantworten, müssen analog zur ersten Befragung relevante Interviewpartner bzw. Experten gefunden werden. Die Auswahl der befragten Experten erfolgte vor dem Hintergrund, möglichst unterschiedliche Blickwinkel auf das Themengebiet zu erhalten. Daher wurden Experten mit unterschiedlichen Berührungspunkten im Bereich mobiler Unternehmensapplikationen gewählt. Diese sollten aufgrund ihrer Position im Unternehmen Berührungspunkte mit dem Einsatz und/oder der Entwicklung mobiler Unternehmensapplikationen besitzen. Um geeignete Experten zu identifizieren wurden relevante Messen und Kontaktveranstaltungen mit dem Themenschwerpunkt mobile Unternehmensapplikationen besucht. Beispiele sind ein Treffen der Arbeitsgruppe „Mobile Business“ der Deutschsprachigen SAP Anwendergruppe (DSAG) in St. Leon Roth, ein „Mobility“ Workshop von IBM in Böblingen oder der Besuch der Messe „Communications World“ in München. Zudem wurde auf bestehende Industriekontakte des Lehrstuhls zurückgegriffen. Insgesamt konnten so 9 geeignete Experten identifiziert werden, deren Profil in Tabelle 4-2 dargestellt ist.

Tabelle 4-2: Übersicht über die Interviewpartner der Befragungsrunde II
Quelle: Eigene Darstellung

Interviewpartner	Position und Erfahrungshintergrund
Experte A	Sprecher des Vorstandes, Chief Financial Officer (CFO) und Mitinhaber einer mittelständischen Beratungsgesellschaft für Business Intelligence Lösungen - seit mehreren Jahren auch mit Angeboten für mobile Endgeräte.
Experte B	Softwareentwickler eines IT-Dienstleisters mit Erfahrung in der Entwicklung mobiler Applikationen.
Experte C	IT-Infrastrukturberater mit hoher Reisetätigkeit und mehrjähriger Erfahrung in der Nutzung mobiler Unternehmensapplikationen.
Experte D	Softwareentwickler eines IT-Dienstleisters mit mehrjähriger Erfahrung in der Entwicklung mobiler Applikationen.
Experte E	Mitgründer und Geschäftsführer eines kleineren Unternehmens mit Fokus auf der Entwicklung mobiler Applikationen.
Experte F	Innovationsmanager eines auf die "Mobilisierung" von SAP Prozessen spezialisierten Unternehmens.
Experte G	Mitgründer und Technologieverantwortlicher eines kleineren Unternehmens mit Fokus auf der Entwicklung mobiler Applikationen.
Experte H	Bereichsleiter für mobile Applikations-Entwicklung eines mittelständischen Softwareentwicklungsunternehmens.
Experte I	Anforderungsmanager für mobile Lösungen bei einem Automobil-Hersteller.

Analog zum Vorgehen aus der ersten Befragungsrunde wurde auf Basis des formulierten Untersuchungszieles sowie der dazugehörigen Fragestellungen ein Interviewleitfaden konzipiert, welcher als grobes Strukturierungsinstrument für die Interviewdurchführung diente. Der Aufbau des Leitfadens orientiert sich am Leitfaden von Weßel (2010, 931). Übertragen auf die formulierten Fragestellungen gliedert sich der Interviewleitfaden in die folgenden Bestandteile:

1. **Einleitung**

   Zu Beginn werden die Themen des Interviews und der Ablauf kurz dargestellt. Zudem besteht die Möglichkeit Fragen des Interviewpartners zu klären.

2. **Exploration des persönlichen Erfahrungshintergrundes und der gegenwärtigen Situation**

   In diesem Kapitel wird nach der Erfahrung des Befragten im Bereich mobiler Unternehmensapplikationen gefragt. Zudem wird die aktuelle Position des Experten im Unternehmen sowie dessen Aufgabengebiet erfragt. Anschließend wird je nach genanntem Aufgabengebiet und Erfahrungshintergrund des Experten gefragt, welche mobile Unternehmensapplikationen aktuell genutzt werden und/oder wie diese entwickelt und bereitgestellt werden.

3. **Exploration der Erfahrungen und Einschätzungen zur Endbenutzer-Entwicklung im Allgemeinen**

   In diesem Kapitel wird die Einstellung des Experten gegenüber der Endbenutzer-Entwicklung im Allgemeinen abgefragt. Da nicht davon ausgegangen werden kann, dass jeder Experte bereits Kenntnisse oder Erfahrungen in der Endbenutzer-Entwicklung hat, wird der zugehörige Kenntnisstand zu Beginn des Kapitels abgefragt. Sollten keine Kenntnisse in diesem Themenbereich existieren, wird eine kurze Einführung durch den Interviewer gegeben. Anschließend wird die Interviewperson nach ihrer Einschätzung zum Vorgestellten befragt. Sollten hingegen bereits Kenntnisse und Erfahrungen im Bereich Endbenutzer-Entwicklung existieren, wird die Interviewperson direkt nach ihren Erfahrungen und ihrer Einschätzung gefragt.

4. **Exploration der Erfahrungen und Einschätzungen zur Endbenutzer-Entwicklung speziell für mobile Unternehmensapplikationen**

   In diesem Kapitel wird die Einstellung des Experten gegenüber der Endbenutzer-Entwicklung speziell für mobile Unternehmensapplikationen erfragt. Der Fokus liegt auf möglichen Abweichungen zu den Aussagen des vorherigen Kapitels.

5. **Exploration von Anforderungen an ein Endbenutzer-Entwicklungswerkzeug für mobile Unternehmensapplikationen**

   In diesem Kapitel wird nach Anforderungen an ein Werkzeug zur Endbenutzer-Entwicklung mobiler Unternehmensapplikationen gefragt. Nach der Sammlung erster Anforderungen soll die Anforderungsermittlung durch die Demonstration eines beispielhaften Werkzeuges zur Endbenutzer-Entwicklung unterstützt werden. Hierzu wird das Werkzeug Appery.io[17] (siehe Kapitel 6.2.4) verwendet. Anschließend werden die Interviewpartner nach Ihrer Einschätzung zu diesem Werkzeug gefragt, um weitere Anforderungen zu ermitteln.

6. **Abschluss**

   Im Schlussteil des Interviews wird nochmals darauf hingewiesen, dass die Auswertung und Verwendung der erhobenen Interviewdaten in anonymisierter Form erfolgt, gefolgt von Dank und Abschied.

Der zugehörige Leitfaden der Interviews ist Anhang B zu finden.

4.3.1.3 Datenerhebung und –auswertung

Die Interviews fanden im Zeitraum vom 11. November 2013 bis zum 13. Dezember 2013 statt. Die Interviews wurden größtenteils persönlich, entweder in den Räumen des Lehrstuhls oder Vorort bei den Interviewpartnern, durchgeführt. Aufgrund der räumlichen Distanz und schwierigen Terminfindung wurden zwei der Interviews telefonisch durchgeführt. Die Präsen-

[17] http://appery.io/, zugegriffen am 10.10.2013

tation von Powerpoint-Folien zur kurzen Vorstellung des Endbenutzer-Entwicklungsansatzes sowie die Demonstration des beispielhaften Werkzeuges Appery.io erfolgte dabei durch das Werkzeug TeamViewer[18].

Die Interviews dauerten zwischen 49 und 71 Minuten. Von allen Interviews wurden Tonbandaufzeichnungen angefertigt. Diese wurden anlog zur Auswertung der Interviews aus der ersten Befragungsrunde transkribiert. In diesem Zuge wurden paraverbale- und nonverbale Aspekte nicht berücksichtigt und die gesprochenen Sätze geglättet sowie um Füllwörter bereinigt.

Anschließend wurden die transkribierten Interviews mehrfach gelesen. Analog zur ersten Befragung wurde aufgrund des explorativen Erkenntnisziels und der relativ geringen Stichprobengröße keine Quantifizierung der Interviews durchgeführt. Stattdessen wurden die informationstragenden Aussagen der Interviews in Anlehnung an Scholl (2009, 72) schrittweise zusammengefasst, thematisch gruppiert und systematisiert. Zur strukturierten Darstellung des Untersuchungsergebnisses wurden die formulierten Fragenstellungen aus dem Interviewleitfaden als Auswertungskategorien verwendet. Zudem wurden die zugeordneten Ergebnisse weiterhin in geeignete Zwischenkategorien unterteilt (Scholl 2009, 72). Im Zuge der Auswertung durch eine qualitativen Inhaltsanalyse wurden in Anlehnung an die Ratschläge von Gläser und Laudel (2009, 210) Absätze als Analyseeinheit verwendet.

4.3.1.4 Untersuchungsergebnisse

Nachfolgend werden die Aussagen der 9 Interviewpartner zu den vier formulierten Fragestellungen präsentiert.

4.3.1.4.1 Beschreibung der aktuelle Situation

Der Inhalt dieses Kapitels beschreibt die aktuelle Situation der Interviewpartner in der Nutzung und Entwicklung mobiler Unternehmensapplikationen.

Die Antworten der Interviewpartner wurden in die folgenden drei Kategorien zusammengefasst: 1) Genutzte und entwickelte mobile Unternehmensapplikationen, 2) eingesetzte Entwicklungswerkzeuge und 3) Vorgehensweise bei der Entwicklung.

Genutzte und entwickelte mobile Unternehmensapplikationen

In Bezug auf die genutzten und entwickelten mobilen Unternehmensapplikationen wurden unterschiedliche Anwendungsfälle genannt. Beispiele sind: eine mobile Datenerfassungsapplikation für die Lagerlogistik (Experte A), eine mobile Payment Applikation (Experte H), eine Kundenverwaltungsapplikation (Experte C), Einkaufsassistenten (Experte G), Zeiterfassungs- und Projektmanagement Applikationen (Experten B, D und F) sowie eine mobile Applikation zur Steuerung bestimmter Einstellungen eines Autos (Experte D). Dabei bestätigt

[18] http://www.teamviewer.com/de/, zugegriffen am 10.10.2013

sich der aus der Literatur identifizierte Anwendungsschwerpunkt auf den klassischen Aktivitäten mobiler Mitarbeiter (siehe Kapitel 2.2.6).

Eingesetzte Entwicklungswerkzeuge

Bezüglich der eingesetzten Entwicklungswerkzeuge war das Ergebnis ähnlich wie bei der ersten Befragungsrunde. Am meisten wurden XCode für die iOS-Entwicklung und die Android Development Tools zur Android-Entwicklung genannt. Zusätzlich wurde das Konkurrenzprodukt zu XCode, AppCode[19] von IntelliJ, genannt (Experte D) sowie Mobile Add-Ons für die Business Intelligence Lösungen Business Objects von SAP und Cognos von IBM (Experte A).

Vorgehensweise bei der Entwicklung

Auch bei der geschilderten Vorgehensweise bei der Entwicklung mobiler Unternehmensapplikationen waren die Aussagen der Experten ähnlich wie die Ergebnisse aus der ersten Befragung. Generell nutzen die entwickelnden Experten eine agile Vorgehensweise. Nach Aussagen der Experten B, C und E ist das Ziel hierbei die Änderungswünsche des Kunden inkrementell aufzunehmen und umzusetzen. Experte E erläuterte in diesem Zusammenhang, dass seine Auftraggeber ein mehrseitiges Pflichtenheft oftmals nicht lesen und eine darauf basierende Festhaltung der Anforderungen daher nicht geeignet ist. Experte E erläutert hierzu:

> *„In den Anfangszeiten haben wir einmal einem Kunden ein 30-seitiges Pflichtenheft vorgelegt. Es war ein Riesenprojekt und wir wollten auf Nummer Sicher gehen. Das war absolut notwendig, weil dahinter ein großes System für Web, Android und iOS lag. Der Kunde hat das Pflichtenheft nie gelesen. Das ist so. Die Leute lesen es nicht."*

Zudem erklärt Experte C in diesem Zusammenhang, dass die Anforderungen der Auftraggeber am Anfang i.d.R. unvollständig und wenig konkret sind und sich erst im Laufe der Zusammenarbeit entwickeln und konkretisieren. Experte C erläutert hierzu:

> *„Es ist immer eine kontinuierliche Zusammenarbeit, weil die Anforderungen häufig sehr unscharf sind. […] Unsere Stärke ist es, diese Dinge zu verstehen und mit dem Kunden gemeinsam herauszufinden, was er jetzt eigentlich braucht oder was er will. Das impliziert, dass Sie eine hohe Iterationsrate und sehr wenig Vorhersagbarkeit haben."*

4.3.1.4.2 Beurteilung der Endbenutzer-Entwicklung im Allgemeinen

Der Inhalt dieses Kapitels beschreibt die Beurteilung der Interviewpartner in Bezug auf einen Endbenutzer-Entwicklungsansatz im Allgemeinen, d.h. ohne Fokussierung auf mobile Unternehmensapplikationen.

[19] http://www.jetbrains.com/objc zugegriffen am 25.4.2014

Die Antworten der Interviewpartner wurden in die folgenden fünf Kategorien zusammengefasst: 1) Einsatzmöglichkeiten und Vorteile der Endbenutzer-Entwicklung, 2) Erfolgsfaktoren, 3) Risiken und Herausforderungen, 4) Charakteristiken eines Endbenutzers und 5) Änderungen der Arbeitsorganisation.

Einsatzmöglichkeiten und Vorteile der Endbenutzer-Entwicklung

In Bezug auf die Einsatzmöglichkeiten der Endbenutzer-Entwicklung ist ein Großteil der Experten skeptisch gegenüber der Entwicklung komplexerer Applikationen durch Endbenutzer. Sie sehen jedoch Potenzial in der Fokussierung auf ein spezielles, eingeschränktes Anwendungsspektrum (Experten A, E, F, G). Experte A erläutert hierzu:

> *„Deshalb taugt die Idee heute häufig für Einzellösungen oder maximal Abteilungslösungen mit einem überschaubaren Anwendungsspektrum, wo man auch relativ sicher ist. Aber weniger für unternehmensweite, komplexe Lösungen, insbesondere auch für internationale Lösungen“*

Diese Einschätzung deckt sich mit den Aussagen aus der Literatur, dass die Endbenutzer-Entwicklung auf eine spezifische Domäne fokussiert sein sollte (vgl. z.B. (Reppening/Ioannidou 2006, 82)). In diesem Zusammenhang führt Expert E den Aufwand zur Bereitstellung eines zugehörigen Entwicklungswerkzeuges als weiteren Aspekt an. Nach seiner Aussage sollte ein Endbenutzer-Entwicklungswerkzeug ein überschaubares Spektrum an Anwendungsfällen abdecken und nur wenige, ausgewählte Funktionalitäten bereitstellen. Dies reduziert den Aufwand zur Entwicklung des Werkzeuges. Experte E erläutert hierzu:

> *„Je mehr das System können soll, umso aufwändiger wird es. Der Aufwand steigt exponentiell an. Da muss man den gesunden Zwischenweg finden und Elemente integrieren, die immer verwendet werden. Dann gibt es eine Reihe von Sonderfällen, die man in dem Werkzeug auf jeden Fall nicht umsetzen sollte, bspw. wenn diese nur einmal im halben Jahr angewendet werden. Das macht dann der Techniker. Ich würde mich nicht der Illusion der eierlegenden Wollmichsau, die alles kann, hingeben. Es wird immer ein Spezialbereich sein.“*

Experte A empfindet es als Vorteil, wenn sich ein Endbenutzer im Zuge der Endbenutzer-Entwicklung selbst mit den Datenstrukturen einer Applikation auseinandersetzen muss, da dies sein Verständnis verbessert. Ein weiterer Vorteil aus Sicht des Experte B ist der emotionale Aspekt der Motivation. Nach seiner Ansicht können sich die Mitarbeiter im Rahmen der Endbenutzer-Entwicklung selbst kreativ und gestaltend betätigen, was ihre Motivation fördert. Zudem sieht Experte B den Vorteil, dass sich Endbenutzer durch einen solchen Ansatz Applikationen entwickeln können, welche nur ihre benötigen Funktionalitäten besitzen und somit ein optimales Arbeitswerkzeug darstellen. Experte B nennt als einziger den Aspekt der Kosteneinsparung, welcher seiner Ansicht nach durch die Einsparung von professionellen Softwareentwicklern entsteht.

Erfolgsfaktoren

Als Erfolgsfaktor für die Endbenutzer-Entwicklung nennen die Experten G, H und I die Motivation der Endbenutzer, selbst eine Applikation zu entwickeln. Diese muss nach Ansicht der Experten bei mehreren potenziellen Endbenutzern vorhanden sein, damit das Konzept erfolgreich wird. Nach Ansicht von Experte H muss zusätzlich zeitlicher Spielraum sowie eine bestimmte Unternehmenskultur vorhanden sein, welche die spielerische Auseinandersetzung mit dem Thema, ohne Angst vor Fehlern, ermöglicht. Experte H erläutert hierzu:

> *„So etwas geht nicht in einer Phase von extremem Druck oder Stress. Dafür muss ich bereit sein und die Zeit haben, mich hinein zu denken, hinein zu lesen und einfach 1-2 Stunden abzuschalten. Man braucht ein Klima, das auch ein Scheitern erlaubt, damit man sich trauen kann in eine Richtung zu gehen, auch wenn es dann nicht klappt“*.

Ein weiterer wichtiger Punkt war für die Experten C und G, dass das Aufwand-Nutzen-Verhältnis stimmen muss, d.h. dass der erzielte Nutzen den benötigen Aufwand zur Einarbeitung und Bedienung eines zugehörigen Entwicklungswerkzeuges übersteigen muss. Für Experte C ist es zudem wichtig, dass der erzielbare Nutzenvorteil deutlich vorab sichtbar ist, damit ein Endbenutzer sich mit dem Thema beschäftigt. Experte C erläutert hierzu:

> *„Der Endbenutzer muss ganz klar verstehen, was sein Vorteil ist. Er wägt diesen Vorteil immer automatisch mit dem Aufwand, den er damit hat, ab. Dann trifft er eine Entscheidung. […] Es muss den Benefit wirklich geben, d.h. den Trade-Off zwischen Aufwand und Nutzen. Zweites muss er transportiert werden und man braucht noch die positive Erfahrung beim Anwender oder Kunden, dass der Benefit auch wirklich eintritt“*.

Nach Meinung der Experten A, B und I ist eine Schulung interessierter Endbenutzer notwendig, bevor diese einen solchen Ansatz und die zugehörigen Entwicklungswerkzeuge nutzen können. Nach Ansicht von Expert B würden ein Video-Trainings ausreichen.

Nach Einschätzung von Expert A muss es in einem Unternehmen eine bestimmte Personengruppe geben, die diesen Ansatz unterstützt. Sie muss an den Ansatz glauben, Überzeugungsarbeit bei Kritikern leisten und auch das notwendige Budget für die erforderlichen Investitionen bereitstellen. Experte C ist zudem überzeugt davon, dass sich ein solcher Ansatz nur dann etablieren kann, wenn er inkrementell eingeführt wird und sich die Endbenutzer hierdurch Schritt-für-Schritt an die entsprechenden Aktivitäten und Werkzeuge gewöhnen können. Experte A ergänzt zudem, dass das Feedback der Endbenutzer stetig aufgenommen und entsprechend berücksichtigt werden sollte.

Risiken und Herausforderungen

Für die Experten A, B , D und H ist die Einhaltung von Sicherheitsregeln eine zentrale Herausforderung bei der Endbenutzer-Entwicklung. Experte A empfiehlt in diesem Kontext, dass vorab ein detailliertes Berechtigungskonzept erstellt werden sollte. Dieses sollte festlegen, wer welche Daten speichern und manipulieren darf. Experte A erläutert hierzu:

„Der Evergreen ist ja in diesem Umfeld: Wie sensibel sind diese Daten und auf welchen Medien darf ich was abziehen, speichern und manipulieren."

Experte H ergänzt hierzu:

„Applikationen die durch den Endbenutzer entstehen, abzusichern, so dass diese nichts tun, was sie nicht dürfen. Also in irgendeinen Kontext einsperren. Und wenn die Endbenutzer da heraus wollen – natürlich wollen sie das, denn sie wollen auf andere Systeme zugreifen – dann muss man eigentlich einen Schnittstellenvertrag machen, in dem steht, welche Schnittstellen man wo benutzten darf. Man braucht dann explizite Schnittstellen und explizite Berechtigungen – ganz feingranular."

Nach Aussage des Experten D ist es zudem wichtig, dass ein Endbenutzer mit seinen Entwicklungsaktivitäten keinen Schaden anrichten kann. Dies verringert das Risiko für die gesamte Organisation als auch die Akzeptanz des einzelnen Endbenutzers.

Für die Experten B und I stellt die potentielle Vielfallt an individuellen Lösungen ein Risiko dar. Als Grund nennen Sie die Pflege und Wartung der großen Anzahl an Applikationen. Dies würde nach Meinung der Experten eine erhebliche Erhöhung des Aufwands darstellen. Experte I erläutert in diesem Zusammenhang:

„Gehen wir einmal davon aus, dass ein Unternehmen eine solche Idee wirklich umsetzt. Es gibt 20 verschiedene Leute, die sich 20 verschiedene Apps aus der Plattform, die ihnen zur Verfügung steht, generieren. Am Ende hat man 20 unterschiedliche Softwarestücke. Was ist wenn es einmal Probleme gibt? Wenn man Glück hat, dann hat man das so modular aufgebaut, dass man nur das Modul reparieren muss und es dann wieder funktioniert. Aber das Zusammenspiel aller kleinen Funktionalitäten könnte so komplex werden, dass die Wartung eine solchen Systems wahnsinnig schwierig wird".

Experte A nennt als einziger den zusätzlichen Aspekt des Urheberrechts der entwickelten Applikationen, welcher seiner Ansicht nach vorab geklärt werden sollte. Experte A erläutert hierzu:

„Ich unterstelle dass Sie in einem betrieblichen Kontext auch als Endbenutzer den Quellcode in Ihrer Arbeitszeit entwickeln. Damit entstehen die Rechte im Unternehmen und nicht durch das Individuum. Selbst wenn er oder sie das in seiner Freizeit täte, würde das trotzdem durch das zur Verfügung stellen der Instrumente durch das Unternehmen ein Schutzrecht des Unternehmens darstellen. Jetzt kann das Unternehmen aber auch darauf verzichten, weil das so gebräuchlich ist und weil es ja gewünscht ist, dass die Mitarbeiter das machen und sich damit auseinandersetzten".

Charakteristiken eines Endbenutzers

Nach Ansicht des Experten A gibt es den typischen Endbenutzer nicht. Der Experte erklärt, dass Endbenutzer aus unterschiedlichen fachlichen Bereichen und Hierarchieebenen kommen

und sich in ihren Fähigkeiten mit Technologien umzugehen, unterscheiden. Auch die Experten B und E nennen das Kriterium der Technologieaffinität als zentrales Unterscheidungsmerkmal der Endbenutzer. Experte E erläutert in diesem Zusammenhang:

> *„[…] weil es keinen Metatypus Softwareanwender gibt. Wir haben die technikaffinen, die wiederrum unterteilt sein können in: Manche verstehen den Sinn und freuen sich, dass es diese Technik gibt, andere sind Techniknerds […]. Dann gibt es noch einen mittendrin der sagt: ‚Top, dass es das endlich gibt. Das spart mir viel Zeit'. Dann gibt es andere, die technikavers sind und sagen: ‚Das ist etwas Neues. Damit komme ich nicht klar und das will ich nicht lernen.'"*

Für den Experten B sollten gerade die technikaffinen Endbenutzer die Zielgruppe eines Endbenutzer-Entwicklungswerkzeuges sein. Der Experte begründet dies damit, dass technikaverse Endbenutzer die Nutzung eines solchen Werkzeuges als Zusatzlast sehen werden, wohingegen technikaffine Endbenutzer darin eine Möglichkeit sehen könnten, ihre eigenen Arbeitsabläufe zu optimieren.

Aus Sicht der Experten C und H zeichnen sich Endbenutzer vor allem dadurch aus, dass sie oftmals keine konkrete Vorstellung vom Endergebnis einer Applikation haben. In dieser Charakteristik sehen diese Experten auch eine große Herausforderung für einen Endbenutzer-Entwicklungsansatz. Nach Meinung der Experten denken Endbenutzer nur sehr eingeschränkt über typische Umsetzungsdetails nach, wie beispielsweise Fehlerbehandlungen oder detaillierte Abläufe. Nach Ansicht des Experten C benötigen Endbenutzer zunächst konkrete Vorschläge, um eine Entscheidung treffen zu können, was sie gernen möchten. Er ergänzt hierzu, dass die Endbenutzer zwar oftmals keine genaue Vorstellung vom Endergebnis haben, jedoch häufig wissen, wie es nicht sein sollte. Experte C erläutert hierzu:

> *„Dieses Vorstellungsvermögen vom Endergebnis ohne etwas Konkretes in der Hand zu haben, finde ich zumindest bei unseren Kunden sehr selten vor. […] Das eine ist, dass der Endbenutzer nicht so ganz genau weiß, was er will. Er weiß aber sehr häufig ganz genau was er nicht will, weil er Erfahrungen hat, welche Dinge schlecht funktionieren."*

Dies deckt sich mit der Meinung des Experten H, der erläutert, dass ein Endbenutzer i.d.R. nicht alle Möglichkeiten in der Bedienung einer Applikation durchdenkt und daran oftmals auch kein Interesse hat. Experte H schildert in diesem Zusammenhang:

> *„Es gibt einfach ganz viele Dinge, die der Endbenutzer nicht im Kopf hat und nicht betrachtet. […] Ich halte die Gesellschaft und die breite Masse daran für nicht interessiert. ‚Das Ding soll das machen, was ich will. Ich will nicht jeden Fall extra durchdenken'".*

Änderungen der Arbeitsorganisation

In Bezug auf die Arbeitsorganisation war für alle Experten die Rolle der IT-Abteilung bei der Umsetzung eines Endbenutzer-Entwicklungsansatzes wichtig. Die Experten hatten eine einheitliche Meinung darüber, dass die IT-Abteilung das Entwicklungswerkzeug für die

Endbenutzer bereitstellen muss und auch die notwendige Unterstützung leisten sollte. Experte A schlägt vor, dass die IT-Abteilung Steuerungs- und Normierungsaufgaben übernimmt und hierbei die aufgestellten Regeln überwacht und ggf. anpasst. Experte A und I nennen zusätzlich die Schulung der Endbenutzer als Aufgabe der IT-Abteilung. Experte A und H schlagen zudem vor, dass die IT-Abteilung die Entwicklung eines geeigneten Sicherheitskonzeptes übernehmen sollte.

Nach Ansicht der Experten G, F und H sollte die IT-Abteilung die Backend-Integration übernehmen. Darunter verstehen die Experten, dass geeignete Schnittstellen für das Endbenutzer-Entwicklungswerkzeug zu den Backendsystemen einheitlich von der IT-Abteilung bereitgestellt werden. In diesem Fall müsste sich ein Endbenutzer nicht selbst mit den Backendsystemen auseinandersetzen, wofür einem Großteil der Endbenutzer ohnehin das notwendige Wissen fehlt. Experte H nennt hierzu folgendes Beispiel:

> *„Die IT-Abteilung könnte beispielsweise einen Webservice schreiben, mit dem sie eine gewisse Schnittstelle nach außen anbietet, die über ein gewisses Zertifikat nur von dieser einen Endbenutzer-Applikation verwendet werden kann“.*

Experte G erläutert in diesem Kontext:

> *„Wie kommt jetzt das Tool an die Daten aus dem Großrechnersystem? Das funktioniert ja nur, wenn eine IT-Abteilung einen Service bereitstellt wie eine API oder eine Anbindung, die irgendwie mit der App kommunizieren kann sonst funktioniert das alles nicht. Der Vertriebler kann sich jetzt auch nicht hinsetzen und diese Schnittstelle selbst programmieren.“*

Laut Ansicht des Experten B sollten Mitarbeiter der IT-Abteilung die eigentliche Zielgruppe eines Endbenutzer-Entwicklungswerkezuges sein. Damit meint der Experte Mitarbeiter der IT-Abteilung, welche durch ihre Arbeitsumgebung eine hinreichende IT-Affinität besitzen, jedoch den Einarbeitungsaufwand in die entsprechenden Programmiersprachen und Entwicklungswerkzeuge aus dem professionellen Softwareentwicklungsbereich vermeiden möchten. Diese Mitarbeiter könnten Applikationen für eine Gruppe von Mitarbeitern aus anderen Fachabteilungen bereitstellen. Experte B erläutert in diesem Zusammenhang:

> *„Ich kann mir nicht vorstellen, dass jeder Mitarbeiter dies machen möchte. Aber in dem Fall fände ich es eher sinnvoll - um Entwickler zu sparen – dass man dieses Werkzeug nicht jedem Mitarbeiter gibt, sondern nur einen Mitarbeiter aus der IT-Abteilung zuweist, der das für alle macht und sich um den Support kümmert. Dadurch spart man auch Budget ein“.*

Ähnlich argumentieren die Experte H und I, welche auch eine Gruppe ausgewählter Mitarbeiter mit entsprechender Motivation und technischer Affinität für das Thema als Zielgruppe sehen. Diese ausgewählte Mitarbeitergruppe entwickelt benötigte Applikationen und stellt diese anderen Mitarbeitern zur Verfügung. Experte H schildert hierzu:

„Das sind eigentlich Informatiker, also Leute, die keine Berührungsängste haben und sich mit dem System befassen. Es müssen auch nicht unbedingt Informatiker sein, sondern es können auch Personen sein, die mit einem Spieltrieb herangehen und sich fragen, was man mit dem Werkzeug machen kann. Nur so kriegt man das Know-How - außer man baut es gezielt auf. Aber dafür ist das Interesse wahrscheinlich nicht da."

Experte I schildert in diesem Zusammenhang:

„Da man vieles nur ein einziges Mal erstellen muss, würde ich einen Einzigen nehmen, der die Ideen von allen sammelt und diese dann auch umsetzt. Dieser eine kann seine Kenntnisse, die er sich erarbeitet hat, dann auch öfters einsetzten".

4.3.1.4.3 Beurteilung der Endbenutzer-Entwicklung speziell für mobile Unternehmensapplikationen

In Bezug auf die Endbenutzer-Entwicklung speziell für mobile Unternehmensapplikationen sehen die Experten generell kaum Unterschiede zur Endbenutzer-Entwicklung im Allgemeinen. Falls das Entwicklungswerkzeug selbst auf einem mobilen Endgerät ausgeführt werden soll, sehen die Experten vor allem die Einschränkungen mobiler Endgeräte im Vergleich zu Desktop-PCs als Herausforderung. Experte D erläutert in diesem Zusammenhang:

„Der erste Unterschied ist, dass man auf einer mobilen Anwendung weniger Platz hat, wenn es lesbar sein soll. Eine mobile Anwendung ist logischerweise mobil, d.h. ich kann sie mitnehmen, der Desktop dagegen ist stationär. Bei mobilen Anwendungen habe ich Verbindungsbeschränkungen und muss Netz haben, um remote arbeiten zu können. Ich habe natürlich Akku-Probleme und muss ständig dafür sorgen, dass mein Gerät nicht ausgeht. Das habe ich beim Desktop nicht. Ich habe bei regulären mobilen Geräten Probleme bei der Eingabe, wenn ich keine Tastatur oder Ähnliches anschließe. Die Eingabe dauert immer länger und ist immer fehleranfälliger über eine On-Screen-Tastatur, als über eine Hardware-Tastatur. Das bedeutet, dass Schreibtasks oder auch andere Aufgaben, bei denen ich irgendwie mit Text arbeiten muss, auf mobilen Geräten länger dauern. Präzision geht verloren"

Experte A sieht die Heterogenität im Bereich mobiler Endgeräte als eine größere Herausforderung als im Desktop-Bereich, speziell bei der Umsetzung und Einhaltung von Unternehmensstandards. Experte A erläutert in diesem Zusammenhang:

"[...] oder wird eine Strategie wie Bring-your-own-device verfolgt. Das heißt jedes Format, ob Apple, Android/Google oder Blackberry Infrastruktur, welches ein Endnutzer verwendet wird unterstützt. Das bringt natürlich ganz unterschiedliche Herausforderungen mit Pros und Cons mit sich. In der Regel habe ich eine Desktopvariante, bei der eine Infrastruktur mit einem bestimmten Austauschmechanismus festgelegt ist. Im mobilen Umfeld habe ich das in der Regel im Unternehmen und das was für den Endnutzer richtig und wichtig ist, nicht mehr so im Griff. [...] Alle Betriebssysteme wie Android und auch Micro-

soft unterscheiden sich in Ihren Charakteristika. Das bedeutet, dass ich abhängig davon welches Gerät ich verwende, es auch unterschiedlich einsetze und bediene."

Die Experten D und E führen hingegen den i.d.R. geringeren Funktionsumfang und die geringeren Freiheitsgrade bei der Gestaltung mobiler Applikationen als Vorteile gegenüber der Endbenutzer-Entwicklung im Desktop-Bereich an. Den geringeren Funktionsumfang begründen die Experten mit der Fokussierung mobiler Applikationen auf die Funktionalitäten, welche in einer „mobile Situation" benötigt werden. Die geringeren Freiheitsgrade begründen die Experten mit den Vorgaben der Betriebssystem-Hersteller, welche den Gestaltungsspielraum durch die Vorgabe von Gestaltungsrichtlinien einschränken. Experte D schildert in diesem Zusammenhang:

"Ich glaube auch, dass es gerade im mobilen Bereich leichter ist als im Desktopbereich, weil es im Desktopbereich eigentlich keine Regeln gibt, wie eine Anwendung auszusehen hat oder wie etwas dargestellt wird. Auf den mobilen Plattformen gibt es ja sehr strenge Regeln. Man hat auch nur eine gewisse Grundmenge an UI-Elemente und Designguidelines von Android oder iOS, wie etwas auszusehen hat. Das heißt, wenn man auf dieser gemachten Arbeit aufsetzen kann und diese ganzen Ideen aufgreift, so dass es eben tatsächlich wie jede Standard-App aussieht, kann man diese ganzen Eintrittsbarrieren massiv abbauen, so dass die Leute dann damit arbeiten können und sich nicht gleich frustriert wegdrehen."

4.3.1.4.4 Anforderungen an ein Endbenutzer-Entwicklungswerkzeug für mobile Unternehmensapplikationen

Die von den Experten genannten Anforderungen an ein Endbenutzer-Entwicklungswerkzeug für mobile Unternehmensapplikationen können in die folgenden Kategorien unterteilt werden: 1) einfache Bedienung, 2) schnelle Entwicklung, 3) Benutzungsschnittstelle und Interaktionstechnik und 4) Laufzeitumgebung für das Entwicklungswerkzeug.

Einfache Bedienung

Eine wichtige Anforderung für alle Experten ist eine einfache Bedienung des Werkzeuges. Die Experten fordern, dass das Werkzeug intuitiv erlernbar und selbsterklärend in der Bedienung sein sollte. Experte D fordert zusätzlich „Joy-of-Use" bei der Nutzung der Applikation. Für den Experten G sollte das Werkzeug selbst eine mobile Applikation sein und auch so einfach wie eine mobile Applikation bedienbar sein. Unter einer einfachen Bedienung versteht Experte D die Nutzung von üblichen Anzeige- und Bedienelementen, einen geringen Einarbeitungsaufwand sowie eine gute Benutzerunterstützung bei Bedienfehlern. Experte D erläutert in diesem Zusammenhang:

„Es muss ein einfaches Benutzerinterfache haben, wobei einfach auch ein sehr weites Feld ist. Das fängt damit an, dass man jetzt keine neuen Benutzerinterfaces erfindet, sondern das man sich daran orientiert, was es heute schon gibt, um die Benutzer auch nicht zu verschrecken. Das Werkzeug muss clever mit Fehler umgehen, beispielsweise mit Bedienfehlern, aber auch mit Irrtümern, also wenn ich aus Versehen irgendwo hin klicke und dann

alles gelöscht ist. Das heißt, es muss ein cleveres Fehlerhandling haben. Es muss Spaß machen und schnell zu lernen sein.“

Für die Experten A und F muss bei der Gestaltung des Werkzeuges eine Abwägung zwischen der Einfachheit der Bedienung und seinem Funktionsumfang getroffen werden. Nach ihrer Ansicht geht eine einfache Bedienung einher mit einem eingeschränkten Funktionsumfang. Daher muss bei der Gestaltung des Werkzeuges sorgfältig abgewogen werden, welche Anwendungsfälle fokussiert werden sollten und welche nicht. Experte A schildert in diesem Zusammenhang:

„Ich bin bereit eine bestimmte Flexibilität aufzugeben, weil sich ein paar Arten und Weisen durchsetzen. Oder ich entscheide, dass das Werkzeug vielleicht nicht so intuitiv sein muss und das ein bisschen Learning und Ausbildung doch ganz gut ist, wen man dadurch weitere Dinge machen kann“.

Für den Experten H steht eine geeignete Handhabung von Benutzerfehlern im Vordergrund. Darunter versteht der Experte eine Vermeidung von Benutzerfehlern und ggf. eine geeignete Hilfestellung bei Benutzerfehlern. Experte H erläutert hierzu:

„Der typische Endbenutzer ist ein DAU und das ist gut so. Das klingt ein bisschen gehässig, aber der DAU ist ein sehr wichtiges Instrument, um ein System zu designen. Ich stelle mir wirklich den dümmsten anzunehmenden User vor und was er mit einem System machen würde. So muss man herangehen, um ein fool-proof System zu bauen.“

Benutzungsschnittstelle und Interaktionstechnik

In Bezug auf die Benutzungsschnittstelle war dem Großteil der Experten eine übersichtliche Benutzungsschnittstelle wichtig. Um diese Übersichtlichkeit zu erreichen, schlagen die Experten B und F vor, nur die Basisfunktionalitäten anzuzeigen und die restlichen Funktionalitäten über einen Expertenmodus zugänglich zu machen. Experte B erläutert in diesem Zusammenhang:

„Es muss simpel und übersichtlich sein und für jeden nutzbar, der nicht so technisch-affin ist. Jeder der nicht programmieren kann, muss damit etwas anfangen können, weil die meisten Mitarbeiter ja keine Entwickler sind. […] Der Benutzer muss die Möglichkeit haben in den Expertenmodus zu wechseln. Er kann dann für sich selbst entscheiden, welche Funktionen er selbst oft benutzt.“

In Bezug auf die gewünschte Interaktionstechnik waren sich die Experten einig, dass die Endbenutzer keinen Quellcode schreiben sollten. Nach Ansicht der Experten B, D, E und I wäre ein assistentenbasierten Ansatz, d.h. eine formularbasierte Programmierung (siehe Kapitel 2.4.3.1.3) geeignet. Dabei sollen die Endbenutzer Schritt-für-Schritt durch den Entwicklungsprozess geführt werden und geeignete Optionen zur Auswahl angeboten bekommen. Experte G erläutert in diesem Zusammenhang:

„Das ist quasi ein Prozessmodellierungstool. Es gibt verschiedene Schritte. Der Endbenutzer kann sich erst einmal ein Ablaufdiagramm erstellen, indem dann irgendetwas passiert. Beispielsweise gibt es dann einen Knoten mit Eingabe. Der Benutzer kann definieren, welche Art von Eingabe er machen möchte, beispielsweise Vorname und Nachname. Das Tool weiß dann – wenn der Endbenutzer an dieser Stelle in der Anwendung ist, dass es einen Schirm aufbauen muss, wo man Vorname und Nachname eingaben kann […]“

Experte I erläutert hierzu:

„Ich glaube, es ist zu kompliziert, wenn man alle Möglichkeiten hat. Ein Prozess bzw. eine Art Wizard wäre bestimmt nicht schlecht. Es gibt ja beispielsweise auch Wizards von Photoshop, die einen durch die Bildbearbeitung führen. Man kann zuerst ausrichten, dann beschneiden und anschließend Kontraste und Helligkeit anpassen. Eigentlich ist es ja egal in welcher Reihenfolge das gemacht wird. Es ist aber durchaus sinnvoll einen Prozess zu haben, um den Benutzer an die Hand zu nehmen. In unserem Fall muss man erst einmal einen Rahmen schaffen, also: Wie sieht denn die Applikation überhaupt aus? Im einfachsten Fall ist das ein Farbschema. Ich würde mich vom Rahmen nach innen arbeiten.“

Für den Experten G ist es insbesondere wichtig, dass der Endbenutzer keine freingranulare Gestaltung der Benutzungsschnittstelle der zu entwickelnden Applikation durchführen muss. Nach seiner Erfahrung ist beispielsweise das Positionieren von Anzeige- und Bedienelementen oder die Auswahl einer bestimmten Ausprägung eines Textfeldes oder einer Schaltfläche für Endbenutzer tendenziell frustrierend. Experte G erläutert in diesem Zusammenhang:

„Ich glaube, dass man das UI nicht beeinflussen können soll. Weil es komplizierter wird. Man hat dann noch mehr Möglichkeiten Einstellungen vorzunehmen. Es wird komplizierter und ist wahrscheinlich auch unnötig“.

Das Unternehmen von Expert E hat ein Werkzeug entwickelt, mit welchem sich auf Basis von Schablonen mobile Applikationen parametrisieren und anschließend generieren lassen. Bei der Erstellung des Werkzeugs hat der Experte die Erfahrung gemacht, dass die Bereitstellung vieler Wahlmöglichkeiten nachteilig ist. Er propagiert daher eine Fokussierung auf wenige Optionen. Experte E beschreibt seine Grundidee für ein entsprechendes Werkzeug wie folgt:

„Wir haben Templates vorgegeben, so dass die Endbenutzer verschiedene Auswahlmöglichkeiten besitzen, wie beispielsweise ‚Text links‘, ‚Text rechts‘, ‚Bild oben‘ oder ‚Bild unten‘. Das sind fixe Templates. Innerhalb der App haben die Endbenutzer sehr wenige Möglichkeiten. Sie können beispielsweise nicht die Größe der Überschriften oder Textfarben verändern. Dies wird meistens über die Templates oder auch Template-Kombinationen vorgenommen. Man kann ein Hintergrundbild austauschen, aber ansonsten sind die Variationsmöglichkeiten aus einem einfachen Grund sehr begrenzt. Ursprünglich haben ich vor 12 Jahren angefangen ein Content-Management-System für Webseiten zu schreiben. In den ersten Versionen, die es gab, haben die Kunden sehr viele Freiräume bekommen, welche sie auch genutzt haben. Das war nicht gut.“

Schnelle Entwicklung

Für die befragten Experten war die Geschwindigkeit der Applikationsentwicklung ein wichtiger Aspekt. Experte B erklärt, dass die Applikationsentwicklung für Endbenutzer einen Zusatzaufwand darstellt, welcher so gering wie möglich gehalten werden sollte. Die meisten Experten schlagen eine Wiederverwendung von bestimmten Applikationsteilen als Lösungsmöglichkeit vor, um die Entwicklung zu beschleunigen. Experte A nennt eine einfache und schnelle Integration mit Backendsystemen als zusätzliche Lösungsmöglichkeit. Experte B erläutert in diesem Zusammenhang:

> *„Man sollte die benötigten Funktionen schnell in eine Applikation verpacken können und nicht zwei Tage dafür benötigen […] da es wahrscheinlich nicht jedem Mitarbeiter gefallen wird, dass er sich zusätzlich zu seiner eigentlichen Arbeit, mit der er bereits ausgelastet ist, noch um seine Arbeitsmittel kümmern muss“.*

Laufzeitumgebung für das Entwicklungswerkzeug

Experte C wünscht sich, dass das Entwicklungswerkzeug auf dem mobilen Endgerät selbst als mobile Applikation realisiert wird. Er begründet dies damit, dass er häufig unterwegs ist und ihm in der „mobilen Situation“ die entsprechenden Ideen und Anforderungen an eine gewünschte mobile Applikation einfallen. Experte C erläutert in diesem Zusammenhang:

> *„Denn wenn ich für das Design daran gebunden bin, wieder in die Desktopwelt zu gehen, habe ich auch wieder die ganzen Nachteile. Ich muss mich extra hinsetzen und komme an einem Freitag ins Büro, um das zu machen, weil ich es nicht mobil machen kann. Wahrscheinlich bin ich am Freitag nicht bereit ins Büro zu kommen. Der größere Punkt ist aber, dass ich meinen Bedarf dann nicht im Kopf habe. Den Bedarf habe ich ja, wenn ich draußen in der mobilen Situation bin. Dann weiß ich gerade genau was ich will. Bis zum nächsten Freitag habe ich das dann auch häufig schon wieder vergessen. Oder ich müsste mich dann so strukturieren und meine Arbeit entsprechend einteilen, um das dann im Nachgang aufzuarbeiten. Das wäre nicht gut und würde mir nicht gefallen.“*

Für den Experten B ist hingegen wichtig, dass das Werkzeug sowohl für den Desktop-PC, als auch als mobile Applikation zur Verfügung steht. Außerdem möchte der Experte, dass das Werkzeug für mehrere mobile Betriebssysteme zur Verfügung steht.

Gewünschte Funktionalitäten

In Bezug auf die gewünschten Funktionalitäten wurde die Umsetzung der Corporate Identity (CI) häufig genannt. Die CI umfasst beispielsweise die Platzierung eines Unternehmenslogos oder die Umsetzung eines bestimmten Farbschemas für eine mobile Applikation.

Für die Experten D und F ist eine Art „Vorschaufunktion“ wichtig, bei welcher die getätigten Änderungen sofort begutachtet werden können. Experte F spricht in diesem Zusammenhang von einem schnellen „Turn-Around“ von der Entwicklung hin zum Testen. Er erläutert in diesem Zusammenhang:

„Für mich ist wichtig, dass der Turn-Around von der Entwicklung hin zum Testen sehr schnell geht. Wenn ich eine Änderung mache, dann muss ich auf einen Knopf drücken können und dann sofort das Resultat meiner Änderung sehen“.

Experte A wünscht sich zusätzlich eine sogenannte „Sandbox“ Umgebung, in welcher der Endbenutzer seine entwickelte mobile Applikation in einer sicheren Umgebung testen kann. Beispielsweise sollten in einem solchen Testmodus keine Änderungen an den Daten des Backendsystems durchgeführt werden können.

Die Experten B, C, D und H fordern die Existenz von wiederverwendbaren Bausteinen in Form von Templates, Applikationskomponenten oder Bibliotheken. Experte H vergleicht die Existenz von Schablonen mit den Entwurfsvorlagen in der Präsentationssoftware Powerpoint der Firma Microsoft. Experte H erläutert in diesem Zusammenhang:

„Es sollten Vorgaben bzw. Vorschläge dabei sein. Sie kennen das vielleicht von Office-Programmen. Da gibt es Formatvorlagen und verschiedenen Arten von Diagrammen in allen möglichen Farben. Auch bei Powerpoint ist es ja so, das schon verschiedene Möglichkeiten vordesignt sind.“

Für die Experten A und F ist eine Verteilungsfunktion der entwickelten mobilen Unternehmensapplikationen wichtig. Über eine solche Verteilungsfunktion soll das Bereitstellen der entwickelten mobilen Unternehmensapplikationen an andere Mitarbeiter des Unternehmens möglich sein. Die beiden Experten fordern zusätzlich eine Möglichkeit die entwickelten mobilen Applikationen hinsichtlich ihrer Qualität von Dritten prüfen zu lassen. Experte F beschreibt sein gewünschtes Vorgehen, bei welchem zunächst eine Gruppe von Pilotanwendern Zugriff auf die neu entwickelte mobile Applikation bekommen und diese bei positivem Testergebnis an einen größeren Personenkreis verteilt werden kann. Experte F erläutert in diesem Zusammenhang:

„Das Ausrollen der Versionen, die ich erstellt habe, muss gut koordiniert sein. Die Änderungen, die ich ständig mache, möchte ich nicht an alle Benutzer ausrollen, sondern ich möchte eine Pilotrunde mit bestimmten Usern, die die Applikation erst einmal testen, durchführen. Hinterher möchte ich die Möglichkeit haben, die von den Zielusern getestete Applikation, einfach und schnell auszurollen.“

4.3.2 Fazit und Diskussion

Die Befragung II hat verdeutlicht, dass insgesamt ein Potential für die Endbenutzer-Entwicklung mobiler ERP-Applikationen besteht. Die genannten Vorteile eine bessere Umsetzung der Anforderungen, eine besseres Verständnis der Applikationen, eine höhere Motivation der Benutzer durch eigene Gestaltungsmöglichkeiten und Kostenvorteile durch die den Reduzierung des Bedarfes an professionellen Softwareentwicklern. Demzufolge können die aus der Literaturstudie identifizierten Vorteile (vgl. Kapitel 2.4) bestätigt werden.

Um die Vorteile zu erzielen, sollten bei der Werkzeugunterstützung einige Aspekte berücksichtigt werden. Wichtig für ein entsprechendes Entwicklungswerkzeug ist seine Fokussierung auf einen bestimmten Anwendungsbereich und die bewusste Begrenzung seiner Funktionalitäten. Da kein Einarbeitungsaufwand akzeptiert wird und schnelle Ergebnisse erwartet werden, sollte die Benutzungsschnittstelle des Werkzeuges übersichtlich gestaltet und intuitiv in ihrer Bedienung sein. Dies ist notwendig, da von einer generellen Skepsis der Endbenutzer gegenüber der Idee der Endbenutzer-Entwicklung und der Angst vor einem damit einhergehenden Zusatzaufwand ausgegangen werden muss. Dies deckt sich mit dem Ergebnis der Literaturanalyse, nach welcher jeder Mitarbeiter bei der Beurteilung der Endbenutzer-Entwicklung seinen individuellen Nutzen mit seinen individuellen Aufwänden abwägt (siehe Kapitel 2.4.2). Daher sollten auch Schulungen oder das selbstständige Durcharbeiten von Übungsmaterialien zum Erlernen des Werkzeugs keine Voraussetzung zur Nutzung des Werkzeuges sein. Ein Lösungsvorschlag ist die Umsetzung einer formularbasierten Interaktionstechnik, bei welcher der Endbenutzer Schritt-für-Schritt durch den Erstellungsprozess der mobilen ERP-Applikation geführt wird. Das Werkzeug selbst sollte als mobile Applikation realisiert werden und auf unterschiedlichen mobilen Betriebssystemen verfügbar sein. Ebenso sollten die entwickelten mobilen ERP-Applikationen auf unterschiedlichen mobilen Betriebssystemen lauffähig sein.

Auch bei einem intuitiv bedienbaren Entwicklungswerkzeug muss davon ausgegangen, dass nicht jeder Endbenutzer zur Nutzung des Werkzeuges bereit ist. Daher sollte der fokussierte Anwendungsfall sein, dass eine ausgewählte Anwendergruppe mobile ERP-Applikationen entwickelt und diese anschließend für andere Mitarbeiter zur Verfügung gestellt werden können.

Aus organisatorischer Perspektive erscheint es nach der Auswertung der Interviews sinnvoll, wenn die IT-Abteilung die Bereitstellung und Pflege des Entwicklungswerkzeuges übernimmt und entsprechende Unterstützung bei Fragen sowie in Problemsituationen anbietet.

5 Anforderungsermittlung für ein Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen

Im Folgenden sollen aus den bisher gewonnen Erkenntnissen aus der Literaturstudie als auch aus den durchgeführten Befragungen, Anforderungen an das gewünschte Entwicklungswerkzeug für mobile ERP-Applikation abgeleitet werden. Hierzu werden zunächst die theoretischen Grundlagen der Anforderungsermittlung behandelt und anschließend im Rahmen der vorliegenden Aufgabenstellung angewendet.

5.1 Theoretische Grundlagen der Anforderungsermittlung

Eine Softwareapplikation ist erfolgreich, wenn sie den Bedürfnissen ihrer Nutzer sowie ihrer Umgebung gerecht wird. Das Ziel einer *Anforderung* ist es diese Bedürfnisse bzw. Erwartun-

gen zu spezifizieren (Balzert 2009, 455; Ebert 2012, 24). Anforderungen können generell als Konkretisierung von Zielen einer Softwareapplikation verstanden werden (Ebert 2012, 49). Sie dienen als Vorgaben für den Entwurf und die Entwicklung einer Softwareapplikation und stellen gleichzeitig die Validierungs- und Verifikationsbasis dar (Grechenig et al. 2010, 140). In Anlehnung an das Institute of Electrical and Electronics Engineers (IEEE) ist eine Anforderung wie folgt definiert (IEEE 1990, 62):

(1) A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or proposed by a system or system component to satisfy a contract, standard, specification or other formally imposed document.
(3) A documented representation of a condition or capability as in (1) or (2)

Zusätzlich können unterschiedliche Arten von Anforderungen unterschieden werden. Generell wird zwischen funktionalen- und nicht-funktionalen Anforderungen unterschieden (vgl. z.B. (Sommerville 2011, 84 ff.). *Funktionale Anforderungen* spezifizieren den Funktionsumfang einer Softwareapplikation und legen hierdurch die „bereitzustellenden Funktion oder den bereitzustellenden Service fest“ (Balzert 2009, 456). Dabei bezieht sich der Funktionsumfang auf die Interaktion mit Nutzern und anderen Systemen als auch auf die interne Datenverarbeitung (Pohl/Rupp 2011, 7 f.). Beispielsweise über funktionale Anforderungen festgelegt, wie eine Softwareapplikation auf bestimmte Eingaben und in bestimmten Situationen reagieren soll (Sommerville 2011, 84 f.). Neben der Spezifikation des Funktionsumfangs müssen i.d.R. bestimmte Randbedingungen (engl. constraints) für Softwareapplikationen festgelegt werden (Sommerville 2011, 85). Dieses werden allgemein unter dem Begriff *nicht-funktionale Anforderungen* zusammengefasst (Sommerville 2011, 85). Grund für nicht-funktionale Anforderung sind bspw. Budgetgrenzen, die gewünschte Interoperabilität mit anderen Softwareapplikationen und Hardwaresystemen oder regulatorische Vorgaben (Sommerville 2011, 87 f.). Beispiele für nicht-funktionale Anforderungen sind Anforderungen an das Antwortverhalten, die Skalierbarkeit, die Erweiterbarkeit oder die Benutzungsfreundlichkeit einer Softwareapplikation (Pohl/Rupp 2011, 8; Sommerville 2011, 88). Aber auch sonstige Lieferbestandteile wie Handbücher oder Trainingsunterlagen sowie vertragliche Anforderungen wie bspw. Meilensteine und etwaige Vertragsstrafen werden zu den nicht-funktionalen Anforderungen gezählt (Rupp 2007, 15 ff.).

Da häufig zu viele Anforderungen an eine Softwareapplikation bestehen, ist die Priorisierung der identifizierten Anforderungen wichtig (Robertson/Robertson 2006, 333 ff.). Durch eine solche Priorisierung ist es möglich zu entscheiden, welche Anforderung in welcher Version der Softwareapplikation umgesetzt werden soll (Robertson/Robertson 2006, 333). In der vorliegenden Arbeit wird lediglich eine rudimentäre Anforderungspriorisierung durchgeführt. Hierbei wird zwischen zwingend notwendigen und gewünschten Anforderungen unterschieden. Zwingend notwendige Anforderungen werden durch die Verwendung von „muss“ in der Anforderungsbeschreibung spezifiziert. Hingegen werden gewünschte Anforderungen durch die Verwendung von „sollte“ in der Anforderungsbeschreibung gekennzeichnet.

Im Bereich der Softwareentwicklung wird die Anforderungsermittlung dem Themengebiet *Anforderungsmanagement* (engl. requirements engineering) zugeordnet. Nach Ebert (2012) umfasst das Anforderungsmanagement alle Aktivitäten zur „Ermittlung, Dokumentation, Analyse, Prüfung, Abstimmung und Verwaltung unter kundenorientieren, technischen und wirtschaftlichen Vorgaben“ (Ebert 2012, 35). Abbildung 5-1 illustriert die beschriebenen Aktivitäten des Anforderungsmanagements.

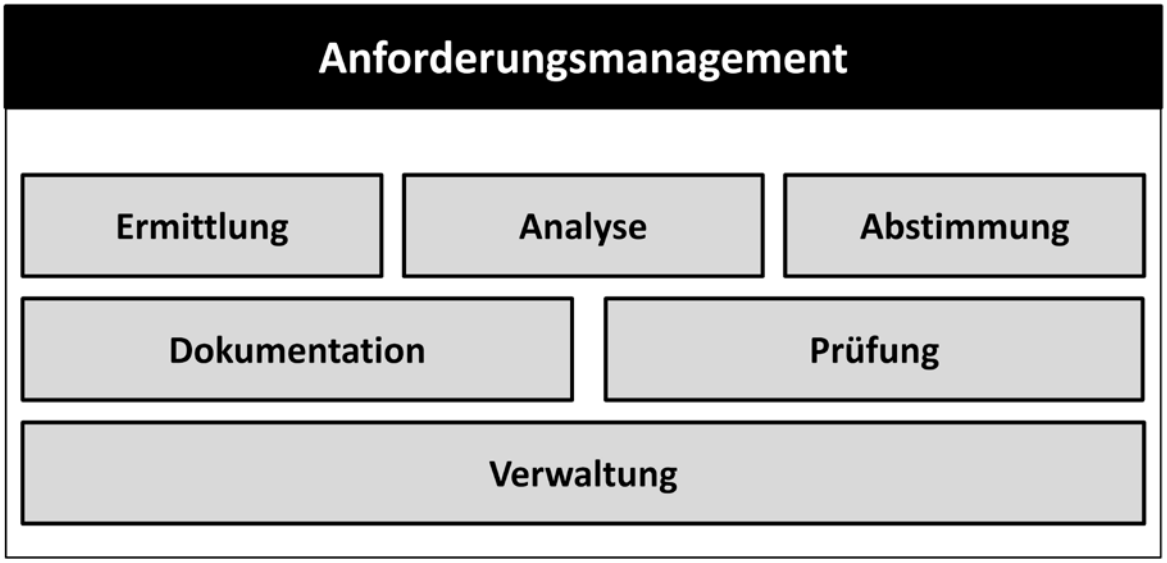

Abbildung 5-1: Aktivitäten des Anforderungsmanagements
Quelle: Eigene Darstellung in Anlehnung an (Ebert 2012, 35)

Generell wird die Erhebung geeigneter Anforderungen als Herausforderung empfunden. Es besteht auch Konsens darüber, dass die Anforderungsermittlung kein streng sequentielles Vorgehen sein kein. Nach Wiegers (2003, xviii) ist es das Ziel zunächst qualitativ gute – nicht perfekte – Anforderungen zu ermitteln, die einen Projektstart mit einem akzeptablen Risiko erlauben.

Nach Ebert (2012) erfolgt die Ermittlung von Anforderungen in zehn Schritten (Ebert 2012, 58 ff.):

(1) Identifizierung der relevanten Interessengruppen

(2) Identifizierung der kritischen Erfolgsfaktoren

(3) Abstimmung von Vision, Umfang und Kontext

(4) Methodische Ermittlung der Anforderungen

(5) Dokumentation und Strukturierung der Anforderungen

(6) Modellierung der Anforderungen

(7) Analyse der Anforderungen

(8) Prüfung der Anforderungen

(9) Priorisierung der Anforderungen

(10) Entscheidung über Annahme der Anforderungen

In Bezug auf die relevanten Interessengruppen (engl. stakeholder) werden oftmals Kunden, Lieferanten, Vertriebspartner und firmeninterne Gruppen unterschieden (Ebert 2012, 58). Die Identifizierung der relevanten Interessengruppen ist entscheidend, da von diesen die Anforderung an die Softwareapplikation stammen (Balzert 2009, 504). Bei der Identifizierung der kritischen Erfolgsfaktoren sollen die Ziele und Rahmenbedingungen der Interessensgruppen verstanden werden. Anschließend kann analysiert werden, wie die zu entwickelnde Softwareapplikation diese Faktoren positiv beeinflussen kann. Bei der Abstimmung von Vision, Umfang und Kontext werden die Ziele der Softwareapplikation und deren Einschränkungen mit dem Auftraggeber festgelegt. Zudem werden die Rahmenbedingungen wie bspw. das Projektbudget und der zeitliche Rahmen abgestimmt. Bei der methodische Entwicklung der Anforderungen können unterschiedlichen Quellen zum Einsatz kommen. Beispiele für mögliche Quellen sind: Marktstudien, Internetrecherche, Kundeninterviews oder bestehenden Lasten- und Pflichtenhefte. Bei der Dokumentation und Strukturierung der Anforderungen werden diese in einem strukturierten Format festgehalten und vorzugsweise elektronisch gespeichert. Ziel der Modellierung der Anforderungen ist es Zusammenhänge zwischen den Anforderungen und deren Einflüsse aus der Umgebung zu erkennen. Bei der Analyse der Anforderungen werden diese Zusammenhänge und gegenseitigen Einflüsse sowie die geschätzten Aufwände der Anforderungsimplementierung untersucht. Schließlich werden die dokumentierten Anforderungen hinsichtlich ihrer Vollständigkeit, Widerspruchsfreiheit, Validierbarkeit, Erweiterbarkeit, Konsistenz und Verständlichkeit geprüft. In den letzten beiden Schritten wird über die Annahme oder Ablehnung der Anforderungen und ggf. auch über ihre Umsetzungsreihenfolge entschieden (Ebert 2012, 69 f.).

5.2 Anforderungen an ein Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen

Im Folgenden werden aus den Ergebnissen der Literaturstudie sowie der beiden Befragungen Anforderungen an das Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen abgeleitet. Die Einteilung der Anforderungen erfolgt in Anforderungen an das Entwicklungswerkzeug und Anforderungen an die damit entwickelten mobilen ERP-Applikationen.

5.2.1 Anforderungen an die entwickelten mobilen ERP-Applikationen

Aus den beiden Befragungen ging hervor, dass vor allem die Heterogenität mobiler Betriebssystemen derzeit eine große Herausforderung darstellt. Zudem haben die Literaturstudie sowie die Befragungen eine aktuelle Dominanz der mobilen Betriebssysteme iOS und Android

gezeigt. Daher sollten die entwickelten mobilen ERP-Applikationen analog zum Entwicklungswerkzeug (siehe Anforderung 11) zumindest auf diesen beiden mobilen Betriebssystemen lauffähig sein. Hieraus lässt sich die Anforderung 1 wie folgt formulieren:

Anforderung 1 (A1): Die entwickelten mobilen ERP-Applikationen müssen auf den beiden mobilen Betriebssystemen Android und iOS lauffähig sein.

Die innerhalb der mobilen SAP-ERP-Applikationen angezeigten und änderbaren Daten müssen aus einem SAP-ERP-System stammen. Kapitel 2.1.5.2.1 hat BAPIs als zugehörige Programmierschnittstellen für die Datenabfrage und –änderung in SAP-ERP-Systemen vorgestellt. Anforderung 2 lässt sich demzufolge wie folgt formulieren:

Anforderung 2 (A2): Die entwickelten mobilen ERP-Applikationen müssen auf die Daten eines SAP-ERP-Systems über BAPI-Schnittstellen zugreifen.

5.2.2 Anforderungen an das Entwicklungswerkzeug

Im Folgenden werden die Anforderungen an das Entwicklungswerkzeug zur Erstellung mobiler ERP-Applikationen beschreiben.

Anforderung 3 bezieht sich auf den Funktionsumfang des Entwicklungswerkzeuges. Die Literaturstudie hat gezeigt, dass ein Erfolgskriterium für ein Endbenutzer-Entwicklungswerkzeug die Fokussierung auf eine bestimmte Domäne ist (vgl. z.B. (Reppening/Ioannidou 2006, 82)). Auch die Befragung II hat gezeigt, dass eine Fokussierung auf einen bestimmten Anwendungsbereich gefordert wird. Die vorliegende Domäne sind mobile ERP-Applikationen. Da dies aufgrund des hohen Funktionsumfanges von ERP-Systemen keine ausreichende Einschränkung darstellt, muss eine weitere Eingrenzung erfolgen. Die zusätzliche Einschränkung wird unter Berücksichtigung der identifizierten Charakteristiken existierender mobiler ERP-Applikationen vorgenommen. Charakteristik C3 unterscheidet zwischen unterschiedlichen Applikationsschwerpunkten unterschieden. Aufgrund der vergleichsweise eingeschränkten Funktionalität soll eine Fokussierung auf Produktivitätsapplikationen erfolgen. Diese sollen sich gemäß Charakteristik C8 auf ausgewählte Business Objekt Typen fokussieren. Für die gewählten Business Objekt Typen soll die Auflistung der existierenden Business Objekte sowie das Anzeigen, Erzeugen, Ändern und Löschen ausgewählter BOs unterstützt werden. Hieraus lässt sich Anforderung 3 wie folgt formulieren:

Anforderung 3 (A3): Das Entwicklungswerkzeug muss die Erstellung von Produktivitätsapplikationen unterstützen mit den folgenden Funktionalitäten für die bereitgestellten Business Objekt Typen bzw. Business Objekte unterstützen:

- Auflistung aller verfügbaren Business Objekt Typen
- Auflistung aller Business Objekte eines selektierten Business Objekt Typs
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten Business Objekts
- Erzeugung eines neuen Business Objekts
- Änderung eines existierenden Business Objekts
- Löschung eines existierenden Business Objekts

Aus Befragung II ging hervor, dass Endbenutzer tendenziell keine Einarbeitung in ein entsprechendes Entwicklungswerkzeug für mobile ERP-Applikationen akzeptieren. Daraus lässt sich die Anforderung 4 wie folgt formulieren:

Anforderung 4 (A4): Die Erstellung mobile ERP-Applikationen mit Hilfe des Entwicklungswerkzeuges muss ohne Trainingsaufwand möglich sein.

Aus Befragung II ging weiterhin hervor, dass Endbenutzer eine schrittweise Führung durch den Entwicklungsvorgang benötigen. Dabei sollen die möglichen Optionen angeboten und ausgewählt werden können. Daraus lässt sich Anforderung 6 ableiten:

Anforderung 5 (A5): Das Entwicklungswerkzeug muss den Endbenutzer schrittweise durch den Entwicklungsvorgang führen.

Aus beiden Befragungen ging hervor, dass die Unterstützung der Corporate Identiy von Unternehmen für die mobilen ERP-Applikationen wichtig ist. Dabei sind insbesondere die Platzierung von Unternehmenslogos und die Spezifikation von Farbschemas notwendig. Daraus lässt sich folgende Anforderung formulieren:

Anforderung 6 (A6): Das Entwicklungswerkzeug muss die Auswahl unterschiedlicher Layoutschablonen zur Umsetzung der Corporate Identity anbieten.

In Bezug auf die Entwicklungsgeschwindigkeit hat Befragung II gezeigt, dass eine schnelle Entwicklung der mobilen ERP-Applikationen notwendig ist, um das Interesse und die Motivation potenzieller Endbenutzer zu wecken bzw. aufrecht zu erhalten. Um dieses Ziel zu konkretisieren wird ein konkreter Zeitrahmen von 15 Minuten als Maximum für den Erstellungsprozess festgelegt. Daraus lässt sich Anforderung 7 wie folgt formulieren:

Anforderung 7 (A7): Der Entwicklungsvorgang einer mobilen ERP-Applikation darf maximal 15 Minuten betragen.

Aus Charakteristik C1 geht hervor, dass die Daten und Funktionalitäten aus einem ERP-System stammen und über bereitgestellte Programmierschnittstellen auf sie zugegriffen wird. Im Falle eines SAP-ERP-Systems erfolgt der Zugriff über BAPIs. Aus der Befragung II ging hervor, dass der Zugriff auf die Funktionalitäten des Backendsystems eine potenzielle Herausforderung für Endbenutzer darstellt. Zudem muss aus Sicherheitsgründen gewährleistet sein, dass Endbenutzer an dieser Stelle keine Schäden am ERP-System verursachen können. Aus diesem Grund soll die Konfiguration und der Zugriff auf die BAPIs des SAP-ERP-Systems durch das Entwicklungswerkzeug ohne zusätzlichen Konfigurations- und Entwicklungsaufwand des Endbenutzers erfolgen. Daraus lässt sich folgende Anforderung formulieren:

Anforderung 8 (A8): Das Entwicklungswerkzeug bietet keine Konfigurationsmöglichkeit für die Verbindung zu einem SAP-ERP-System für den Endbenutzer.

Aus Charakteristik C11 geht hervor, dass existierende mobile ERP-Applikationen aktuell durch ähnliche Benutzungsschnittstellen realisiert sind. Aus Befragung II geht hervor, dass die Gestaltungsmöglichkeiten der Benutzungsschnittstellen eingeschränkt werden sollen, um Fehlerquellen zu vermeiden. Zudem haben beide Befragungen ergeben dass die Wiederverwendung parametrisierbarer Bausteine für die Entwicklung erwünscht ist. Demzufolge lassen sich die Anforderungen 9 und 10 wie folgt formulieren:

Anforderung 9 (A9): Das Entwicklungswerkzeug bietet nur eingeschränkte Möglichkeiten zur Gestaltung der Benutzungsschnittstelle einzelner Bildschirmmasken einer mobilen ERP-Applikation. Beispielsweise wird das Hinzufügen oder Positionieren einzelner Anzeige- und Bedienelemente nicht unterstützt.

Anforderung 10 (A10): Das Entwicklungswerkzeug stellt wiederverwendbare Schablonen für die Umsetzung der Benutzungsschnittstellen für die unter Anforderung 3 beschriebenen Funktionalitäten bereit.

Ein Ergebnis der Befragung II war, dass die gewünschte Laufzeitumgebung das mobile Endgerät selbst sein sollte. Da in dieser Arbeit das Smartphone als mobile Endgeräteklasse fokussiert wird, sollte das Entwicklungswerkzeug für das Smartphone konzipiert werden. Aufgrund der Einschränkungen von Smartphones, u.a. hinsichtlich ihrer kleinen Displaygröße und eingeschränkten Eingabemöglichkeiten, ist ein Umsetzungserfolg jedoch ungewiss. Daher wird die zugehörige Anforderung bewusst nicht als notwendige Anforderung, sondern stattdessen als gewünschte Anforderung charakterisiert. Aufgrund der aktuellen Dominanz der mobilen Betriebssysteme Android und iOS sollte das Entwicklungswerkzeug zumindest auf diesen beiden mobilen Betriebssystemen lauffähig sein. Die zugehörige Anforderung wird wie folgt formuliert:

Anforderung 11 (A11): Das Entwicklungswerkzeug sollte auf einem Smartphone genutzt werden und auf den beiden mobilen Betriebssystemen Android und iOS lauffähig sein.

Die Literaturstudie hat gezeigt, dass sich die Marktverteilung der mobilen Betriebssysteme in den letzten Jahren dynamisch entwickelt hat. Auch die aktuelle Dominanz der mobilen Betriebssysteme iOS und Android kann sich zukünftig ändern. Daher sollte das Entwicklungswerkzeug insofern erweiterbar sein, als dass zukünftig auch mobile ERP-Applikationen für andere mobile Betriebssysteme unterstützt werden können. Demzufolge lässt sich Anforderung 12 wie folgt formulieren:

Anforderung 12 (A12): Das Entwicklungswerkzeug sollte erweiterbar sein, so dass bei Bedarf mobile ERP-Applikationen neben iOS und Android auch für weitere mobile Betriebssysteme entwickelt werden können.

Befragung II kam zu dem Ergebnis, dass es vermutlich nicht möglich sein wird, alle Mitarbeiter für die eigene Entwicklung mobiler ERP-Applikationen zu motivieren. Aus diesem Grund ist eine Entwicklung mobiler ERP-Applikationen durch bestimmte Mitarbeiter wahrscheinlich, welche diese anschließend ihren Kollegen zur Verfügung stellen. Auf Basis dieser Erkenntnis kann die Anforderung 13 wie folgt formuliert werden:

Anforderung 13 (A13): Das Entwicklungswerkzeug muss die Verteilung entwickelter mobiler ERP-Applikationen an weitere Mitarbeiter unterstützen.

5.3 Fazit und Diskussion

Insgesamt wurden 11 Anforderungen an das Entwicklungswerkzeug selbst, als auch 2 Anforderungen an die damit zu entwickelnden mobilen ERP-Applikationen abgeleitet. Bei der Anforderungsermittlung konnten jedoch nicht alle aus der Literaturanalyse und den Expertenbefragungen identifizierten Forderungen und Wünsche berücksichtigt werden. Die Hauptanforderung des Werkzeuges ist die einfache Erlernbarkeit und intuitive Bedienung. Hierdurch sollen schnelle Resultate erzielt werden können und die Endbenutzer hierdurch vom Nutzen des Werkzeuges überzeugt werden. Ziel ist es hierbei, den Aufwand für die Endbenutzer zu reduzieren und den Nutzen zu demonstrieren. Damit soll das zentrale Erfolgskriterium, die Motivation der Endbenutzer zur Nutzung des Werkzeuges (siehe Kapitel 2.4.2), gesteigert werden. Die einfache Erlernbarkeit und intuitive Bedienung soll durch eine Fokussierung des Werkzeuges auf einen bestimmten Anwendungstyp sowie einen formularbasierten, inkrementellen Entwicklungsansatz verwirklicht werden.

In den Anforderungen wurde bewusst auf einige Aspekte verzichtet, um den Funktionsumfang zunächst auf das Wesentliche zu beschränken. Nicht berücksichtigt wurde beispielsweise der Aspekt der Bereitstellung von Datenservices, da dies bspw. Schwerpunkt des SAP-

Produktes NetWeaver Gateway ist. Auch ein Expertenmodus für fortgeschrittenere Endbenutzer ist aktuell nicht Teil der Anforderungen. Außerdem wurde der von einem Experten geforderte Testbetrieb der entwickelten Applikation in einer „Sandbox“-Umgebung aktuell nicht berücksichtigt. Teilweise sind einige der genannten Anforderungen auch konfliktär zueinander. So steht die Anforderungen nach einem Drag & Drop von Anzeige- und Bedienelementen bei der Gestaltung von Benutzungsschnittstellen im Konflikt zu einer minimalistischen Gestaltungsfreiheit bei Benutzungsschnittstellen. Aufgrund der übergeordneten Ziele des Entwicklungswerkzeuges wurde die minimalistische Gestaltungsfreiheit als Anforderung aufgenommen, um Fehler bei der Entwicklung zu vermeiden und die Entwicklungskomplexität zu verringern.

Ein weiterer Schwerpunkt der ermittelten Anforderungen ist die Verfügbarkeit des Werkzeuges für mehrere mobile Betriebssysteme. Der Grund hierfür ist die aktuelle Marktdynamik und die unsichere Marktentwicklung für mobile Endgeräte und der zugehörigen mobile Betriebssysteme. Zumindest auf den beiden am weitesten verbreiteten mobilen Betriebssystemen, iOS und Android, sollte sowohl das Entwicklungswerkzeug, als auch die damit entwickelten mobilen ERP-Applikationen lauffähig sein.

6 Untersuchung ausgewählter Entwicklungswerkzeuge

Im Folgenden werden existierende Entwicklungswerkzeuge für mobile Applikationen untersucht. Hierbei werden zum einen Werkzeuge zur Entwicklung mobiler Unternehmensapplikationen analysiert. Diese haben oftmals einen hohen Funktionsumfang und sind teilweise Bestandteil einer mobilen Unternehmensplattform. Zum anderen werden Endbenutzer-Entwicklungswerkzeuge für mobile Applikationen untersucht. Deren Anwendungsgebiet sind oftmals Anwendungsszenarien aus dem privaten Umfeld. Ziel der Untersuchung ist es, den Funktionsumfang sowie den Entwicklungsprozess mit diesen Werkzeugen zu analysieren. Dabei sollen Herausforderungen bei der Nutzung dieser Werkzeuge für die fokussierten Entwicklungsaktivitäten in dieser Arbeit aufgezeigt werden. Zudem sollen Ideen für die Umsetzung der Anforderungen des eigenen Entwicklungswerkzeuges gesammelt werden. Insgesamt liegt der Schwerpunkt der Untersuchung auf den folgenden Fragestellungen:

- Welche Implementierungsvarianten werden unterstützt? (hybrid, nativ, etc.)
- Welche Laufzeitumgebung ist für das Werkzeug erforderlich?
- Wie erfolgt die Integration mit dem ERP-System?
- Wie ist der Entwicklungsprozess mit dem Werkzeug gestaltet?
- Welche Interaktionstechniken werden bei der Entwicklung eingesetzt?
- Mit welchem Einarbeitungsaufwand ist zu rechnen?

6.1 Entwicklungswerkzeuge für mobile Unternehmensapplikationen

6.1.1 Marktüberblick

Nach Gartner (2013, 1 ff.) gibt es drei verschiedene Arten von Werkzeugen für die Entwicklung mobiler Unternehmensapplikationen: native-, Web- und spezialisierte Entwicklungswerkzeuge. Gartner fasst diese drei Typen unter dem Begriff „Mobile Application Development Platform“ (MADP) zusammen. Im Folgenden werden die Charakteristiken der genannten Arten von Entwicklungswerkzeugen erläutert:

Native-Entwicklungswerkzeuge

Native Entwicklungswerkzeuge ermöglichen die Entwicklung nativer Applikationen. Native Applikationen sind auf ein spezielles mobiles Betriebssystem, wie beispielsweise iOS oder Android, zugeschnitten und auch nur auf diesem mobilen Betriebssystem lauffähig (siehe Kapitel 3.2). Native Entwicklungswerkzeuge werden gewöhnlich vom Hersteller des zugehörigen mobilen Betriebssystems bereitgestellt. Beispiele für native Entwicklungswerkzeuge sind XCode von Apple zur Entwicklung von iOS Applikationen (Homann et al. 2013b, 395 ff.) oder das Android Development Toolkit von der Open Handset Alliance zur Entwicklung von Android Applikationen (Homann et al. 2013b, 359 ff.).

Web-Entwicklungswerkzeuge

Web-Entwicklungswerkzeuge ermöglichen die Entwicklung von Web-Applikationen. Web-Applikationen werden in einem Web-Browser ausgeführt und sind auf unterschiedlichen mobilen Betriebssystemen lauffähig (siehe Kapitel 3.2). Web-Applikationen werden in gängigen HTML-Editoren unter Nutzung von speziellen HTML5-Frameworks für mobile Applikationen wie bspw. jQT, jQuery-Mobile und Sencha Touch (Bai 2011, 9 ff.) entwickelt.

Spezialisierte-Entwicklungswerkzeuge

Spezialisierte-Entwicklungswerkzeuge stellen oftmals die Möglichkeit zur Verfügung, sowohl native Applikationen als auch Web-Applikationen zu implementieren. Sie bieten i.d.R. eine Reihe von zusätzlichen Funktionalitäten, um die Entwicklung mobiler Applikationen zu vereinfachen. Ein Beispiel ist Adobe PhoneGap[20], mit welchem Web-Applikationen in native Applikationen transformiert werden können. Zudem existieren umfangreichere spezialisierte Entwicklungswerkzeuge wie Titanium[21] von Appcelator, oder die Mobile Application Development Platform[22] von DSI (Gartner 2013). Diese Entwicklungswerkzeuge besitzen u.a. grafische Modellierungswerkzeuge, einen Quellcode-Editor, Testwerkzeuge und unterschiedliche Assistenten zur komfortablen Bewältigung wiederkehrender Aufgabenstellungen.

Unter den spezialisierten Entwicklungswerkzeugen gibt es Werkzeuge, welche als Teil einer *mobilen Unternehmensplattform* ausgeliefert werden (siehe Kapitel 3.2). Eine solche Platt-

[20] http://phonegap.com, zugegriffen am 13.04.2014
[21] http://www.appcelerator.com/titanium, zugegriffen am 13.04.2014
[22] http://www.dsiglobal.com/mobile-platform, zugegriffen am 13.04.2014

form beinhaltet neben einem Entwicklungswerkzeug u.a. auch einen speziellen Middleware-Server, welcher die Kommunikation zwischen den mobilen Applikationen und den Backendsystemen steuert (Homann et al. 2014a, 33). Die zugehörigen Entwicklungswerkzeuge sind in diesen Fällen speziell auf den jeweiligen Middleware-Server abgestimmt. Beispielsweise ist es möglich mobile Applikationen über die Entwicklungsumgebung in den Middleware-Server zu laden und anschließend zu testen. Beispiele für spezialisierte Entwicklungswerkzeuge als Teil einer mobilen Unternehmensplattform sind das SAP Mobile Workspace der SAP Mobile Platform[23] oder das IBM Worklight Studio aus IBM Worklight[24].

Gartner hat in seiner Untersuchung 22 verschiedene MADPs untersucht und bewertet. Die Bewertung erfolgte dabei auf Basis von zwei Dimensionen: 1) die Fähigkeit zur Umsetzung (engl. ability to execute) und 2) die Vollständigkeit der Vision (engl. completeness of vision). Die Fähigkeit zur Umsetzung wird u.a. anhand des Produktumfangs, der finanziellen Stabilität des Anbieters, der Qualität der zugehörigen Marketingmaßnahmen und der evtl. bereits vorhandenen Kundenerfahrungen mit dem Produkt bewertet. Zur Bewertung der Vollständigkeit der Vision werden u.a. Kriterien wie das Marktverständnis des Anbieters, das Geschäftsmodell sowie die Vertriebs- und Marketingstrategie des Anbieters. Abbildung 6-1 illustriert das Ergebnis der Bewertung.

[23] http://www.sapmobile-platform.com, zugegriffen am 13.04.2014
[24] http://www-03.ibm.com/software/products/en/worklight, zugegriffen am 13.04.2014

Abbildung 6-1: Magic Quadrant for Mobile Application Development Platforms
Quelle:(Gartner 2013, 4)

Für die Untersuchung in dieser Arbeit waren insbesondere solche Entwicklungswerkzeuge interessant, welche bereits spezielle Funktionalitäten zur Anbindung an ein ERP-System mitbringen. Ein weiterer Aspekt für die Untersuchung eines Werkzeuges dessen Verfügbarkeit. In einigen Fällen wäre zunächst die Beschaffung von Softwarelizenzen notwendig gewesen, um das jeweilige Entwicklungswerkzeug installieren und testen zu können. Aufgrund der Einbettung der Dissertation in das Lehrstuhlprojekt „SAP University Competence Center“ standen Lizenzen für die entsprechenden Produkte aus dem Portfolio der SAP zur Verfügung. Aufgrund der guten Positionierung der SAP Produkte in der genannten Gartner-Studie, der verfügbaren Integrationsmöglichkeiten mit dem SAP-ERP-System sowie der uneingeschränkten Verfügbarkeit der zugehörigen Softwarepakete wurden die Produkte der SAP AG als Untersuchungsgegenstand gewählt.

Die SAP AG ist bereits seit einigen Jahren im Umfeld mobiler ERP-Applikationen aktiv. In diesem Zeitraum sind einige Produktangebote entstanden. Ältere Angebote sind bspw. die WebSAPConsole, SAP NetWeaver Mobile oder Mobile Web Dynpro Online (Beckert et al. 2012, 109 ff.). In jüngster Vergangenheit hat SAP durch die Übernahme der beiden Unternehmen Sybase und Syclo sein Lösungsportfolio im Bereich mobiler Applikationen vergrö-

ßert. Die entsprechenden Lösungen von Sybase und Syclo wurden anschließend in einer einheitlichen MEAP, der sogenannten *SAP Mobile Platform*, konsolidiert. Das primäre Entwicklungswerkzeug der SAP Mobile Platform ist der sogenannte SAP Mobile Workspace. Ein weiteres Entwicklungswerkzeug für mobile SAP-ERP-Applikationen ist das sogenannte *SAP NetWeaver Gateway-Plug-In*. Dieses nutzt eine weitere spezielle Middleware-Plattform, den sogenannten SAP NetWeaver Gateway, um auf die Daten und Funktionalitäten des SAP-ERP-Systems zuzugreifen. Zudem hat SAP mit dem *AppBuilder* ein Webbrowser-basiertes Entwicklungswerkzeug im Portfolio, dessen Einsatz keine vorherige Installation und Konfiguration erfordert. Aufgrund der unterschiedlichen Entwicklungsansätze der drei genannten Entwicklungswerkzeuge stellen diese eine geeignete Auswahl dar, um den aktuellen Stand bei der Entwicklung mobiler ERP-Applikationen aus der Perspektive existierender Entwicklungswerkzeuge zu untersuchen. In den folgenden drei Kapiteln werden der SAP Mobile Workspace, das SAP NetWeaver Gateway-Plug-In, als auch der SAP AppBuilder hinsichtlich der zu Beginn des Kapitels genannten Fragestellungen untersucht.

6.1.2 Entwicklung mobiler SAP-ERP-Applikationen mit dem SAP Mobile Workspace

Der SAP Mobile Workspace ist ein auf der populären Entwicklungsumgebung Eclipse[25] basiertes Entwicklungswerkzeug der SAP Mobile Platform (SMP) (Homann et al. 2013b, 167). Für die Untersuchung wurde die zum Untersuchungszeitpunkt aktuellste Version 2.3 verwendet. In dieser Version wird die Erstellung von Hybrid- als auch von nativen Applikationen unterstützt. Während bei Hybrid-Applikationen der gesamte Entwicklungsprozess unterstützt wird, wird bei nativen Applikationen hingegen nur die Entwicklung der Datenschnittstelle für den Zugriff auf das SAP-ERP-System unterstützt. Die zugehörige Benutzungsschnittstelle wird hingegen in den jeweiligen nativen Entwicklungswerkzeuge, beispielsweise XCode für iOS-Applikationen, entwickelt. Für die Entwicklung von Hybrid-Applikationen steht eine zugehörige Container-Applikation in den jeweiligen App Stores kostenlos zu Verfügung. In der aktuellen Version 2.3 werden die mobilen Betriebssysteme iOS, Android, Blackberry OS und Windows Mobile von der SMP unterstützt.

Zur Integration mit dem SAP-ERP-System wird eine spezielle Datenschnittstelle entwickelt. Kernelemente dieser Datenschnittstelle sind sogenannte Mobile Business Objects (MBO). Die Datenschnittstelle eines MBO von der mobilen Applikation unabhängig vom verknüpften Backendsystem immer über die gleiche Programmierschnittstelle aufgerufen. Hierzu wird gemäß der Konzepte der Objektorientierten Programmierung ein Objekt mit Methoden für das Lesen, Erzeugen, Ändern und Löschen von Objektinstanzen erstellt. Durch diese zusätzliche Abstraktionsebene zwischen der mobilen Applikation und der Programmierschnittstelle des Backendsystems muss sich der Entwickler einer mobilen Applikation nicht mit den Spezifika des jeweiligen Backendsystems auseinandersetzen. Abbildung 6-2 veranschaulicht das Konzept der MBOs.

[25] http://www.eclipse.org, zugegriffen am 4.12.2012

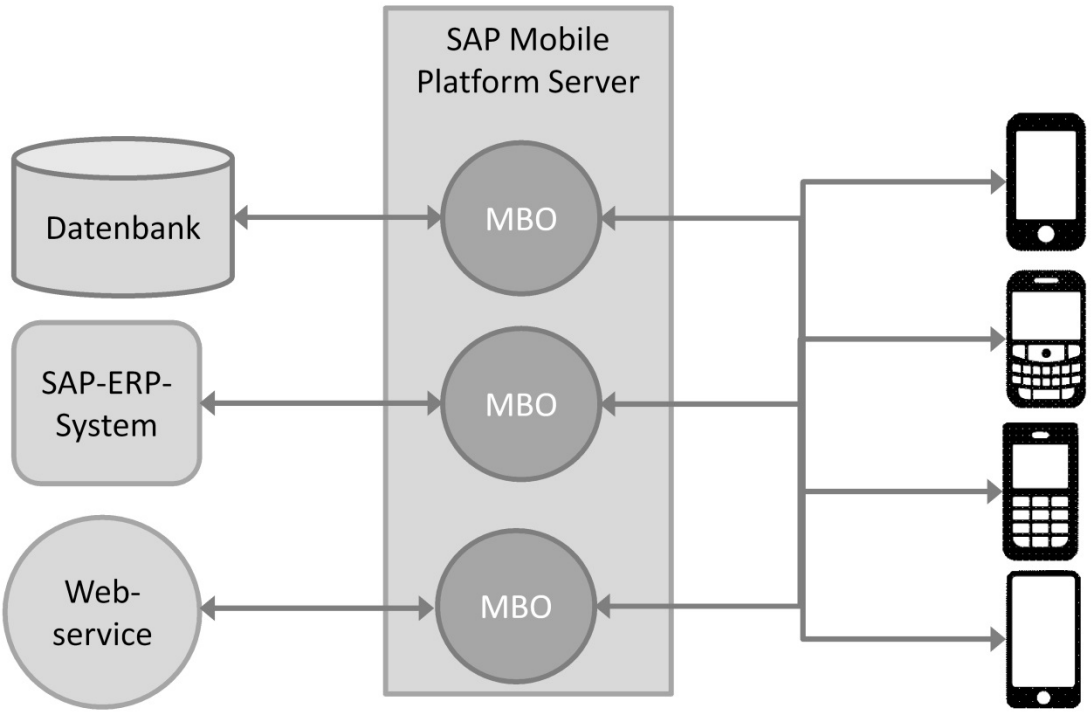

Abbildung 6-2: Konzept der Mobile Business Objects
Quelle: In Anlehnung an (Homann et al. 2013b, 185)

Der SAP Mobile Workspace stellt unterschiedliche Möglichkeiten bereit, um auf die Programmierschnittstellen eines SAP-ERP-Systems zuzugreifen. Eine Möglichkeit ist die Nutzung des Java Connectors (JCo), welcher die BAPIs des SAP-ERP-Systems über das proprietäre SAP-Protokoll RFC aufruft (siehe Kapitel 2.1.5.2.1). Eine weitere Möglichkeit ist der Aufruf von SOAP-basierten Webservices. Hierzu müssen die zugehörigen Webservices sowie deren WSDL zunächst im SAP-ERP-System auf der Basis von BAPIs oder Funktionsbausteinen generiert werden. Schließlich ist es auch möglich RESTful-Webservices zu nutzen. Hierzu müssen die zugehörigen RESTful-Webservices zunächst über ein spezielles Add-On des SAP-ERP-Systems, dem sogenannten SAP NetWeaver Gateway, auf der Basis von BAPIs oder Funktionsbausteinen erstellt werden. In allen beschriebenen Varianten stehen komfortable Assistenten für den MBO-Erstellungsprozess zur Verfügung. Durch ihre Nutzung beschränkt sich die Erstellung eines MBO im Wesentlichen auf die Auswahl eines entsprechenden BAPIs bzw. Funktionsbausteines und der zugehörigen Ein- und Ausgabeparameter. Abbildung 6-3 illustriert zwei exemplarische Bildschirmaufnahme des Assistenten im SAP Mobile Workspace zur Erstellung eines MBO für den Zugriff auf einen BAPI eines SAP-ERP-Systems.

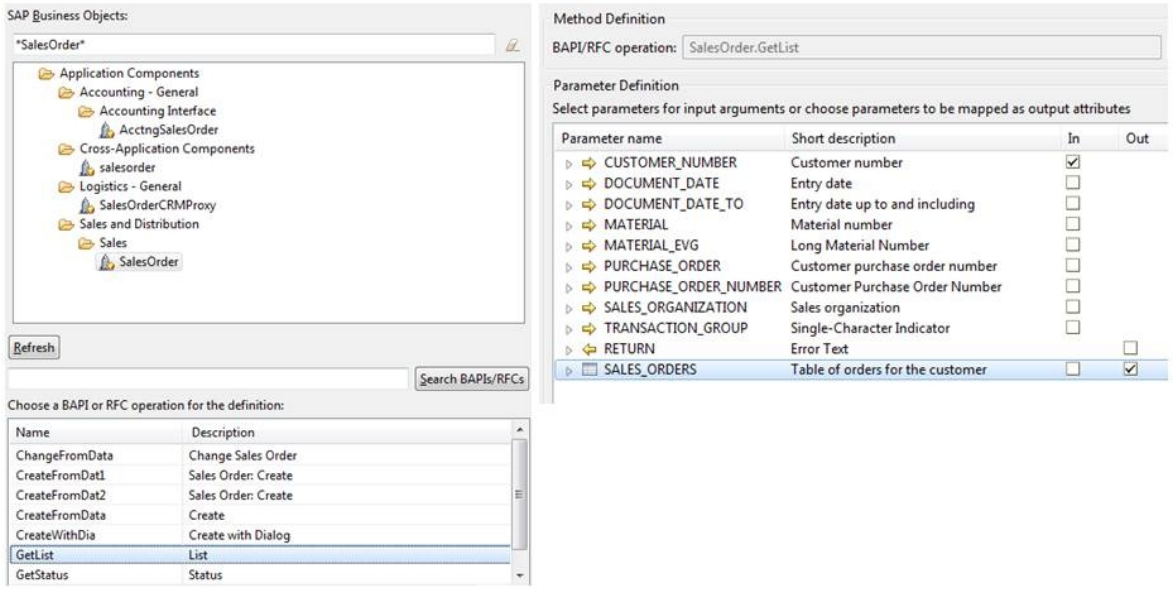

Abbildung 6-3: Assistent zur Erstellung eines MBO für den Zugriff auf ein SAP-ERP-System
Quelle: Bildschirmaufnahme aus dem SAP Mobile Workspace in der Version 2.3

Der Assistent abstrahiert von der jeweils verwendeten Zugriffstechnologie auf die Programmierschnittstellen des SAP-ERP-Systems und stellt hierdurch eine Arbeitserleichterung dar. Als nachteilig kann die Auswahl der Eingabe- und Ausgabeparameter angesehen werden. Hierbei muss der Programmierer die gültigen Kombinationsvarianten kennen, um ein funktionsfähiges Ergebnis zu erzielen. Aktuell erlaubt der Assistent auch die Auswahl nicht funktionsfähiger Kombinationen, welche beim Aufruf der Programmierschnittstelle des SAP-ERP-Systems zu einem Fehler führen.

Der Entwicklungsprozess mit dem SAP Mobile Workspace beginnt mit der Entwicklung der benötigten MBOs. Diese werden anschließend in den SAP Mobile Platform Server installiert. Im Falle einer Container-Applikation wird deren Benutzungsschnittstelle anschließend mit dem SAP Mobile Workspace erstellt und in den SAP Mobile Platform Server installiert und schließlich einem mobilen Gerät zugewiesen. Im Falle einer nativen Applikation wird im SAP Mobile Workspace zunächst eine Programmierschnittstelle für die erstellten MBOs in der jeweiligen Programmiersprache der nativen Applikation generiert. Diese wird anschließend in die jeweilige native Entwicklungsumgebung importiert. Im nächsten Schritt wird die Benutzungsschnittstelle für die native Applikation mit Hilfe der nativen Entwicklungsumgebung entwickelt. Abbildung 6-4 illustriert den beschriebenen Entwicklungsprozess mit Hilfe des SAP Mobile Workspace.

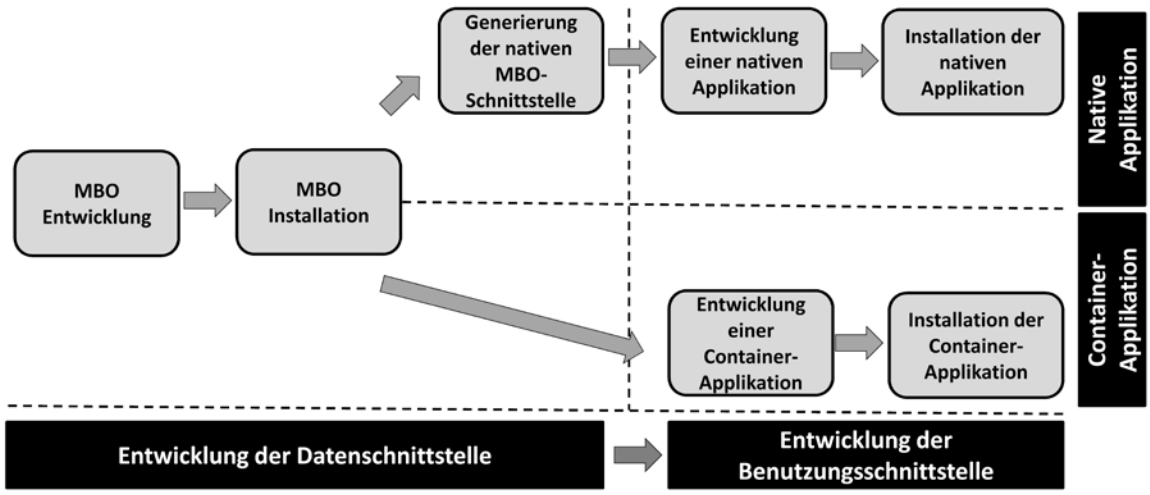

Abbildung 6-4: SAP Mobile Workspace Entwicklungsprozess
Quelle: In Anlehnung an (Homann et al. 2013b, 67)

Der SAP Mobile Workspace macht bei der Entwicklung mobiler Applikationen Gebrauch von unterschiedlichen Interaktionstechniken. Bei der Entwicklung von MBOs kommen hauptsächlich Assistenten zum Einsatz, welche den Entwickler Schritt-für-Schritt durch den Erstellungsprozess leiten. Nach der initialen Erstellung der MBOs können deren Eigenschaften über einfache Attribut-Wert-Editoren oder über spezielle Menüpunkte des Kontextmenüs angepasst werden. Bei der Entwicklung der Benutzungsschnittstelle verfolgt der SAP Mobile Workspace einen modellgetriebenen Entwicklungsansatz. Beispielsweise können rudimentäre Bildschirmmasken bereits aus den MBOs generiert werden. Die weiterführende Gestaltung des Kontrollflusses der Applikation sowie einzelner Bildschirmmasken erfolgt über einen visuellen Programmieransatz mit Hilfe eines UI-Editors. Dieser erlaubt u.a. die Platzierung von Anzeige- und Bedienelementen auf den Bildschirmmasken über einen Drag & Drop-Mechanismus. Die entwickelte Benutzungsschnittstelle wird in Form einer XML-Datei abgespeichert und wird durch einen Generierungsvorgang in HTML-, CSS- und JavaScript Dateien auf Basis des jQuery Mobile Frameworks transformiert. Die generierten HTML-, CSS- und JavaScript-Dateien können anschließend mit einem Texteditor angepasst werden. Jedoch wird vom SAP Mobile Workspace aktuell kein Reverse-Engineering unterstützt, d.h. die in den HTML-, CSS- oder JavaScript Dateien durchgeführten Änderungen werden nicht in das Modell übertragen. Abbildung 6-5 illustriert den UI-Designer des SAP Mobile Workspace beispielhaft für die Gestaltung des Kontrollflusses einer mobilen Applikation.

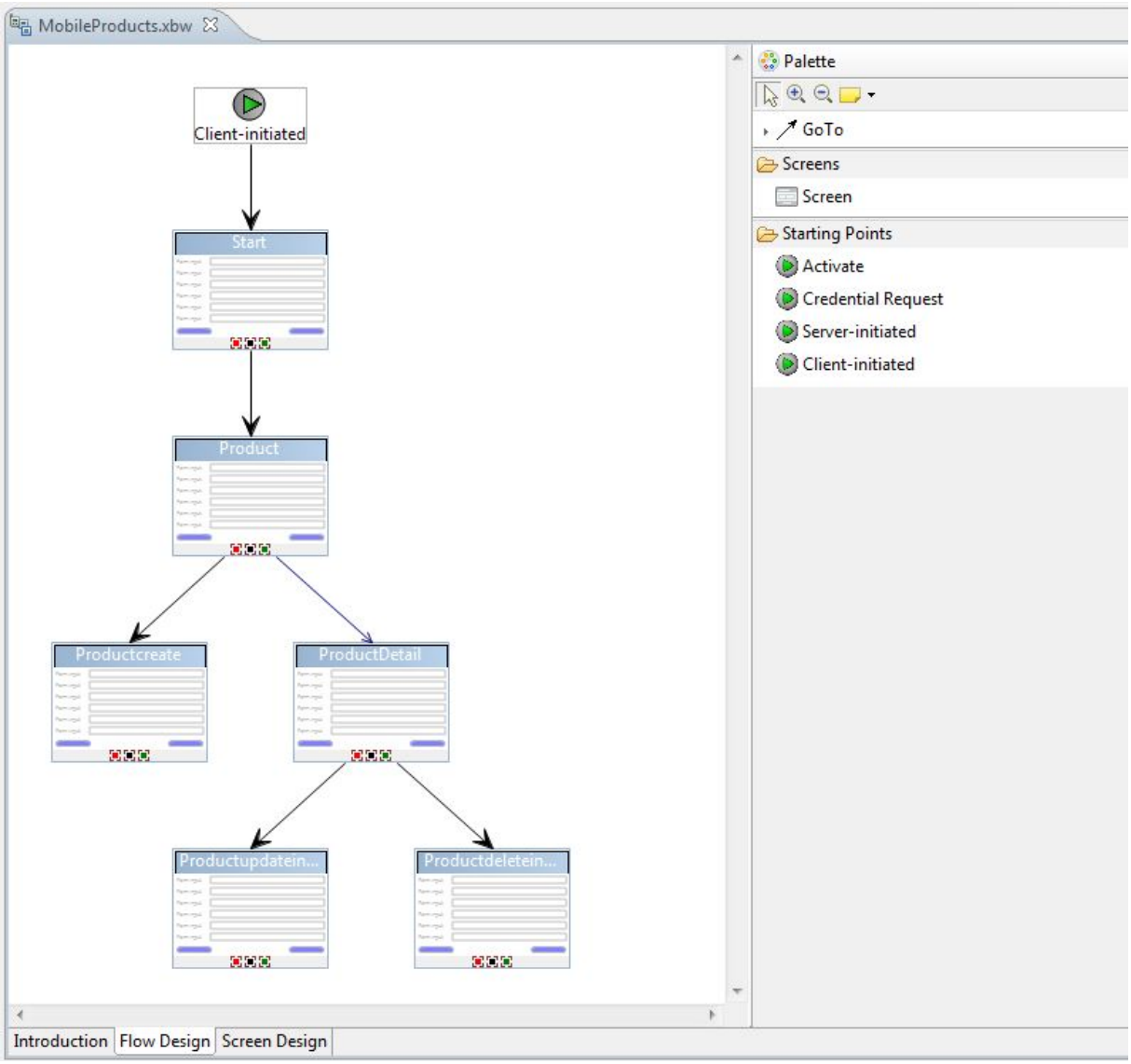

Abbildung 6-5: Gestaltung des Kontrollflusses einer mobilen Applikation im SAP Mobile Workspace
Quelle: Bildschirmaufnahme aus dem SAP Mobile Workspace in der Version 2.3

Insgesamt sind die generierten Benutzungsschnittstellen sehr rudimentär und genügen nicht den in Kapitel 3.3.2.1 identifizierten Benutzungsschnittstellen mobiler ERP-Applikationen. Bei der Anpassung der Benutzungsschnittstellen müssen die Entwicklungskonzepte des SAP Mobile Workspace zunächst erlernt bzw. verstanden werden. Beispielsweise werden Resultate aus MBO Methoden über sogenannte „Keys“ mit den Anzeige- und Bedienelementen der Benutzungsschnittstelle verknüpft. Um die Entwicklung mit dem SAP Mobile Workspace zu erlernen hat SAP im Internet mehrere Tutorials bereitstellt. Zudem bietet SAP derzeit eine 5-tägige Klassenraumschulung[26] an, um die Entwicklung mobiler Applikationen mit dem SAP Mobile Workspace zu erlernen. Dies zeigt, dass es sich beim SAP Mobile Workspace nicht

[26] Siehe https://training.sap.com/v2/course/sup521-sybase-unwired-platform-21-mobile-application-development-classroom-095-de-de/schedules, zugegriffen am 6.12.2013

um ein intuitiv zu bedienendes Entwicklungswerkzeug handelt. Vielmehr ist ein mehrtägiger Einarbeitungsaufwand notwendig ist, um erste mobile ERP-Applikationen zu entwickeln. Zudem sind grundlegende Programmierkenntnisse notwendig um Konzepte wie beispielsweise Kontrollflüsse oder Variablen zu verstehen.

6.1.3 Entwicklung mobiler SAP-ERP-Applikationen mit den NetWeaver Gateway Plug-In

Bei SAP NetWeaver Gateway, kurz Gateway, handelt es sich sowohl um ein Entwicklungs-Framework als auch um eine Integrationsschicht, welche als Add-on des SAP NetWeaver Application Servers ABAP zur Verfügung steht (Beckert et al. 2012, 239 f.). Motivation für Gateway ist die hohe Komplexität der aktuellen Programmierschnittstellen des SAP-ERP-Systems, den BAPIs. Diese Komplexität resultiert aus der hohen Parameteranzahl und den vielfältigen Parametrisierungsoptionen, deren korrekte Auswahl i.d.R. Fachwissen erfordert (Beckert et al. 2012, 239 f.) sowie der Verwendung des proprietären SAP-Netzwerkprotokolls RFC (Homann et al. 2013b, 40 f.). Gateway verfolgt das Ziel, die Nutzung der Programmierschnittstellen des SAP-ERP-Systems zu vereinfachen und das Datenvolumen bei der Kommunikation zu reduzieren. Dies erfolgt durch die Verwendung einer offenen, standardbasierten Technologie auf der einen Seite und der Reduktion der Programmierschnittstelle auf ein für den jeweiligen Applikationsfall notwendiges Maß (Beckert et al. 2012, 239 f.; Homann et al. 2013b, 40 f.). Der Fokus von Gateway liegt auf der Integration von browserbasierten Applikationen und mobilen Applikationen (Beckert et al. 2012, 239 f.).

Mit Hilfe der Gateway-Werkzeuge können aus bestehenden BAPIs Webservices generiert werden, welche auf den REST-Architekturprinzipien (Representational State Transfer) basieren. Die generierten Webservices werden als Gateway-Services bezeichnet (Beckert et al. 2012, 265). Gateway-Services nutzen i.d.R. nur eine bestimmte Parameterkombination des zugrundeliegenden BAPIs und sind daher einfacher nutzbar. Gateway-Services basieren auf dem offenen OData-Protokoll (Open Data), welches wiederrum auf dem offenen Atom Publishing Protocol (AtomPub) und dem Atom Syndication Format (ASF) basiert. AtomPub ist ein einfaches HTTP-basiertes Protokoll zum Erstellen und Bearbeiten von Webressourcen. ASF ist ein XML-basiertes Protokoll für den plattformunabhängigen Datenaustausch (Beckert et al. 2012, 246 ff.; Homann et al. 2013b, 45 ff.).

Zur Entwicklung von Applikationen auf Basis von Gateway-Services stellt SAP eine Reihe von Eclipse-Plug-Ins zur Verfügung. Beispielsweise werden die Entwicklungstechnologien HTML5, Java und PHP unterstützt (SAP 2013). Zur Entwicklung von mobilen Applikationen wird ein Eclipse-Plug-In zur Entwicklung nativer Android Applikationen bereitgestellt. Dieses Plug-In und der zugehörige Entwicklungsprozess sind im Folgenden Gegenstand der Untersuchung.

Voraussetzung für die Nutzung des Gateway-Plug-Ins ist eine existierende Eclipse-Entwicklungsumgebung sowie ein installiertes Android Development Toolkit (ADT) (Homann et al. 2013b, 488 f.). Das Gateway-Plug-In stellt folgende Funktionalitäten bereit (Homann et al. 2013b, 488):.

- Die Nutzung und Erstellung von Schablonen (engl. templates) zur komfortablen Generierung von nativen Android Applikationen.
- Die Generierung von OData-basierten Kommunikationskomponenten (engl. proxys) für die Kommunikation mit den zugehörigen Gateway-Services.
- Eine Werkzeug für die Suche nach geeigneten Gateway-Services, die sogenannten „Search Console“.

Der Entwicklungsprozess einer nativen Android Applikation unter Nutzung des Gateway Plug-Ins beginnt mit der Erstellung eines entsprechenden Gateway-Service. Hierzu wird ein Datenmodell mithilfe des sogenannten Service Builders erstellt. Der Service Builder ist aktuell als Transaktion über die SAP GUI nutzbar. Anschließend muss eine sogenannte „Metadata Provider Class“ als auch eine sogenannte „Runtime Data Provider Class“ entwickelt werden. In einem nächsten Schritt kann der entwickelte Gateway-Service generiert werden. In einem letzten Schritt muss der Service aktiviert und im Gateway-System registriert werden. Damit ist die Entwicklung der Datenschnittstelle abgeschlossen (Beckert et al. 2012, 268 ff.).

Zur Entwicklung der Benutzungsschnittstelle wird ein entsprechender Assistent des Gateway Plug-Ins verwendet. Dieser leitet den Entwickler Schritt-für-Schritt durch den Erstellungsprozess einer mobilen Applikation auf Basis eines oder mehrerer Gateway-Services. Nachdem allgemeine Informationen der mobilen Applikation, wie bspw. der Applikationsname spezifiziert wurden, muss eine Applikationsschablone ausgewählt werden. Aktuell stehen folgende Applikationsschablonen zur Verfügung (vgl. Homann et al. 2013b, 498):

- **Basic Application:** hierbei wird das Grundgerüst einer Android Applikation zur Nutzung eines oder mehrerer Gateway-Services generiert. Generierte Elemente sind u.a. die Ordnerstruktur des Entwicklungsprojektes, die Verlinkung mit den notwendigen Programmierbibliotheken sowie ein Proxy für die Kommunikation mit einem ausgewählten Gateway-Service. Die generierten Elemente sind bewusst allgemein gehalten und sollen als Ausgangsbasis zur Entwicklung einer eigenen Applikation dienen.
- **List/Details-Application:** bei dieser Applikationsschablone werden alle Elemente einer Basic Application sowie eine zusätzliche Bildschirmmaske für die Auflistung aller Instanzen eines Business Objekt Typen generiert. Zudem wird eine zugehörige Bildschirmmaske zur Detailansicht der Attributwerte eines selektierten Business Objekts erstellt.
- **Workflow Application:** bei dieser Applikationsschablone werden ebenfalls alle Elemente einer Basic Application generiert. Zusätzlich wird eine Bildschirmmaske für typische Genehmigungsarbeitsschritte erzeugt. Diese Bildschirmmaske umfasst die Anzeige offener Genehmigungsanfragen mit entsprechenden Details zur Anfrage sowie einer Möglichkeit zur Genehmigung oder Ablehnung einer selektierten Anfrage.

Im Anschluss daran kann über ein Suchwerkzeug nach dem entwickelten Gateway-Service gesucht werden. Nach dessen Selektion können die Datenfelder des Gateway-Service ausgewählt werden, welche in den Bildschirmmasken der mobilen Applikation angezeigt werden sollen. Abbildung 6-6 illustriert eine beispielhafte Bildschirmaufnahme des Assistenten zur Konfiguration einer nativen Android Applikation über das Gateway Plug-In. In diesem Beispiel wurde die Schablone „List/Detaills Application“ ausgewählt und ein Gateway-

Service für den Zugriff auf ein Demo-Beispiel eines SAP-ERP-Systems entwickelt, der sogenannten Flugdaten-Applikation (SAP o.J.-b).

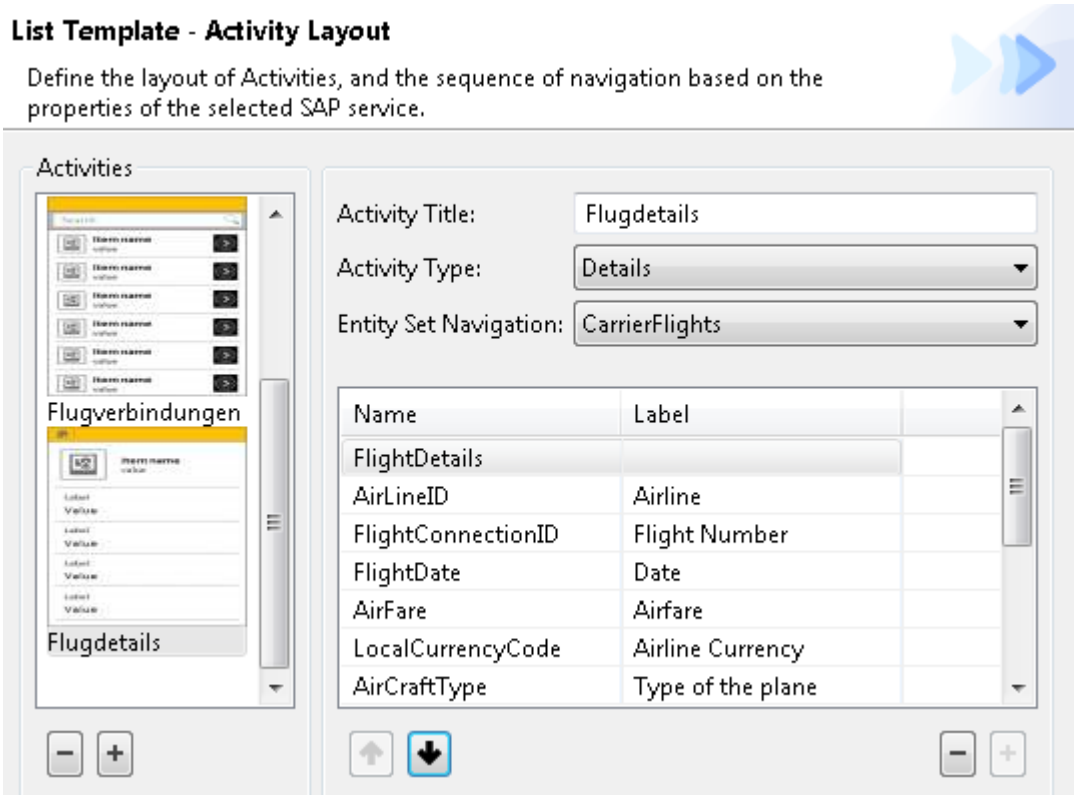

Abbildung 6-6: Konfiguration der Bildschirmmasken einer mobilen Applikation mit dem Gateway Plug-In

Quelle: Bildschirmaufnahme aus dem SAP NetWeaver Gateway Plug-In in der Version 2.6.4

Nachdem die benötigten Gateway-Services inkl. ihrer Datenfelder ausgewählt wurden, kann die Android Applikation generiert werden. Hierbei wird eine entsprechende Ordnerstruktur in Eclipse erzeugt und zugehörige Android-Konfigurationsdateien und Java-Quellcode-Dateien generiert. In einem letzten Schritt können die generierten Android-Projektdateien kompiliert und auf einem mobilen Gerät installiert werden. Abbildung 6-7 illustriert den beschriebenen Entwicklungsprozess mit Hilfe des Gateway Plug-Ins.

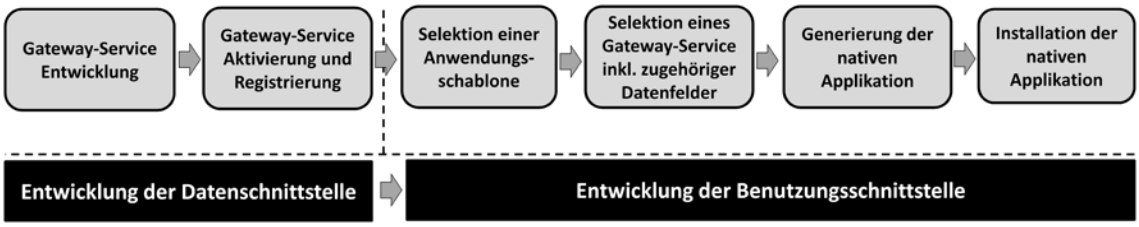

Abbildung 6-7: SAP NetWeaver Gateway Entwicklungsprozess
Quelle: Eigene Darstellung

In Bezug auf die Interaktionstechnik setzt das Gateway Plug-In hauptsächlich auf eine formularbasierte Interaktion. Hierbei wird der Entwickler Schritt-für-Schritt durch den Erstellungsprozess der Applikation geleitet. Durch die Fokussierung auf Applikationsschablonen wird die Entwicklung auf ausgewählte Benutzungsschnittstellen begrenzt. Aktuell sind die Hauptanwendungsfälle die Anzeige von Business Objekten und Genehmigungs-Workflows. Der Assistent ermöglicht die Erstellung einer nativen Android Applikation in wenigen Minuten. Zuvor müssen allerdings einige Konfigurationsschritte im Gateway Plug-In durchgeführt werden, beispielsweise die Verbindungseinstellungen zum entsprechenden Gateway-Server. Bevor das Gateway Plug-In genutzt werden kann, müssen zudem die entsprechenden Schritte zur Bereitstellung der benötigten Gateway-Services durchgeführt werden. Die Entwicklung einer mobilen Applikation mit Hilfe des Gateway Plug-Ins ist relativ intuitiv und leicht erlernbar. Allerdings ist der sichere Umgang mit der Entwicklungsumgebung Eclipse für die Nutzung des Plug-Ins erforderlich oder zumindest vorteilhaft. Für eine anschließende Anpassung der generierten Android-Dateien sind zudem Android-Entwicklungskenntnisse notwendig. Auch die Bereitstellung geeigneter Gateway-Services über die Gateway Entwicklungswerkzeuge erfordert eine gewisse Einarbeitungszeit. Hierzu stellt SAP beispielsweise eine 3-tägige Schulung[27] sowie einige frei verfügbare Tutorials zur Verfügung.

Insgesamt genügen die generierten Benutzungsschnittstellen einem großen Teil der in Kapitel 3.3.2.1 identifizierten wiederkehrenden Funktionalitäten und Benutzungsschnittstellen mobiler ERP-Applikationen. Jedoch fehlen aktuell Schablonen zur Manipulation von Business Objekten. Nachteilig ist zudem die erforderliche Kenntnis im Umgang mit der Entwicklungsumgebung Eclipse und die Einschränkung auf native Android Applikationen. Die formularbasierte Konfiguration durch einen Assistenten und die anschließende Generierung der konfigurierten mobilen Applikation erscheint hingegen als intuitive Interaktionstechnik zur Erstellung mobiler ERP-Applikationen.

6.1.4 Entwicklung mobiler SAP-ERP-Applikationen mit dem SAP AppBuilder

Beim SAP *AppBuilder* handelt es sich um ein Webbrowser-basiertes Entwicklungswerkzeug für mobile HTML5-basierte Applikationen. Die erstellten mobilen Applikationen nutzen das

[27] https://training.sap.com/v2/course/gw100-sap-netweaver-gateway---building-odata-services-remoteclassroom-098-de-en, zugegriffen am 09.12.2013

HTML5-Framework SAPUI5 der SAP AG. SAP UI5 stellt eine Menge von Anzeige- und Bedienelementen für die Erstellung mobiler Applikationen zur Verfügung, welche durch ihr vordefiniertes Layout und Verhalten die Entwicklungsproduktivität erhöhen und die Konsistenz bzgl. des Layouts sowie die Bedienung der erstellten Applikationen verbessert (SAP o.J.-d).

Die Integration mit einem SAP-ERP-System erfolgt über entsprechende Gateway-Services eines NetWeaver Gateway Systems. Neben Gateway-Services werden auch andere OData- oder RESTful-Webservices unterstützt. Der Entwicklungsprozess einer mobilen ERP-Applikation mit Hilfe des AppBuilder beginnt ebenso wie beim Gateway Plug-In mit der Entwicklung eines geeigneten Gateway-Service für den Datenaustausch mit dem SAP-ERP-System. Anschließend muss der erstellte Gateway-Service für die Nutzung registriert und aktiviert werden. Danach wird der erstellte Gateway-Service im AppBuilder konfiguriert. Dies erfolgt über die Definition einer sogenannten Data-Source, über welche die Verbindungsinformationen zum Gateway-Service, wie beispielsweise dessen URL sowie zugehörige Authentifizierungsdaten angegeben werden. Danach wird eine Metadaten-Datei definiert, welche die aus dem Gateway-Service darzustellenden Attribute definiert. Hierbei können u.a. einzelne Attribute des Gateway-Service ausgeblendet werden, die darzustellende Reihenfolge der Attribute festgelegt werden, ebenso wie die hierbei zu nutzenden Anzeige- und Bedienelemente. Durch die Definition der Metadaten-Datei wird ein spezielles Anzeige- bzw. Bedienelement generiert, welches anschließend über einen What-You-See-Is-What-You-Get (WYSIWYG) -Editor auf einer Bildschirmmaske der Applikation platziert werden kann. Über den WYSIWYG-Editor erfolgt anschließend auch die Gestaltung der restlichen Benutzungsschnittstelle. Anschließend kann die erstellte mobile Applikation auf einem mobilen Endgerät installiert werden. Hierzu stellt der AppBuilder unterschiedliche Möglichkeiten zur Verfügung. Eine Möglichkeit ist eine Exportfunktion, welche eine mobile Applikation auf Basis von *Apache Cordova* erstellt. Bei Apache Cordova[28] handelt es sich um einen Runtime-Wrapper zur Entwicklung von Hybrid-Applikationen. Aktuell unterstützt der AppBuilder das Exportieren einer Android- und iOS-Applikation auf Basis von Apache Cordova. Eine weitere Möglichkeit ist die Installation der erstellten mobilen Applikation in einen SAP Mobile Platform Server. Anschließend kann der Zugriff auf die erstellte Applikation über den SAP Mobile Platform Server verwaltet werden. Abbildung 6-8 illustriert den beschriebenen Entwicklungsprozess mit Hilfe des SAP AppBuilders.

[28] http://cordova.apache.org, zugegriffen am 10.12.2013

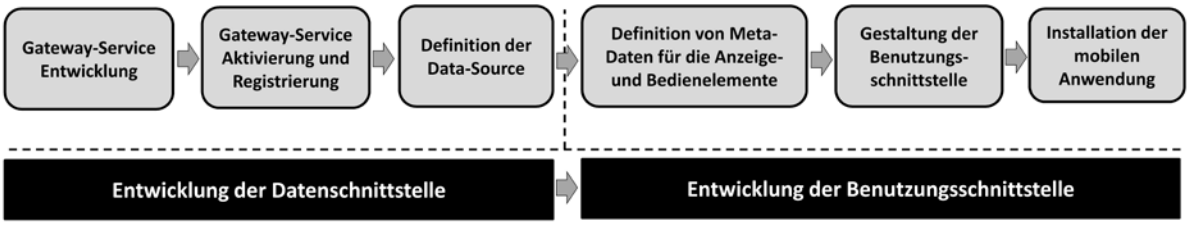

Abbildung 6-8: Entwicklungsprozess mit dem SAP AppBuilder
Quelle: Eigene Darstellung

Der AppBuilder erinnert an einen UI-Builder aus klassischen Entwicklungsumgebungen. Anzeige- und Bedienelemente stehen über eine Toolbar zur Verfügung und können via Drag & Drop auf die Bildschirmmasken der mobilen Applikation platziert und anschließend über bereitgestellte Eigenschaftswerte konfiguriert werden. Die hierbei verwendete Interaktionstechnik kann daher der visuellen Programmierung zugeordnet werden. Für häufig verwendete Schritte, wie bspw. die Definition einer Data-Source oder das Exportieren bzw. Installieren der entwickelten mobilen Applikation stehen Assistenten zur Verfügung, welche den Entwickler Schritt-für-Schritt durch die entsprechenden Aktivitäten führen. Daher kommt auch die formularbasierte Programmierung als Interaktionstechnik zum Einsatz. Aktionen auf bestimmte Ereignisse, z.B. auf Benutzeraktionen, müssen derzeit über JavaScript implementiert werden. Daher ist im AppBuilder auch ein JavaScript-Texteditor integriert.

Der Vorteil des AppBuilders ist seine schnelle Lauffähigkeit ohne vorherige Installation. Für die Kommunikation mit einem Gateway-Service muss jedoch ein separater Webserver, wie beispielsweise der Apache Webserver, installiert werden. In diesem Fall muss auch die Kommunikationsverbindung zwischen dem AppBuilder und dem Webserver konfiguriert werden. Durch die Anlehnung an das Bedienkonzept gängiger UI-Builder sollten sich erfahrene Softwareentwickler relativ schnell im AppBuilder zurechtfinden. Für Endbenutzer ist dies sicherlich eine größere Herausforderung. Eine weitere Herausforderung für Endbenutzer ist die Bereitstellung von Ereignisbehandlungsroutinen über JavaScript. SAP hat eine Reihe von Tutorials und Video-Anleitungen bereitgestellt, um den Umgang mit dem AppBuilder zu erlernen. Der Einarbeitungsaufwand ist abhängig von den Vorkenntnissen des Entwicklers. Erfahrene Entwickler, insbesondere Webentwickler, sollten sich im AppBuilder relativ schnell zurechtfinden. Endbenutzer müssten sich zunächst generelle Kenntnisse bzgl. Ereignisbehandlungsroutinen und JavaScript aneignen, um den Umgang mit dem AppBuilder zu erlernen.

6.2 Endbenutzer-Entwicklungswerkzeuge für mobile Applikationen

Im Folgenden werden aktuell existierende Endbenutzer-Entwicklungswerkzeuge für mobile Applikationen untersucht.

6.2.1 Marktüberblick

Aktuell existieren bereits einige Entwicklungswerkzeuge für Endbenutzer zur Erstellung mobiler Anwendungen. Diese Werkzeuge werden oftmals als „App-Builder“ bezeichnet und werben damit, dass sie es Laien ermöglichen mobile Applikationen nach einem „Baukastenprinzip“ erstellen lassen (Wyllie 2013). Grundlage für die folgende Untersuchung stellt der Marktüberblick über Endbenutzer-Entwicklungswerkzeuge für mobile Applikationen von Diego Wyllie dar, welcher im Mai 2013 im t3n Magazin erschienen ist (Wyllie 2013). In dieser Marktübersicht befinden sich hauptsächlich Werkzeuge, welche von ihren Herstellern als Software-as-a-Service (SaaS) Angebot gegen eine monatliche Nutzungsgebühr angeboten werden. Dabei handelt es sich i.d.R. um webbasierte Entwicklungswerkzeuge, welche keine vorherige Installation oder Konfiguration benötigen. Hauptfokus dieser Werkzeuge sind i.d.R. Kundenapplikationen. Die Übersicht enthält jedoch auch Entwicklungswerkzeuge wie bspw. AppConKit[29] der Firma Webtun, welche durch Funktionalitäten wie lokale Datenspeicherung oder vordefinierte Adapter zur Integration unterschiedlicher Backendsysteme bewusst ihren Fokus auf Unternehmensapplikationen setzen. Da letztere jedoch hinsichtlich ihres Entwicklungsprozesses als auch ihrer eingesetzten Interaktionstechniken ähnlich zum bereits untersuchten SAP Mobile Workspace sind, werden sie im Folgenden nicht betrachtet. Aufgrund ihrer vergleichsweise umfangreichen Funktionalität werden im Folgenden die beiden Werkzeuge Mobile Roadie[30] und der ShoutEm Mobile App Maker[31] untersucht. Aufgrund seiner sich unterscheidenden Herangehensweise wird zusätzlich das Werkzeug Appery.io[32] analysiert, welches im Rahmen von Befragung II zu Demonstrationszwecken verwendet wurde. Hierzu werden die gleichen Gesichtspunkte analysiert wie bei den Entwicklungswerkzeugen für mobile Unternehmensapplikationen in Kapitel 6.1.

6.2.2 Mobile Roadie

Das Entwicklungswerkzeug Mobile Roadie ist als Webanwendung implementiert. Nach einer Registrierung kann das Werkzeug kostenlos getestet werden. Aktuell werden die beiden Betriebssysteme iOS und Android unterstützt. Als Implementierungsvariante kommen Container-Applikationen zum Einsatz. Dabei wird die Container-Applikation „Mobile Roadie Connect“ vom jeweiligen App Store heruntergeladen. Nachdem in den Einstellungen der Container-Applikation die Zugangsdaten des Benutzerkontos hinterlegt wurden, kann auf die erstellten Applikationen zugegriffen werden. Die mobilen Applikationen werden als HTML5-Applikationen erstellt und werden innerhalb der Container-Applikation über ein Webbrowser-Plug-In dargestellt. Mobile Roadie bietet jedoch auch eine Option an, aus den entwickelten mobilen Applikationen native Applikationen für die unterstützten Betriebssysteme iOS und Android zu generieren, welche anschließend über die zugehörigen App Stores bereitgestellt werden können.

[29] http://www.weptun.de/appconkit, zugegriffen am 10.09.2013
[30] https://mobileroadie.com, zugegriffen am 16.12.2013
[31] http://www.shoutem.com, zugegriffen am 17.12.2013
[32] http://appery.io, zugegriffen am 17.12.2013

Mobile Roadie unterstützt die Integration unterschiedlicher Internetdienste, wie beispielsweise Facebook, Twitter, YouTube oder Foursquare. Zudem wird das RSS-Format (Really Simple Syndication) zum Lesen von Nachrichten über sogenannte RSS-Feeds unterstützt. Im Mobile Roadie Entwicklungsprozess wird zwischen einer initialen Entwicklung und einer Nachbearbeitung unterschieden. Die initiale Entwicklung erfolgt Schritt-für-Schritt mit Unterstützung eines Assistenten. Zu Beginn müssen allgemeine Informationen der mobilen Applikation, wie bspw. der Name und die Kategorie der Applikation, spezifiziert werden. Mobile Roadie bietet derzeit eine Liste von auswählbaren Kategorien an, bspw. Konferenz, Radiosender, Finanzen oder Bildung. Für die einzelnen Kategorien existieren Schablonen, welche den Aufbau der einzelnen Bildschirmmasken der Applikation festlegen. Anschließend kann das vorgeschlagene Design der einzelnen Bildschirmmasken individuell angepasst werden. Dies umfasst beispielsweise das Hinzufügen eines Firmenlogos oder die Festlegung eines Farbschemas. Danach lassen sich Inhalte hinzufügen. Hierbei werden bekannte Internetdienste wie bspw. Twitter oder Facebook, zur Auswahl angeboten. Nach deren Auswahl müssen im Allgemeinen Zugangsparameter und dienstspezifische Parameter konfiguriert werden. Danach kann die erstellte Anwendung bereits über die Container-Applikation „Mobile Roadie Connection“ genutzt werden. Während des Entwicklungsprozesses wird dem Endbenutzer seine aktuelle Position im Entwicklungsprozess angezeigt. Zudem kann er flexibel zwischen den einzelnen Entwicklungsschritten wechseln. Abbildung 6-9 illustriert zwei ausgewählte Schritte aus dem Mobile Roadie Entwicklungsassistenten. Die linke Bildschirmaufnahme illustriert die Festlegung der allgemeinen Applikationsdaten. Die rechte Bildschirmmaske illustriert die Auswahl des Applikationslayouts.

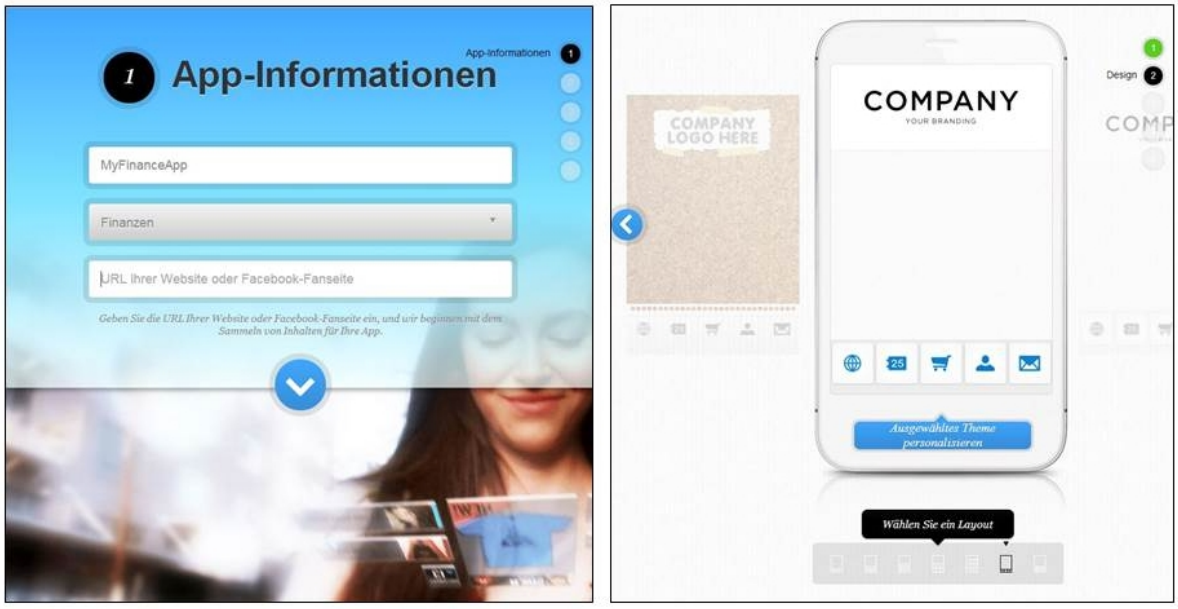

Abbildung 6-9: Mobile Roadie Entwicklungsassistent

Quelle: https://mobileroadie.com, zugegriffen am 16.12.2013

Nach der Durchführung der initialen Entwicklungsschritte besteht die Möglichkeit die Applikation in einer Nachbearbeitungsperspektive zu verändern oder zu erweitern. Hierzu werden verschiedene Entwicklungskategorien wie beispielsweise „Design“ und „Interagieren“ (mit Internetdiensten) angeboten. Zudem finden sich hier zusätzliche Funktionalitäten wie bspw. „Analytik“. Abbildung 6-10 illustriert die Werkzeugansicht in der Perspektive zur Nachbearbeitung der initial entwickelten Applikation.

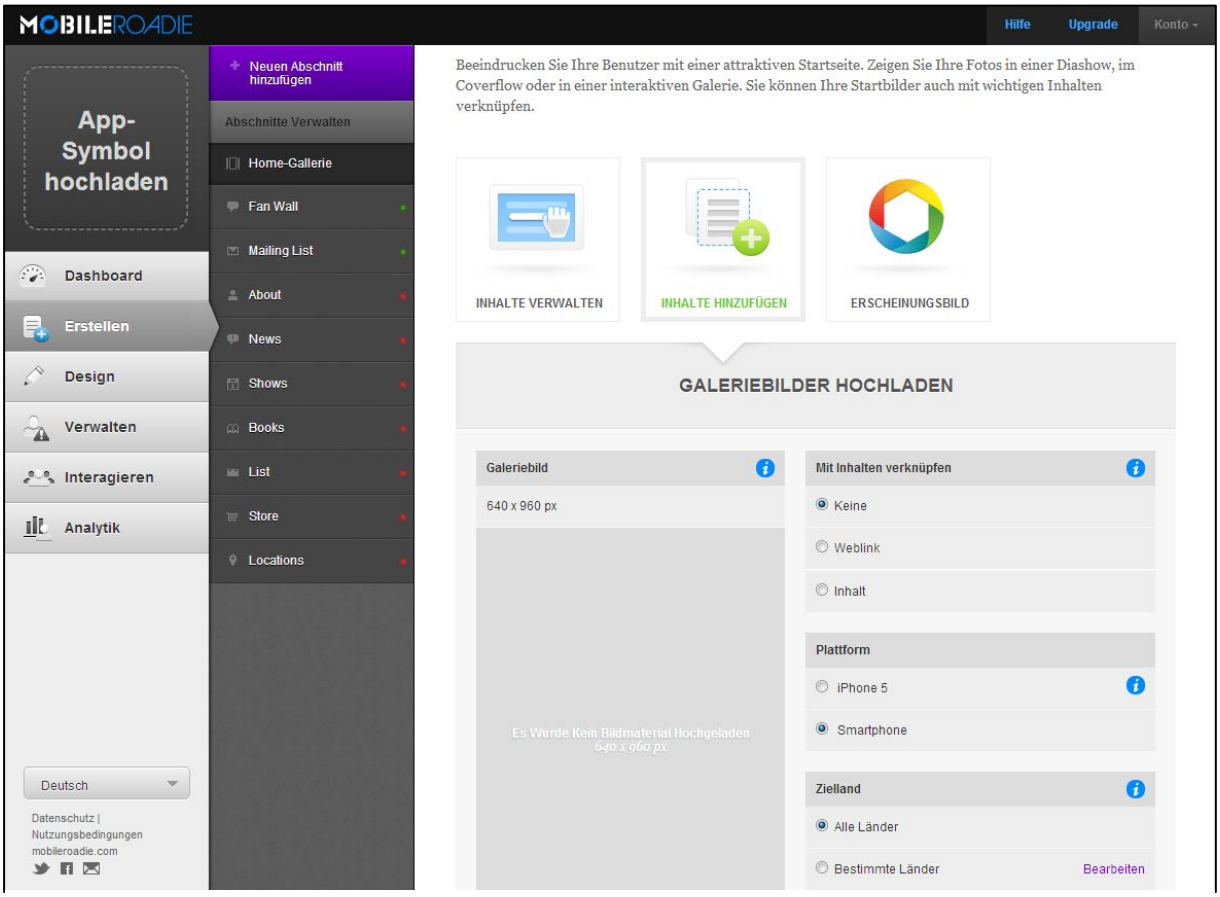

Abbildung 6-10: Mobile Roadie in der Nachbearbeitungsperspektive
Quelle: https://mobileroadie.com, zugegriffen am 16.12.2013

Als Interaktionstechniken kommt ausschließlich die formularbasierte Programmierung zum Einsatz. Für jeden Entwicklungsschritt existieren vordefinierte Schablonen mit vordefinierten Parametern, welche durch die Angabe von individuellen Werten an die eigenen Bedürfnisse angepasst werden können. Es gibt keine Möglichkeit selbst Programmcode zu schreiben.

Der Entwicklungsprozess ist intuitiv. Eine vorherige Einarbeitung ist nicht erforderlich. Durch die Führung durch den Assistenten werden werden fehlerhafte Parameterwerte umgehend bemerkt. Da die Freiheitsgrade insgesamt eingeschränkt sind, ist die Fehlerwahrscheinlichkeit insgesamt gering.

6.2.3 ShoutEm Mobile App Maker

Der ShoutEm Mobile App Maker ist ähnlich wie Mobile Roadie ein SaaS-Angebot. Er steht nach einem kostenlosen Testzeitraum gegen Zahlung einer monatlichen Gebührt zur Verfügung. Aktuell werden die Implementierungsvarianten native iOS- und native Android Applikationen sowie Webapplikationen unterstützt. Ähnlich wie bei Mobile Roadie wird die Integration von vordefinierten Internetdiensten unterstützt. Darunter befinden sich bekannte Internetdienste wie Facebook oder Flickr.

Der Entwicklungsprozess startet mit der Festlegung eines Namens für die Applikation. Anschließend kann eine Applikationsschablone aus einer Liste verfügbarer Schablonen ausgewählt werden. Darunter befinden sich Schablonen wie eine „Sports app“ oder eine „Business App“. Durch die Wahl der Schablone wird das Layout der zu entwickelnden Applikation festgelegt. Anschließend wechselt das Werkzeug in die Hauptperspektive, in welcher zwischen den unterschiedlichen Entwicklungsschritten flexibel gewechselt werden kann. Hier können sämtliche Einstellungen der Applikation vorgenommen werden. Beispielsweise können Inhalte über Internetdienste angebunden werden oder das Layout der einzelnen Bildschirmmasken angepasst werden. Abbildung 6-11 illustriert den Hauptbildschirm des ShoutEm Mobile App Maker.

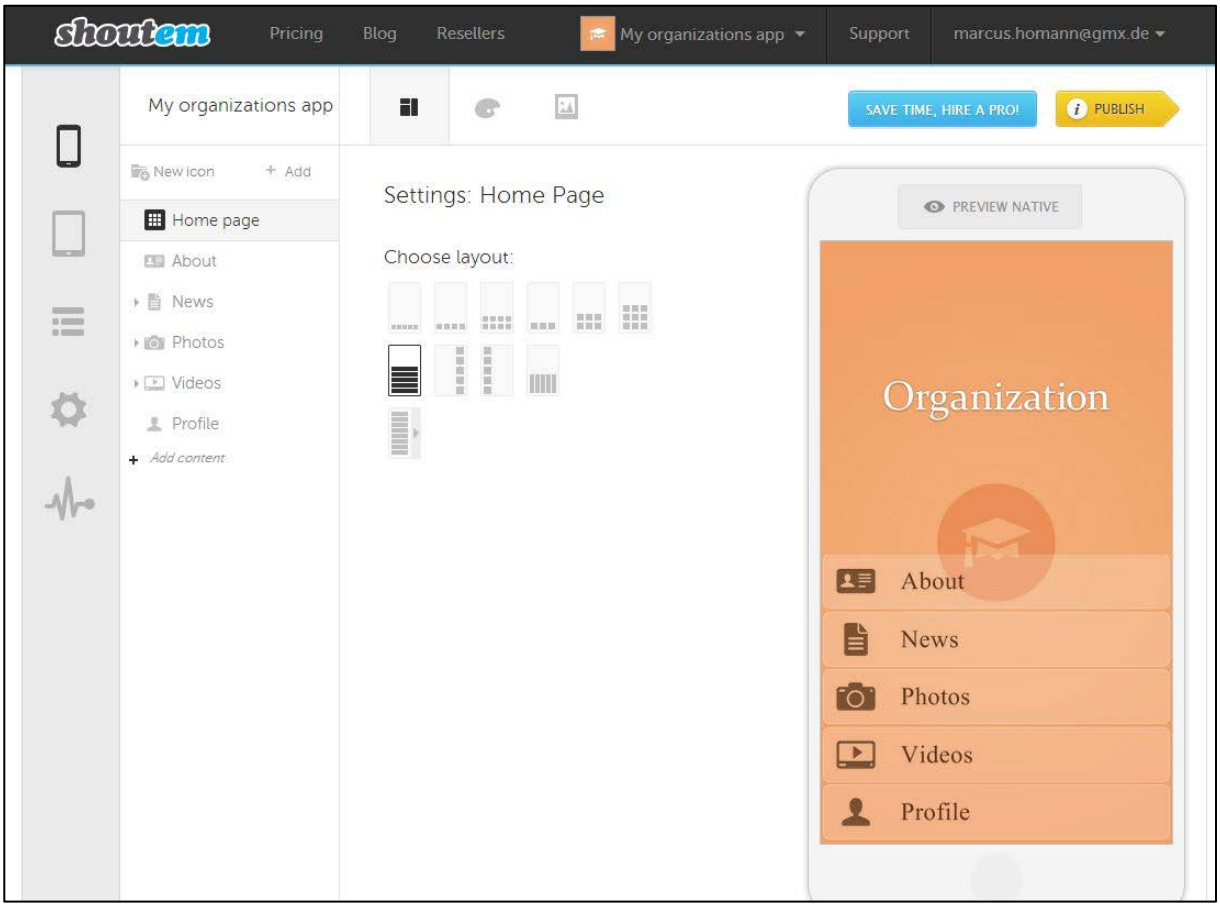

Abbildung 6-11: ShoutEm Mobile App Maker

Quelle: http://www.shoutem.com, zugegriffen am 16.12.2013

Als Interaktionstechnik setzt der Mobile App Maker auf die formularbasierte Programmierung. Der Endbenutzer hat keine Möglichkeit bzw. Notwendigkeit selbst Programmcode zu schreiben. Die Entwicklung erfolgt über die Auswahl vorgegebener Optionen oder die Zuweisung von Parameterwerten. Der initiale Entwicklungsprozess beschränkt sich beim Mobile App Maker auf die Festlegung eines Applikationsnamens und die Auswahl eines Layouts.

Insgesamt ist die Entwicklung mit dem ShoutEm Mobile App Maker relativ intuitiv. Das Werkzeug enthält zudem kontextbezogene Videos zur Unterstützung bei den einzelnen Entwicklungsschritten.

6.2.4 Appery.io

Das Entwicklungswerkzeug Appery.io ist ebenfalls als SaaS-Angebot konzipiert. Über einen Testzugang kann eine mobile Applikation umgesetzt werden; sollen mehrere mobile Applikationen umgesetzt werden, stehen verschiedene Vertragsoptionen über eine monatliche Gebühr zur Verfügung. Appery.io unterstützt sowohl Web-Applikationen, als auch Hybrid-Applikationen. Letztere werden über den Runtime-Wrapper von Apache Cordova[33] realisiert. Aktuell unterstützt das Werkzeug mobile Applikationen für die mobilen Betriebssysteme Android, iOS und Windows Phone. Appery.io ist als Webapplikation für Desktop-Rechner realisiert.

Die entwickelten Webapplikationen benötigen lediglich einen Webbrowser als Laufzeitumgebung; Hybrid-Applikationen benötigen zusätzlich die Apache Cordova Bibliotheken. Die Datenanbindung erfolgt in Appery.io über RESTful Webservices. Demzufolge ist eine Integration mit einem SAP-ERP-System nur über NetWeaver Gateway und zugehörige Gateway-Services möglich. Der im Werkzeug abgebildete Entwicklungsprozess besteht aus drei Schritten. In einem ersten Schritt wird die Benutzungsschnittstelle entwickelt. Hierbei kommt eine Kombination aus einer visuellen Programmierung und einer formularbasierten Programmierung als Interaktionstechnik zum Einsatz. Es stehen unterschiedliche Anzeige- und Bedienelemente zur Verfügung, welche über Drag & Drop auf den Bildschirmmasken einer mobilen Applikation platziert werden können. Deren Eigenschaften können anschließend über ein Formular mit einer Liste von Eigenschaften und zugehörigen Wertoptionen zugewiesen werden. Nach der Auswahl und Positionierung der Anzeige- und Bedienelemente können Interaktionsereignisse hinzugefügt werden und bei Bedarf umfangreichere Funktionalitäten programmiert werden. Hierbei kommt die Webprogrammiersprache JavaScript zum Einsatz. Anschließend werden die benötigten Data-Services über einen Assistenten Schritt-für-Schritt konfiguriert. Schließlich müssen die Elemente der Data-Services mit den Elementen der Benutzungsschnittstelle über den sogenannten „Visual Data Mapping Editor“ miteinander verknüpft werden. Hierbei kommt wiederrum ein visueller Programmieransatz über einen Drag & Drop Mechanismus zum Einsatz. Über sogenannte Plug-Ins stehen wiederverwendba-

[33] https://cordova.apache.org, zugegriffen am 16.12.2014

re Komponenten zur Verfügung, bspw. für den Aufruf von Funktionalitäten aus einem SalesForce CRM System[34]. Abbildung 6-12 illustriert den Hauptbildschirm von Appery.io.

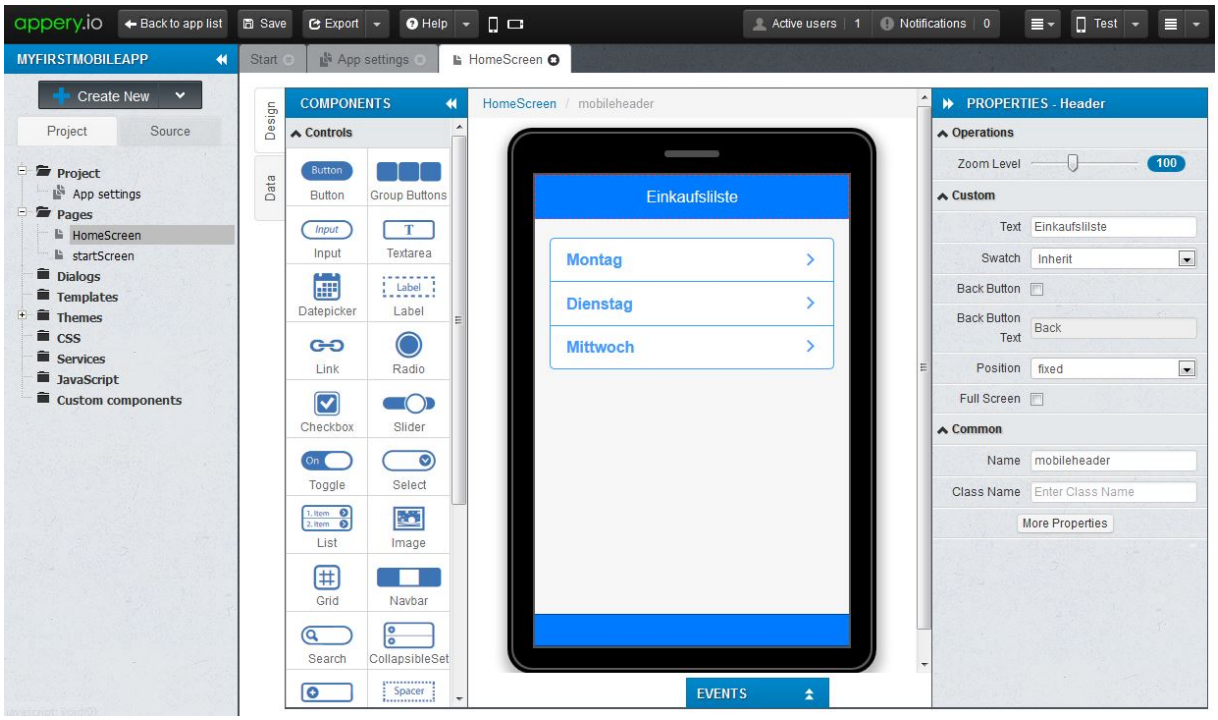

Abbildung 6-12: Appery.io

Quelle: http://appery.io/appbuilder, zugegriffen am 17.12.2013

Während der Entwicklung besteht die Möglichkeit den aktuellen Stand der Applikation über den genutzten Webbrowser auf dem Desktop-Computer oder über eine Appery.io Testapplikation auf dem Smartphone zu testen.

Im Gegensatz zu den beiden anderen in diesem Kapitel vorgestellten Endbenutzerwerkzeugen, ist Appery.io eher für fortgeschrittenere Endbenutzer mit ersten Programmiererfahrungen geeignet. Die Benutzungsschnittstelle des Werkzeuges ähnelt UI-Designern in klassischen Programmierumgebungen. Das Werkzeug bietet jedoch im Vergleich zu den anderen Werkzeugen umfangreichere Möglichkeiten. Um diese zu nutzen wird jedoch der Umgang mit der Webprogrammiersprache JavaScript vorausgesetzt. Um den Umgang mit dem Werkzeug zu erlernen werden unterschiedliche Materialien, wie bspw. Dokumentationen oder Videos über YouTube zur Verfügung gestellt.

6.3 Fazit und Diskussion

In der Untersuchung wurden drei Entwicklungswerkzeuge zur Erstellung mobiler Unternehmensapplikationen aus dem SAP-Portfolio sowie drei Endbenutzer-Entwicklungswerkzeuge

[34] http://www.salesforce.com/de, zugegriffen am 16.12.2013

zur Erstellung von mobilen Applikationen untersucht. Der Entwicklungsprozess der untersuchten Werkzeuge für mobile Unternehmensapplikationen unterscheidet sich bei den drei Werkzeugen. Bei allen drei Werkzeugen wird jedoch zunächst eine Datenschnittstelle entwickelt und anschließend eine zugehörige Benutzungsschnittstelle. Während der SAP Mobile Workspace unterschiedliche Technologien unterstützt, um auf ein SAP-ERP-System zuzugreifen, nutzen das NetWeaver Gateway Plug-In und der AppBuilder RESTful-Webservices des NetWeaver Gateways. Die Gestaltung der Benutzungsschnittstelle im AppBuilder als auch im SAP Mobile Workspace erfolgt über einen UI-Designer, bei welchem Anzeige- und Bedienelemente per Drag & Drop auf die Bildschirmmasken der mobilen Applikation platziert und anschließend über Eigenschaftswerte konfiguriert werden. Beim Gateway Plug-In ist die Gestaltung der Benutzungsschnittstelle hingegen standardisiert. Im Gateway Plug-In werden über einen Assistenten lediglich die Attribute des verknüpften Gateway-Service selektiert und deren Anzeigereihenfolge festgelegt. Die Benutzungsschnittstelle wird anschließend auf Basis einer zuvor gewählten Applikationsschablone generiert.

Um ein möglichst breites Anwendungsspektrum abzudecken, erlauben der SAP Mobile Workspace als auch der AppBuilder relativ viele Freiheitsgrade. Obwohl beide Produkte aus dem SAP Portfolio stammen, wird neben der Anbindung an SAP-ERP-Systeme auch die Anbindung anderer Backendsysteme unterstützt. Beispiele für andere unterstützte Backendsysteme sind Datenbanken unterschiedlicher Hersteller oder die Anbindung von Webservices. Die höheren Freiheitsgrade machen sich jedoch im Entwicklungsprozess und in der verwendeten Terminologie der Werkzeuge bemerkbar. Dies erfordert die Kenntnis von grundsätzlichen Programmierkonstrukten und erscheint daher für Endbenutzer weniger geeignet.

Daher kann in beiden Fällen nicht von einem domänenspezifischen Entwicklungswerkzeug gesprochen werden. Hingegen spezialisiert sich das Gateway-Plug-In auf Gateway-Services mit einem verknüpften SAP-ERP-System. Durch diese Spezialisierung ist der Einarbeitungsaufwand in das Werkzeug deutlich geringer als bei den anderen beiden untersuchten Werkzeugen. Auch der Erstellungsprozess einer neuen mobilen ERP-Applikation ist schneller als bei den beiden anderen Werkzeugen. Weniger Einstiegshürden für Endbenutzer sollten beim Gateway Plug-In auftreten, da dieses über einen Assistenten Schritt-für-Schritt durch den Erstellungsprozess leitet. Nachteilig ist hier jedoch die Installation des Gateway Plug-Ins über Eclipse und die Einschränkung auf Android. Zudem erscheint die aktuelle Terminologie des Assistenten eher für Programmierer geeignet, als für Endbenutzer. So muss beispielsweise die URL eines Gateway-Services spezifiziert, wobei anschließend die Attribute sowie die Datentypen des selektierten Gateway-Services angezeigt werden. Jedoch kann davon ausgegangen werden, dass nicht jeder Anwender die gängigen Datentypen von derzeitigen Programmiersprachen, wie „Integer“ oder „String“ beherrscht.

Die untersuchten Endbenutzer-Entwicklungswerkzeuge, Mobile Roadie und Mobile App Maker, setzten einen ähnlichen Entwicklungsprozess um. Als Interaktionstechnik setzten beide Werkzeuge ausschließlich auf eine formularbasierte Interaktion. Die Freiheitsgrade sind in beiden Werkzeugen deutlich eingeschränkter als bei den untersuchten Werkzeugen für mobile Unternehmensapplikationen. Durch eine Auswahl von Applikationsschablonen sowie die einfache Anbindung verbreiteter Internetdienste ist der Einarbeitungsaufwand bei beiden

Werkzeugen sehr gering. Zudem ist die Erstellung einer neuen mobilen Applikation in kurzer Zeit möglich und die Fehlerwahrscheinlichkeit gering. Aktuell beschränken sich die untersuchten Endbenutzer-Entwicklungswerkzeuge auf die Integration von populären Internetdiensten wie Facebook oder Twitter. Eine Integration mit Programmierschnittstellen von Backendsystemen ist derzeit nicht vorgesehen. Beide Werkzeuge sind als Webapplikation für den Desktop-PC implementiert.

Das dritte untersuchte Endbenutzer-Werkzeug Appery.io ist ebenfalls als Webapplikation umgesetzt. Im Gegensatz zu den anderen beiden Werkzeugen besitzt es jedoch mehr Freiheitsgrade. Dies geht jedoch einher mit einem höheren Einarbeitungsaufwand und höheren Anforderungen bzgl. der Programmierkenntnisse der Endbenutzer.

Der zusammenfassende Vergleich der beiden Werkzeugtypen zeigt, dass die Endbenutzer-Entwicklungswerkzeuge generell weniger Freiheitsgrade besitzen. Dafür ist der Entwicklungsprozess jedoch intuitiver und erlaubt das Entwickeln einer neuen mobilen Applikation in wenigen Minuten. Es stellt sich daher die Frage, ob durch eine Beschränkung auf SAP-ERP-Programmierschnittstellen und damit dem Wegfall von Freiheitsgraden bzgl. der Datenschnittstelle ein ähnlich intuitives Entwicklungswerkzeug für mobile ERP-Applikationen erstellt werden kann. Dabei erscheint die Fokussierung auf eine formularbasierte Interaktionstechnik ebenfalls geeignet zu sein. Alle untersuchten Entwicklungswerkzeuge nutzen aktuell einen klassischen Desktop-PC. Insbesondere bei Verwendung einer formularbasierten Interaktionstechnik besteht jedoch das Potenzial, dass ein derartiges Werkzeug auch für mobile Endgeräte umgesetzt werden kann.

7 Architekturkonzept und Gestaltungsvorgehen

In diesem Kapitel wird zunächst die grobe Architektur für das gewünschte Entwicklungswerkzeug konzipiert. Anschließend wird das Vorgehen bei der Gestaltung der einzelnen Elemente der vorgestellten Architektur beschrieben.

7.1 Architekturkonzept des Entwicklungswerkzeuges

Abbildung 7-1 illustriert das grobe Architekturkonzept des gewünschten Entwicklungswerkzeuges. Die grundlegende Idee hinter diesem Architekturkonzept ist ein zweistufiger Entwicklungsprozess.

In einem ersten Entwicklungsschritt soll ein Endbenutzer seine gewünschte mobile ERP-Applikation auf Basis wiederverwendbarer Elemente entwickeln. Die wiederverwendbaren Elemente sind Teil einer *domänenspezifischen Sprache* für mobile ERP-Applikationen. Diese Sprache legt die Elemente inkl. ihrer Parametrisierungsmöglichkeiten sowie ihrer Verknüpfungsregeln fest.

Die im ersten Entwicklungsschritt erstellte Applikationsspezifikation dient im zweiten Entwicklungsschritt als Grundlage für die Generierung des ausführbaren Programmcodes der mobilen ERP-Applikation. Hierzu soll ein *Codegenerator* bereitgestellt werden, welcher für die Elemente der domänenspezifischen Sprache zugehörige Implementierungsbestandteile zur Verfügung stellt.

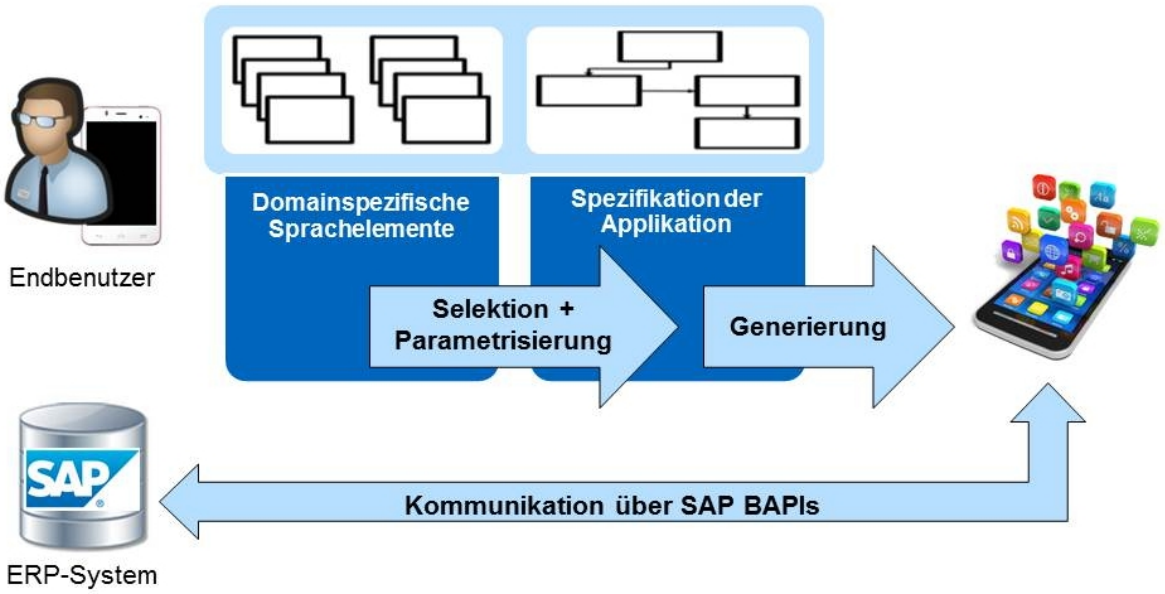

Abbildung 7-1: Grobes Architekturkonzept des Entwicklungswerkzeuges
Quelle: Eigene Darstellung

Diesem Grobkonzept folgend umfasst die Architektur die folgenden drei Bestandteile:

(1) eine domänenspezifische Sprache zur Spezifikation der mobilen ERP-Applikationen

(2) einen Codegenerator zur Generierung der ausführbaren mobilen ERP-Applikationen auf Basis der Spezifikationen

(3) das eigentliche Entwicklungswerkzeug für Endbenutzer mit zugehöriger Benutzungsschnittstelle

Bei dem vorgestellten Grobkonzept handelt es sich um einen *modellbasierten Entwicklungsansatz*. Die mobilen ERP-Applikationen werden durch die Nutzung der domänenspezifischen Sprache auf einer abstrakteren Modellebene spezifiziert. Die Überführung in die Umsetzungsebene erfolgt anschließend vollautomatisiert durch einen Codegenerator.

Zur Kommunikation mit dem SAP-ERP-System sollen entsprechende BAPIs verwendet werden, um auf die gewünschten Daten und Funktionalitäten zugreifen zu können.

7.2 Vorgehen bei der Gestaltung des Entwicklungswerkzeuges

Das Vorgehen bei der Gestaltung des Entwicklungswerkzeuges ist durch die Erstellung von Prototypen geprägt. Unter einem *Prototypen* wird ein repräsentatives Model oder eine Simulation des finalen Systems verstanden (Warfel 2009, 3). Prototypen werden eingesetzt um Designentscheidungen zu evaluieren (Cooper et al. 2007, 70). Der Erstellungsprozess von Prototypen wird als *Prototyping* bezeichnet (Warfel 2009, 13 ff.). Prototyping ist generell durch ein inkrementelles und iteratives Vorgehensweise geprägt (Warfel 2009, 45).

Je nach Phase des Gestaltungsprozesses eignen sich Prototypen mit unterschiedlichem Funktionsumfang und unterschiedlicher Detailierungsstufe. Dieser Aspekt wird häufig durch den englischen Begriff „fidelity“ beschrieben, welcher in diesem Kontext mit dem deutschen Wort „Wiedergabetreue“ übersetzt werden kann. Die „fidelity“ eines Prototypen gibt Auskunft über den Umfang eines Prototypen in Bezug auf die finale Gestaltung und den finalen Funktionsumfang (Hacker 2013, 8). In frühen Phasen des Gestaltungsprozesses eignen sich Prototypen mit geringer Wiedergabetreue, sogenannte *low-fidelity Prototypen* (Hacker 2013, 8). Diese ermöglichen es den Erstellungsaufwand zu reduzieren und eine zügige Evaluation von Designentscheidungen zu durchzuführen (Dahm 2006, 313 f.; Hacker 2013, 8). In späteren Phasen des Gestaltungsprozesses eignen sich hingegen Prototypen mit einer höher Wiedergabetreue, so genannte *high-fidelity Prototypen* (Hacker 2013, 8). Diese versuchen in Bezug auf das Layout, die verwendeten Farbschemas, den Funktionsumfang sowie Anbindung an externe Systeme eine möglichst hohe Ähnlichkeit zum finalen System zu erlangen (Dahm 2006, 313 f.; Hacker 2013, 8).

Um die vorgestellten Architekturbestandteile zu konzipieren und prototypisch zu implementieren, werden zunächst Gestaltungsvorschläge für eine domänenspezifische Sprache für mobile ERP-Applikationen erarbeitet. Diese Gestaltungsvorschläge werden anschließend genutzt um daraus ein Konzept für eine domänenspezifische Sprache zu entwickeln. Zusätzlich wird ein Architekturentwurf für einen zugehörigen Codegenerator erstellt. Dieser soll aus einer mit Hilfe der domänenspezifischen Sprache spezifizierten mobilen ERP-Applikation eine ausführbare mobile ERP-Applikation generieren können. Um die Umsetzbarkeit der entworfenen Konzepte sowie die Umsetzung der gestellten Anforderungen prüfen zu können, wird ein high-fidelity Prototyp des Codegenerators entwickelt und anschließend im Rahmen der Evaluation verwendet.

In einem nächsten Schritt werden erste Gestaltungsvorschläge für die Benutzungsschnittstelle des gewünschten Entwicklungswerkzeugs auf Basis der vorgestellten Anforderungen gesammelt. Diese werden daraufhin in einen low-fidelity Prototypen überführt, welcher anschließend evaluiert wird. Dieser Schritt dient dazu die Gestaltungsideen bereits in einem frühen Entwicklungsstadium zu prüfen.

Die Verbesserungsvorschläge werden anschließend genutzt um einen high-fidelity Prototypen zu implementieren. Neben der optimierten Benutzungsschnittstelle enthält dieser auch die Integration mit dem prototypisch implementierten Codegenerator. Der high-fidelity Prototyp wird schließlich ebenfalls evaluiert, um weitere Verbesserungsvorschläge zu identifizieren.

Abbildung 7-2 illustriert das beschriebene Vorgehen. Insgesamt basiert die gestaltungsorientierte Forschung im Rahmen dieser Arbeit auf drei Design-Zyklen. Dabei werden im ersten Design-Zyklus die domänenspezifische Sprache und der Codegenerator fokussiert. In den beiden folgenden Design-Zyklen liegt der Fokus auf der Benutzungsschnittstelle des Entwicklungswerkzeuges. Der im letzten Design-Zyklus entwickelte high-fidelity Prototyp enthält zudem eine Integration zum prototypisch implementierten Codegenerator, so dass eine durchgängige Entwicklung von mobilen ERP-Applikationen durch Endbenutzer ermöglicht wird.

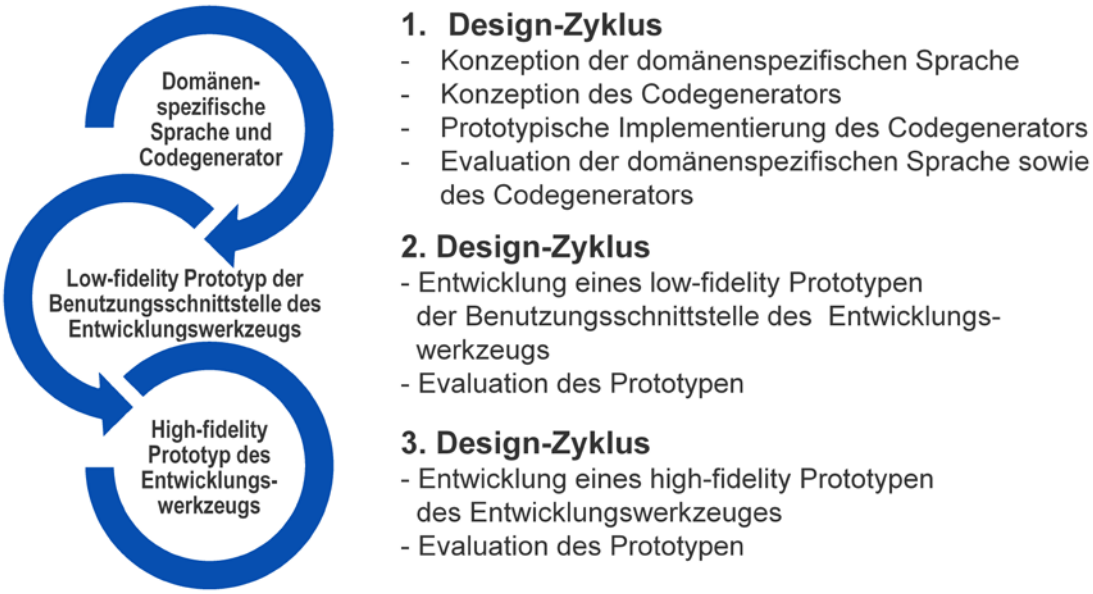

Abbildung 7-2: Design-Zyklen bei der Gestaltung des Entwicklungswerkzeuges
Quelle: Eigene Darstellung

7.3 Vorgehen bei der Evaluation

Neben der Entwicklung des Artefaktes ist die zweite Aktivität der gestaltungsorientierten Forschung die Evaluation, welche sich mit der systematischen Bewertung der entwickelten Artefakte beschäftigt (March/Smith 1995, 254; Hevner/Chatterjee 2010, 109). Das Evaluationsergebnis hilft dabei ein besseres Problemverständnis zu erlangen und hierdurch die Qualität des Artefakts sowie das Gestaltungsvorgehen zu verbessern (Hevner et al. 2004, 78). Die betrachteten Bewertungsdimensionen sind dabei die Nützlichkeit, die Qualität und die Effizienz bei der Lösung der adressierten Problemstellung (Hevner et al. 2004, 85). Zur Bewältigung der Evaluationsaufgabe stellen Hevner et al. (2004, 86) eine Reihe von Methoden vor (siehe Tabelle 7-1). Die vorgestellten Methoden werden gemäß ihrer zugrundeliegenden Vorgehensweise in beobachtende, analytische, experimentelle, testende und beschreibende Methodenkategorien unterteilt.

Tabelle 7-1: Methoden der Artefaktevaluation in der gestaltungsorientierten Forschung
Quelle: Eigene Darstellung in Anlehnung an (Hevner et al. 2004, 86), übersetzt in (Hoffmann 2009, 245)

Art	Methode	Beschreibung
Beobachtend	Fallstudie	Untersuchung des Artefakts im jeweiligen Geschäftsumfeld
	Feldstudie	Beobachtung der Nutzung des Artefakts in Projekten
Analytisch	Statische Analyse	Untersuchung der Struktur des Artefakts
	Architekturanalyse	Untersuchung der Integrationsfähigkeit des Artefakts in die technische Infrastruktur
	Optimierung	Demonstration der Optimalität des Artefakts für den Zweck oder Aufzeigen der Grenzen der Optimierung
	Dynamische Analyse	Untersuchung des Laufzeitverhaltens des Artefakts
Experimentell	Kontrolliertes Experiment	Untersuchung von Artefakteigenschaften unter kontrollierten Bedingungen
	Simulation	Ausführen des Artefakts unter Nutzung nicht-realer Daten
Testend	Funktionale Tests	Verwenden der Schnittstellen des Artefakts, um Fehler zu finden (Black Box Tests)
	Strukturelle Tests	Testen der inneren Funktionsweise des Artefakts, um Fehler zu finden (White Box Tests)
Beschreibend	Experteninterview	Begründen der Nützlichkeit des Artefakts durch Informationen aus der Wissensbasis
	Szenarios	Erstellung von Verwendungsszenarios um die Nützlichkeit des Artefakts zu zeigen

Aufgrund der dreistufigen Gestaltung des Artefakts in dieser Arbeit (siehe Kapitel 7.2) werden drei zugehörige Evaluationsphasen durchlaufen. Ziel der Evaluation ist die Prüfung der gestellten Anforderungen (siehe Kapitel 5.2) an das Artefakt. Aufgrund der unterschiedlichen Blickwinkel der Anforderungen auf das Artefakt erscheint ein *Methodenpluralismus* geeignet.

Um Anforderungen, welche sich auf den Funktionsumfang des Artefakts und die zu unterstützenden Rahmenbedingungen beziehen, z.B. die zu unterstützenden mobilen Betriebssysteme, erscheinen *funktionale Tests* zweckdienlich. Hierdurch können die geforderten Funktionen isoliert geprüft werden. Um Anforderungen zu prüfen, welche sich auf nicht-funktionale Anforderungen beziehen und ohne Einbeziehung von Endbenutzer validiert werden können, erscheint eine *statische Analyse* als geeignete Methodik. Wenn für die Prüfung einer Anforderung hingegen Endbenutzer als Probanden notwendig sind, wird ein *kontrolliertes Experiment* als Evaluationsmethode herangezogen. In diesem Fall ist jedoch ein funktionsfähiger Prototyp erforderlich. Für Zwischenevaluationen, ohne einen funktionsfähigen Prototypen, erscheinen hingegen semi-strukturierte Experteninterviews mit potenziellen Endbenutzern geeignet. Tabelle 7-2 gibt eine Übersicht über die zu prüfenden Evaluationsaspekte (EA), welche mit den identifizierten Anforderungen (A) korrelieren (siehe Kapitel 5.2) sowie die jeweils verwendete Evaluationsmethode.

Tabelle 7-2: Evaluationsaspekte und zugehörige Methoden
Quelle: Eigene Darstellung

Anforderung	Evaluationsaspekt	Beschreibung	Evaluationsmethode
A1	EA1	Unterstützte mobile Betriebssysteme der entwickelten mobile ERP-Applikationen	Funktionaler Test
A2	EA2	Datenkommunikation der entwickelten mobile ERP-Applikationen mit einem SAP-ERP-System	Funktionaler Test
A3	EA3	Funktionsumfang der mobilen ERP-Applikationen	Funktionaler Test
A4	EA4	Trainingsaufwand zur Nutzung des Entwicklungswerkzeuges	Kontrolliertes Experiment
A5	EA5	Schrittweiser Entwicklungsprozess	Statische Analyse
A6	EA6	Nutzung von Layoutschablonen für mobile ERP-Applikationen	Funktionaler Test
A7	EA7	Dauer des Entwicklungsvorganges	Kontrolliertes Experiment
A8	EA8	Konfigurationsmöglichkeit für die SAP-ERP Verbindungseinstellung	Statische Analyse
A9	EA9	Gestaltungsspielraum bei der Entwicklung der Benutzungsschnittstelle der mobilen ERP-Applikationen	Statische Analyse
A10	EA10	Nutzung wiederverwendbarer Codeschablonen	Statische Analyse
A11	EA11	Unterstützte mobile Betriebssysteme des Entwicklungswerkzeuges	Funktionaler Test
A12	EA12	Erweiterbarkeit des Entwicklungswerkzeuges hinsichtlich der Unterstützung weiterer mobiler Betriebssysteme	Statische Analyse
A13	EA13	Verteilungsfunktion für die entwickelten mobilen ERP-Applikationen	Funktionaler Test

Generell ist noch anzuführen, dass der größte Teil der Evaluationsaktivitäten nicht auf konzeptueller Ebene möglich ist. Daher werden die vorgestellten konzeptuell beschriebenen Bestandteile des Artefaktes dieser Arbeit jeweils prototypisch umgesetzt und die Evaluationsaktivitäten an den entwickelten *Prototypen* durchgeführt. Hierdurch wird gleichzeitig die *Umsetzbarkeit* der vorgestellten Konzepte demonstriert.

8 Gestaltung einer domänenspezifischen Sprache für mobile ERP-Applikationen

Im vorliegenden Kapitel wird die Gestaltung der domänenspezifischen Sprache für mobile ERP-Applikationen beschrieben. Hierzu werden zunächst einige theoretische Grundlage bzgl. domänenspezifischer Sprachen vorgestellt. Der Schwerpunkt liegt hierbei in der Beschreibung des Vorgehens zum Entwurf einer eigenen domänenspezifischen Sprache. Anschließend werden verwandte Forschungsarbeiten vorgestellt, welche ebenfalls eine domänenspezifische Sprache gestalten. Die dabei erlangten Erkenntnisse werden daraufhin genutzt, um die domänenspezifische Sprache für mobile ERP-Applikationen zu entwickeln.

8.1 Theoretische Grundlagen

Im Folgenden werden wichtige Begriffe und Elemente von domänenspezifischen Sprachen sowie eine Vorgehensweise zur Gestaltung eigener domänenspezifischer Sprachen erläutert.

8.1.1 Begriffliche Klärung

Eine gängige Herangehensweise zur Produktivitätsverbesserung in der Softwareentwicklung ist die Steigerung des Abstraktionsniveaus (Kelly 2005, 64; Balzert 2009, 26 ff.). Unter einer *Abstraktion* wird generell eine Verallgemeinerung und das Absehen von Besonderheiten und Unwesentlichem verstanden (Balzert 2009, 26). Das Gegenteil der Abstraktion ist die Konkretisierung (Balzert 2009, 26). Ein Beispiel für eine Abstraktion aus der Softwareentwicklung ist die stufenweise Steigerung des Abstraktionsniveaus von Maschinensprachen über Assemblersprachen hin zu höheren Programmiersprachen wie C oder Java (Kelly 2005, 64; Tolvanen/Kelly 2004). In diesen Fällen werden aus Konstrukten der höheren Ebene Konstrukte der jeweils darunterliegenden Ebene erzeugt.

Eine *domänenspezifische Sprache* repräsentiert eine Ansatz, um das Abstraktionsniveau der Softwareentwicklung zu erhöhen (Kelly 2005, 64; Tolvanen 2006, 9). Aufgrund ihres hohen Abstraktionsniveaus werden domänenspezifische Sprachen teilweise auch als *High-Level Spezifikationssprachen* bezeichnet (Tolvanen/Kelly 2004, 30; Kelly 2010, 64). Traditionelle Programmiersprachen wie bspw. C oder Java verwenden Konstrukte der *Umsetzungsdomäne*. Beispiele hierfür sind Variablen, Schleifen oder Objektklassen. Ein Vorteil dieser Konstrukte ist, dass sie unabhängig von einer bestimmten Fachdomäne einsetzbar sind. Aufgrund dieser Unabhängigkeit und der daraus resultierenden übergreifenden Einsetzbarkeit werden traditionelle Programmiersprachen auch als general-purpose languages (GPLs) bezeichnet (Mernik et al. 2005, 316). Die Nutzung von GPLs erfordert i.d.R. eine professionelle Ausbildung in der Softwareentwicklung. Aus dieser Perspektive besteht die Aufgabe eines professionellen Softwareentwicklers im Rahmen der Softwareentwicklung darin, die Anforderungen aus der Fachdomäne in zugehörige Konstrukte der Umsetzungsdomäne zu überführen.

Eine domänenspezifische Sprache verwendet hingegen Konstrukte und Regeln aus der *Fachdomäne* (Mernik et al. 2005, 317; Tolvanen 2006, 9). Sie beruht darauf, dass jede Fachdomäne spezifische Abstraktionen, Konstrukte und Regeln besitzt (Tolvanen/Kelly 2004, 30). Die Konstrukte repräsentieren die Objekte aus der Fachdomäne. Beispiele aus dem Versicherungswesen sind „Risiko“, „Bonus“ oder „Schaden“ (Tolvanen/Kelly 2004, 30). Regeln stellen die Korrektheit der erstellen Spezifikationen sicher und verhindern hierdurch die Erstellung unzulässiger Entwürfe (Tolvanen/Kelly 2004, 32). Ein Beispiel aus dem Versicherungswesen könnte sein, dass ein gutes Versicherungsprodukt immer eine Prämienregelung besitzt. Aus diesem Grund kann die domänenspezifische Sprache die Definition einer Prämienregel als Voraussetzung für das Anlegen eines Versicherungsproduktes festlegen (Tolvanen/Kelly 2004, 30).

Einige Publikationen verwenden den Begriff *domänenspezifische Modellierungssprache* (engl. domain-specific modelling language (DSML)). Vertreter aus diesem Bereich verwenden i.d.R. eine grafische Notation zur Spezifikation der Applikationen (vgl. z.B. (Kelly 2005, 64 ff.; Pühler 2011, 144 ff.)). In der vorliegenden Arbeit wird der Begriff „domänenspezifische Sprache“ verwendet, um auch Sprachen mit einer nicht-grafischen Notationsform zu behandeln.

Die Überführung der mit Hilfe einer domänenspezifischen Sprache spezifizierten Applikationen in zugehörige Konzepte der Umsetzungsdomäne wird durch einen *Codegenerator* durchgeführt. Dieser enthält eine Menge an Transformationsvorschriften, wie die Konstrukte der domänenspezifischen Sprache in zugehörige Codefragmente zu überführen sind. Abbildung 8-1 skizziert die beschriebenen Abstraktionsebenen im Kontext einer domänenspezifischen Sprache. Der Codegenerator enthält Transformationsregeln, wie die verfügbaren fachlichen Konstrukte in Programmierungs-Konstrukte der Umsetzungsdomäne abzubilden sind. Diese können anschließend über einen Compiler oder Interpreter in Maschinencode überführt werden.

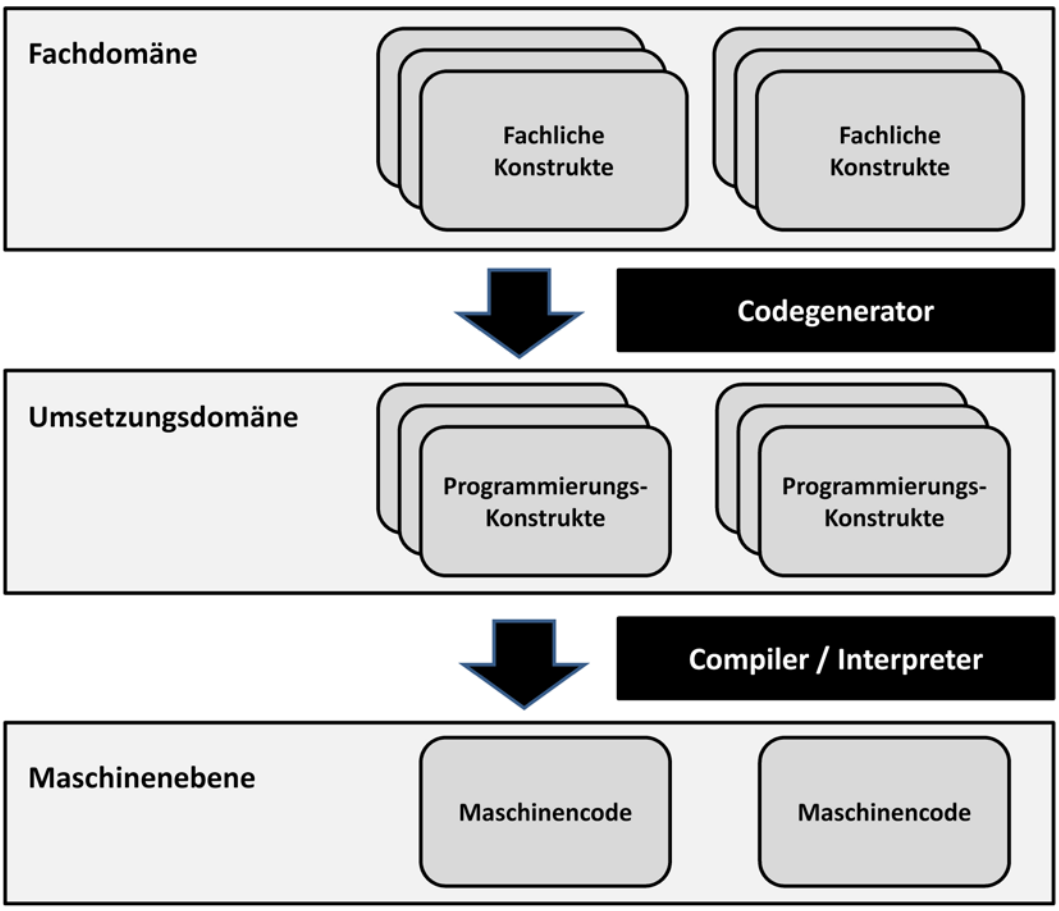

Abbildung 8-1: Abstraktionsebenen im Kontext domänenspezifischer Sprachen
Quelle: Eigene Darstellung

Aufgrund der Verwendung der Semantik aus der Fachdomäne sind die Konstrukte einer domänenspezifischen Sprache für Endbenutzer verständlich. Zudem erfordert die Nutzung einer domänenspezifischen Sprache aufgrund der Abstraktion von der Umsetzungsdomäne keine professionelle Ausbildung in der Softwareentwicklung. Im Gegensatz zu Programmiersprachen der Umsetzungsdomäne sind domänenspezifischen Sprachen auf ausgewählte Problemstellungen fokussiert und besitzen daher einen vergleichsweise geringen Gestaltungsspielraum (Mernik et al. 2005, 317). Jedoch wird durch das höhere Abstraktionsniveau neben der genannten Produktivitätssteigerung auch eine Kapselung der Komplexität, eine höhere Ausdruckskraft und eine einfachere Nutzung erreicht (Voelter 2013, 70 ff.). Zudem kann die Verwendung geprüfter Codefragmente durch den Codegenerator zu einer höheren Qualität der entwickelten Applikationen führen (Kelly 2005, 64; Tolvanen 2006, 9).

Auch modellgetriebene Entwicklungsansätze aus dem Softwareengineering verfolgen das Ziel einer höheren Abstraktionsebene (Balzert 2009, 79 f.). Jedoch nutzen die zugehörigen Ansätze im Gegensatz zu domänenspezifischen Sprachen weiterhin Konzepte der Umsetzungsdo-

mäne (Tolvanen 2006, 9). Ein Beispiel ist die *Model Driven Architecture* [35](MDA) der Object Management Group (OMG). MDA nutzt die die verbreitete Modellierungsnotation *Unified Modeling Language*[36] (UML) und beinhaltet Ansätze zur Umwandung von UML-Modellen höherer Abstraktionsebenen auf UML-Modelle niedrigerer Abstraktionsebenen. Die übergreifende Zielsetzung der MDA ist die Plattformunabhängigkeit und die Standardisierung von Modellformaten der Entwicklungswerkzeuge verschiedener Hersteller (Tolvanen 2006, 9). Die Zielgruppe von MDA sind professionelle Softwareentwickler, da die Kenntnis und Nutzung der verfügbaren UML-Modelltypen bei Endbenutzern nicht vorausgesetzt werden kann. Domänenspezifische Sprachen erlauben jedoch auch Endbenutzern die Entwicklung von Softwareartefakten. Zudem ist die Plattformunabhängigkeit nicht die primäre Zielsetzung bei domänenspezifischen Sprachen; sie kann jedoch durch die Bereitstellung unterschiedlicher Codegeneratoren erreicht werden. (Tolvanen 2006, 11 f.).

8.1.2 Vorgehen bei der Gestaltung domänenspezifischer Sprachen

Nach Tolvanen und Kelly umfasst die Definition einer domänenspezifischen Sprache die folgenden drei Aspekte (Tolvanen/Kelly 2004, 32):

(1) Domänenkonzepte

(2) Domänenregeln

(3) Notation zur Spezifikation der Applikationen

Die *Domänenkonzepte* repräsentieren die bereitgestellten Elemente der Sprache, welche zur Entwicklung genutzt werden können. Die *Domänenregeln* definieren die Anwendungsmöglichkeiten und -einschränkungen der Domänenkonzepte. Sie können beispielsweise festlegen, welche Domänenkonzepte miteinander kombiniert werden dürfen oder welche Voraussetzungen bei der Verwendung eines bestimmten Domänenkonzeptes erfüllt sein müssen. Die *Notation* beschreibt die Repräsentationsform der mithilfe der domänenspezifischen Sprache spezifizierten Applikationen. Im Falle einer grafischen Notation umfasst dies die Syntax und Semantik der eingesetzten Symbole; im Fall einer textbasierten Notation wird hingegen die Syntax und Semantik der einzelnen Textbausteine.

Mernik et al. empfehlen folgende Phasen zur Entwicklung einer domänenspezifischen Sprache (Mernik et al. 2005, 320 ff.):

(1) Entscheidung

(2) Analyse

(3) Design

(4) Implementierung

(5) Einsatz

[35] http://www.omg.org/mda, zugegriffen am 13.01.2014
[36] http://www.uml.org, zugegriffen am 10.01.2014

Die Phase *Entscheidung* geht der Fragestellung nach, ob die Entwicklung einer domänenspezifischen Sprache sinnvoll ist. Nach Mernik et al. (2005) gibt es keine festgelegten Regeln zur Klärung dieser Fragestellung. Nach ihrer Ansicht sollten bei der Beantwortung der Frage insbesondere die ökonomischen Aspekte der Softwareentwicklung und die Zielgruppe berücksichtigt werden. In Bezug auf die Zielgruppe nennen die Autoren Endbenutzer als die primäre Zielgruppe domänenspezifischer Sprachen (Mernik et al. 2005, 321).

In der Phase *Analyse* wird die fokussierte Fachdomäne untersucht und dabei das erforderliche Domänenwissen identifiziert. Dabei können abhängig von der Fachdomäne und den Rahmenbedingungen unterschiedliche Quellen untersucht werden. Mögliche Quellen sind existierende Dokumentationen, existierender GPL-Programmcode, technische Dokumentationen, Interviews mit Fachexperten oder Kundenumfragen (Mernik et al. 2005, 323). Das formale Ergebnis einer Domänenanalyse besteht aus den folgenden Bestandteilen (Mernik et al. 2005, 324):

- **Domänendefinition:** beschreibt den Fokus bzw. den Umfang der domänenspezifischen Sprache.
- **Domänenterminologie:** beschreibt das verwendete Vokabular.
- **Domänenkonzepte:** umfasst die fokussierten Konzepte inkl. deren Beschreibung.
- **Domänenregeln:** beschreiben die Ausprägungsformen der einzelnen Domänenkonzepte sowie die Abhängigkeiten zwischen unterschiedlichen Domänenkonzepten.

Insbesondere die Domänendefinition wird als Herausforderung erachtet (Mernik et al. 2005, 317). Hierbei gilt es einen Kompromiss zwischen einem möglichst hohen Funktionsumfang und einer aus einer geringeren Komplexität resultierenden einfacheren Nutzung zu finden. Um die Schwierigkeit einer Domänendefinition zu demonstrieren, nennen Mernik et al. die Programmiersprache COBOL als Beispiel für unterschiedliche Auffassungen über die Abgrenzung einer Domäne. Die Autoren beschreiben, dass einige Personen COBOL als domänenspezifische Sprache betrachten; die zugehörige Domäne wäre „Geschäftsapplikationen“ (Mernik et al. 2005, 317). Dagegen spricht jedoch, dass COBOL analog zu anderen GPLs auch mit typischen Programmierungs-Konstrukten wie Variablen, Bedingungen und Schleifen arbeitet und damit prinzipiell auch Applikationen jenseits der Unterstützung von Geschäftsprozessen umgesetzt werden können.

In der Phase *Design* bestehen nach Mernik et al. (2005) zwei grundsätzliche Optionen. Die erste Option ist die Erweiterung einer existierenden Sprache, um Domänenkonzepte und Domänenregeln. Der Vorteil ist hierbei, dass Anwender die Basissprache bereits kennen. Da existierende Werkzeuge der Basissprache verwendet werden können, ist der Umsetzungsaufwand oftmals geringer als bei der zweiten Option. Nachteilig ist jedoch, dass die Kenntnis und Fähigkeit zur Nutzung der Basissprache eine Voraussetzung der domänenspezifischen Sprache ist. Die zweite Option ist die Entwicklung einer unabhängigen domänenspezifischen Sprache, welche keine direkte Verbindung zu einer GPL hat. Die Entwicklung einer solchen domänenspezifischen Sprache wird als aufwändiger betrachtet. Durch die Unabhängigkeit von einer GPL kann jedoch der potenzielle Anwenderkreis erhöht werden (Mernik et al. 2005, 326 f.).

Nachdem eine Designoption gewählt wurde, erfolgt in einem nächsten Schritt die Spezifikation des Designs. Hierbei unterscheiden Mernik et al. zwischen einer informalen und einer formalen Design Spezifikation (Mernik et al. 2005, 327). Die *informale Design Spezifikation* kann aus einer Prosatext-Beschreibung bestehen, welche durch beispielhafte Skizzen von DSL-Programmen ergänzt wird. Hingegen erfordert eine *formale Design Spezifikation* eine formale Notation. Beispiele für verwendete formale Notationen sind reguläre Ausdrücke, Attribut-Grammatiken oder abstrakte Zustandsmaschinen (Mernik et al. 2005, 327).

In der Phase *Implementierung* wird die Umsetzung der domänenspezifischen Sprache festgelegt. In Bezug auf die in Abbildung 8-1 illustrierten Abstraktionsebenen wird im Rahmen der Implementierung die Transformation von der Fachdomäne auf die Umsetzungsebene festgelegt. Dies erfolgt durch die Entwicklung eines geeigneten Codegenerators. Ein großer Vorteil domänenspezifischer Sprachen ist die Wiederverwendung. Daher sollte bereits beim Design der Domänenkonzepte die Umsetzbarkeit in Form von wiederverwendbaren Programmierungs-Konzepte bzw. Codeschablonen geprüft werden.

Die Phase *Einsatz* behandelt die eigentliche Nutzung der domänenspezifischen Sprache und zugehörigen Implementierung. Hierzu kann es notwendig sein, dass geeignete Werkzeuge bereitgestellt werden müssen, welche die benutzerfreundliche Nutzung der domänenspezifischen Sprache ermöglichen.

8.2 Ausgewählte Beispiele domänenspezifischer Sprachen

Im Folgenden werden drei Forschungsarbeiten aus dem Umfeld domänenspezifischer Sprachen vorgestellt. Dabei werden die Arbeiten hinsichtlich der im vorherigen Kapitel (Kapitel 8.1.2) vorgestellten Phasen zum Entwurf einer domänenspezifischen Sprache untersucht. Dabei sollen insbesondere folgenden Aspekte untersucht werden:

- Wie ist die fokussierte Domäne definiert bzw. wie wird sie abgegrenzt?
- Welche Domänenkonzepte werden verwendet?
- Welche Domänenregeln werden verwendet?
- Welche Notation wird zur Spezifikation der Applikationen verwendet?
- Welche Werkzeugunterstützung wird bereitgestellt?

8.2.1 Automotive Service Modellig Language

Maximilian Pühler (2011) hat im Rahmen seiner Dissertation am Lehrstuhl für Wirtschaftsinformatik (I17) der Technischen Universität München „einen Ansatz zur modellgetriebenen Gestaltung, Definition und prototypischen Umsetzung von Automotive Services“ (Pühler 2011, I) entwickelt. Kernbestandteil seines vorgestellten Ansatzes ist einen domänenspezifische Modellierungssprache für Automotive Services, die sogenannte Automotive Service Modelling Language (ASML). Unter *Automotive Services* werden in der Arbeit mobile

Mehrwertdienste im Fahrzeug verstanden, welche IT-gestützt umgesetzt werden. Beispiele sind fahrzeugbezogene Fahrerassistenzsysteme und Infotainmentsysteme.

Die im Rahmen der Dissertation fokussierte Domäne sind Automotive Services. Der Anwendungsschwerpunkt liegt in der Entwicklung neuartiger Automotive Services durch eine Kombination wiederverwendbarer Bausteine. Eine weitere Abgrenzung der Domäne erfolgt nicht. ASML verwendet folgende Domänenkonzepte:

- **Bricks:** repräsentieren wiederverwendbare, domänenspezifische Funktionalitäten in einer technisch abstrahierten Form. Neben der abstrahierten Form existiert eine zugehörige technische Implementierung.
- **Stubs:** haben die gleichen Eigenschaften wie Bricks; besitzen jedoch keine technische Implementierung.
- **Composites:** repräsentieren komplexere Bausteine in Form von wiederverwendbaren ASML Teilmodellen.
- **Streams:** repräsentieren funktionale und logische Übergänge von einem Brick oder Stub zu einem anderen.

Neben diesen Modellierungsbausteinen definiert ASML eine Reihe von *Bedienkonzepten* zur Repräsentation der grafischen Benutzungsstelle der modellierten Automotive Services und somit zur Interaktion mit dem Benutzer. Beispiele für Bedienkonzepte sind „Liste“, „Text“ oder „Browser“ (Pühler 2011, 160 ff.). Die eigentliche Funktionalität wird durch sogenannte Funktionskonzepte repräsentiert. In vielen Fällen werden Funktionskonzepte und Bedienkonzepte zu einem Brick zusammengefasst. Ausnahmen sind u.a. die Funktionskonzepte „GPS Position“ oder „Spracherkennung“, welche keine grafische Benutzungsschnittstelle benötigen (Pühler 2011, 164 ff.).

ASML verwendet einen datenflussorientieren Modellierungsansatz. Die zentralen Modellierungsbausteine sind Bricks und Streams. Bricks stellen die eigentliche Funktionalität bereit, während der Kontrollfluss durch der Applikation durch die verwendetet Streams gesteuert wird. Bricks besitzen entsprechende Input- und Output-Konnektoren, um ihre Verbindung untereinander zu ermöglichen. Neben den Konnektoren besitzen Bricks u.a. auch einen Titel, ein grafisches Symbol und können über bereitgestellte Eigenschaftswerte parametrisiert werden. Beispiele für Bricks sind „GPS Position“, „GeoCoder“, „Kalender (Google)“ oder „Kontakte (Google)“ (Pühler 2011, 163 ff.). ASML definiert durch Regeln u.a. die Verbindungsmöglichkeiten von Bricks durch Streams, die Aktivierung von Eigenschaftswerten bei Bricks sowie die Spezifikation der synchronen- und asynchronen Ausführung von Streams (Pühler 2011, 157 ff.).

ASML verwendet ein grafisches Notationsformat. Hierbei wird für jeden Baustein der Modellierungssprache ein Symbol definiert (Pühler 2011, 150 ff.). Auf technischer Ebene existiert zudem eine zugehöriges XML-Format, welches die modellierten Automotive Service Applikationen beschreibt (Pühler 2011, 203 ff.).

Um die Nutzung von ASML durch Endbenutzer zur ermöglichen wurde ein zugehöriges Modellierungswerkzeug, die sogenannte *ASML Workbench* realisiert (Pühler 2011, 175 ff.). Die ASML Workbench basiert auf der sogenannten Eclipse Rich Client Plattform (RCP). Eclipse RCP ist ein Framework, welches die Umsetzung Eclipse-basierter grafischer Modellierungswerkzeuge vereinfacht. Die ASML Workbench repräsentiert einen grafischen Editor, mit welchem u.a. Bricks per Drag & Drop im Modell positioniert werden können und in Form von Streams durch grafische Verbindungslinien verknüpft werden können. Zudem wird ein Assistent bereitgestellt, um Bricks bei ihrer Erstellung zu parametrisieren (Pühler 2011, 182 ff.). Abbildung 8-2 illustriert ein beispielhaftes ASML-Modell in der ASML Workbench.

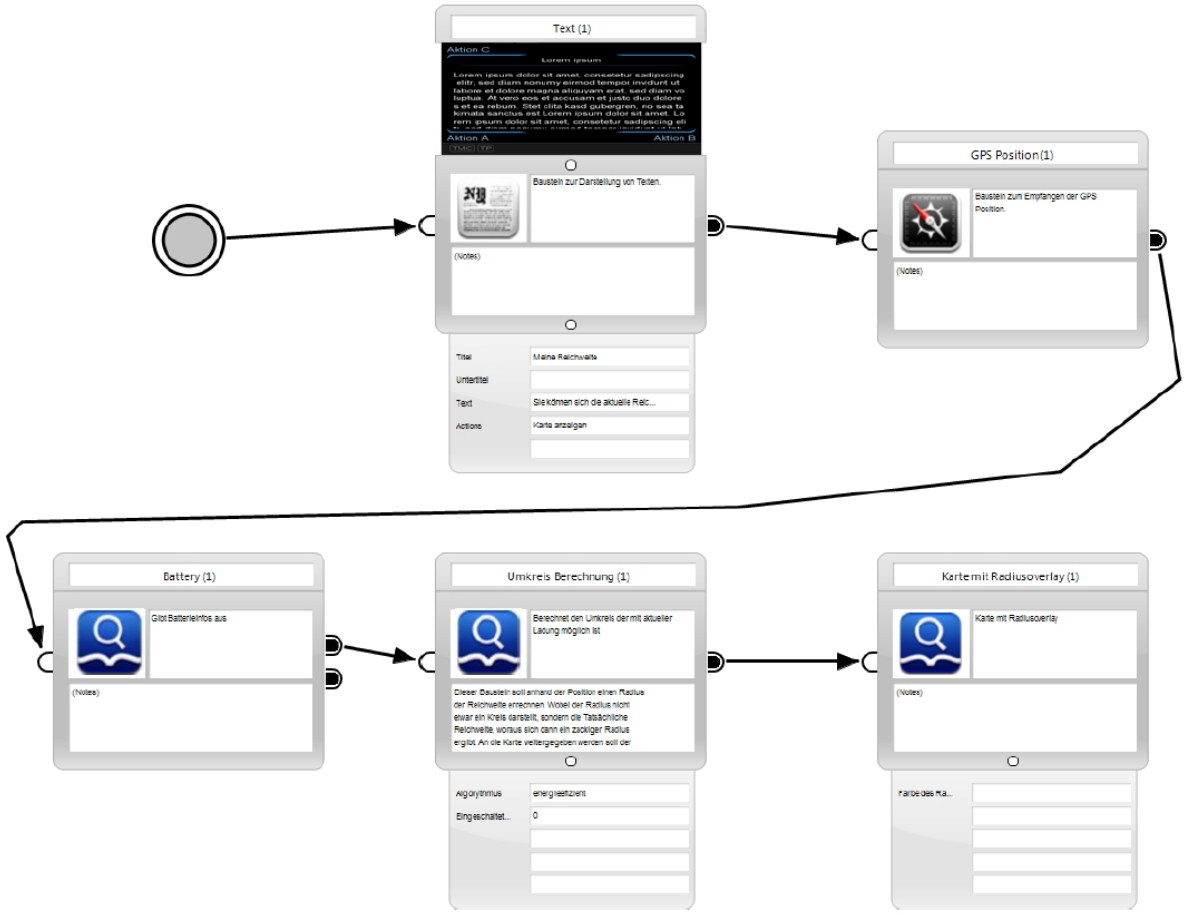

Abbildung 8-2: Exemplarisches domänenspezifisches Modell der Automotive Service Modelling Language
Quelle: (Pühler 2011, 234)

8.2.2 Mobia Framework

Florence Tlu Balagtas-Fernandez (2010) hat im Rahmen ihrer Dissertation an der Ludwig-Maximilians-Universität München einen Ansatz zur modellgetriebenen Entwicklung von

Android Applikationen für die Domäne *mHealth* entwickelt, das sogenannte Mobia Framework. Unter mHealth versteht die Autorin eine Kombination aus mobilen Endgeräten, medizinischen Sensoren und Kommunikationstechnologien (Balagtas-Fernandez 2010, 2). Typische Anwendungsfälle dieser Domäne sind die Diagnose von Krankheiten, die Überwachung von Krankheitsverläufen aus der Ferne oder die Prävention von Krankheiten durch Aufklärung über mobile Endgeräte (Balagtas-Fernandez 2010, 2). Ein besonderer Schwerpunkt der Forschungsarbeit liegt auf der Entwicklung patientenspezifischer mobiler Applikationen durch Endbenutzer. Beispiele für die fokussierte Endbenutzergruppe der Arbeit sind Ärzte und Krankenschwestern.

Das *Mobia Framework* besteht aus einem Entwicklungswerkzeug, einer domänenspezifische Sprache sowie einem zugehörigen Codegenerator. Mit Hilfe des Entwicklungswerkzeuges *Mobia Modeler* kann der Endbenutzer eine neue mobile Applikation über Drag & Drop von bereitgestellten Komponenten spezifizieren. Welche Komponenten jeweils zur Verfügung gestellt werden, wird durch einen Assistenten beim Anlegen eines neuen Entwicklungsprojektes bestimmt. Dieser fragt u.a. nach dem Typ der Applikation und stellt daraufhin geeignete Komponenten bereit. Die Domänenkonzepte der Mobia DSL sind unterschiedliche Komponenten, welche durch Attribute parametrisiert werden können. Mithilfe von *Structure Components* können die Bildschirmmasken sowie der Kontrollfluss der Applikation spezifiziert werden (Balagtas-Fernandez 2010, 126 ff.). Hingegen werden bereitgestellte Dienste des Android Betriebssystems durch *Basic Components* spezifiziert (Balagtas-Fernandez 2010, 131 ff.). Beispiele sind die Nutzung von Funktionalitäten der Kamera oder des Mikrophones. Domänenspezifische Konstrukte werden hingegen durch *Special Components* abgebildet (Balagtas-Fernandez 2010, 134 ff.). Beispiele sind ein Fitness-Ratgeber oder ein Ernährungs-Tagebuch. Zur Repräsentation von externen, medizinischen Sensoren werden sogenannte *Sensor Components* verwendet (Balagtas-Fernandez 2010, 137 ff.). Ein Beispiel wäre das Abfragen der Werte eines Blutdruckmessgerätes über eine drahtlose Kommunikationsschnittstelle. Durch zugehörige Domänenregeln wird definiert, wie die verschiedenen Komponententypen innerhalb einer Applikation verwendet und kombiniert werden können. Der Mobia Modeler stellt eine grafische Notationsform bereit. Zudem existiert eine zugehörige XML-basierte Notation, welche durch den Mobia Modeler erzeugt wird. Schließlich überführt der Codegenerator, der sogenannte *Mobia Framework Processor*, die in XML spezifizierte Applikation bei der Generierung in eine funktionstüchtige Android Applikation. Abbildung 8-3 illustriert eine Reihe von Special Components der Mobia DSL mit Hilfe der grafischen Notation und zugehöriger XML-Notation.

Abbildung 8-3: Exemplarische domänenspezifische Konzepte des Mobia Framworks
Quelle: (Balagtas-Fernandez 2010, 135)

8.2.3 Widget Compostion Platform

Michael Spahn (2010) hat im Rahmen seiner Dissertation an der Universität Siegen einen Ansatz entwickelt, der es Endbenutzern ohne Programmierkenntnisse ermöglicht, Daten aus unterschiedlichen Backendsystemen in eine eigene Applikation zusammenzuführen. Hierdurch sollen die spezifischen Informationsbedürfnisse der Endbenutzer erfüllt werden. Als beispielhaften Anwendungsfall nennt der Autor den Bedarf, die Anfragen eines Kunden aus einem CRM-System und die Stammdaten des Kunden aus einem ERP-System in einer gemeinsamen Ansicht darzustellen (Spahn 2010, 109).

Die Domäne der Forschungsarbeit ist das betriebliche Informationsmanagement und hier insbesondere die Informationsbeschaffung aus heterogenen Backendsystemen. Als Domänenkonzepte kommen Services, Mashups und Widgets zum Einsatz. *Services* repräsentieren einen parametrisierbaren Zugriff auf eine Datenschnittstelle eines Backendsystems. Unabhängig vom jeweiligen Backendsystem und dessen Spezifika stellen sie einen einheitlichen Datenzugriff sicher. Ein *Mashup* kombiniert einen Service mit einer zugehörigen Benutzungsschnittstelle. Ein *Widget* ist eine aus mehreren Mashups bestehende Applikation, welche über eine Widget-Laufzeitumgebung auf dem Anwenderdesktop integriert werden kann. Die Domänenregeln definieren u.a. wie ein Service in ein Mashup integriert wird und wie Mashups untereinander über ein sogenanntes „Wiring“ verknüpft werden können (Spahn 2010, 114 ff.).

Als Werkzeugunterstützung für Endbenutzer wird im Rahmen der Dissertation die Mashup-Umgebung *Widget Composition Platform (WCP)* und der Query-Editor *Semantic Query Design (SQD)* entwickelt. WCP stellt eine webbasierte Entwicklungsumgebung für Mashups zur Verfügung. Zusätzlich enthält WCP eine Exportfunktion für Widgets sowie Funktionalitä-

ten zur Verwaltung des Service Repositories. Die Entwicklung von Widgets findet über einen WYSIWYG-Ansatz statt. Hierbei werden beispielsweise zwei Mashups über ihre Eingangs- und Ausgangsports miteinander verknüpft (Spahn 2010, 117 ff.). Abbildung 8-4 illustriert eine exemplarische Bildschirmaufnahme der Benutzungsschnittstelle des WCP.

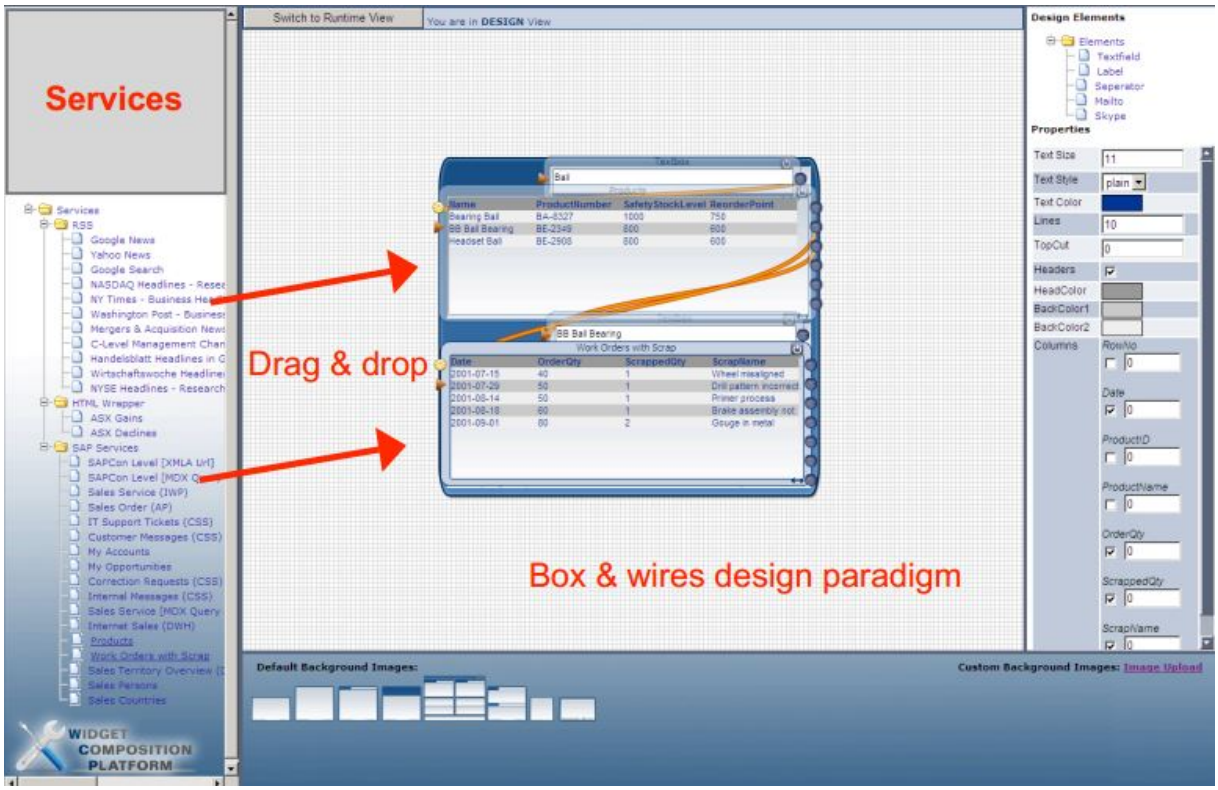

Abbildung 8-4: Exemplarische Bildschirmansicht der Benutzungsschnittstelle der Widget Composition Platform

Quelle: (Spahn et al. 2008b, 19)

Das zweite Werkzeug, der SQD, ermöglicht es Endbenutzern ohne Kenntnisse der technischen Details der Backendsysteme und Datenabfragesprachen (wie bspw. SQL) eigene Services zur Datenabfrage aus Backendsystemen zu entwickeln. Hierzu wird eine ontologiebasierte Abstraktionsschicht, die Business Level Ontologie, eingeführt. Die Business Level Ontologie-Entitäten verbergen die technischen Details der Backendsysteme vor den Endbenutzern und fokussieren sich stattdessen auf die geschäftsorientierten Entitäten sowie ihrer Relationen. Ziel ist die Bereitstellung eines globalen Datenmodells auf Basis der Business Level Ontologie, dessen technische Datenbasis auf mehreren, heterogenen Backendsystemen basiert. Auf Basis der BO wird eine Abfragesprache definiert. Die hiermit spezifizierten Abfragen werden zum Ausführungszeitpunkt über einen Inferenzmechanismus in technische Abfragen des jeweiligen Backendsystems transformiert. Die Benutzungsschnittstelle von SQD stellt die Entitäten des Business Level Ontologie grafisch dar. Um eine Abfrage zu entwickeln können u.a. die notwendigen Business Level Ontologie-Entitäten selektiert und parametrisiert werden sowie deren Relationen aktiviert werden (Spahn 2010, 132 ff.).

8.2.4 Fazit und Diskussion

Obgleich die drei vorgestellten domänenspezifischen Sprachen unterschiedliche Domänen fokussieren, haben sie in Bezug auf ihre formale Spezifikation und Werkzeugunterstützung viele Gemeinsamkeiten. Alle drei Implementierungen folgen einem komponentenorientierten Ansatz; in allen drei Sprachen werden die Konzepte in Form von wiederverwendbaren, parametrisierbaren Softwarekomponenten umgesetzt. Sie nutzen eine visuelle Programmierung, bei welcher die verfügbaren Komponenten über Drag & Drop ausgewählt und mit anderen Komponenten über einen Konnektor verbunden werden. Als Entwicklungswerkzeug für Endbenutzer setzen ASML und das Mobia Framework auf die Entwicklungsumgebung Eclipse, während die Widget Composition Platform eine Webbrowser-basierte Entwicklungsumgebung einsetzt. Zudem setzen alle drei Werkzeuge teilweise auch eine formularbasierte Programmierung ein, um beispielsweise die genutzten Komponenten zu parametrisieren oder um die Entwicklungsumgebung für die bevorstehende Aufgabe anzupassen. In Bezug auf die Benutzungsschnittstellen der verfügbaren Komponenten bieten die drei untersuchten Sprachen relativ wenig Gestaltungsfreiheit. Am ausführlichsten wird der Aspekt der Benutzungsschnittstelle von ASML adressiert. Bei der Entwicklung von ASML stellt der Autor wiederkehrende Muster bei der Umsetzung der Benutzungsschnittstelle von Automotive Services fest (Pühler 2011, 118) und setzt diese in Form sogenannter Bedienkonzepte um (Pühler 2011, 161 ff.). In Bezug auf die verwendete Notationsform setzen sowohl ASML als auch das Mobia Framework auf XML.

Unterschiede sind vor allem in Bezug auf die Gestaltungsfreiheiten vorzufinden. Beispielsweise erlaubt das Mobia Framework nur die Auswahl implementierter Komponenten, während ASML durch Stubs auch die Spezifikation nicht-implementierter Funktionalitäten erlaubt. Auf der anderen Seite ist das Abstraktionsniveau der Konzepte in ASML höher. Während ASML ausschließlich domänenspezifische Konzepte bereitstellt, besitzt das Mobia Framework auch freingranularere, umsetzungsspezifischere Konzepte, z.B. zur Ansteuerung von Sensordaten des Smartphones. Im Falle des Mobia Frameworks erhöht dies die Gestaltungsfreiheiten und damit die Möglichkeiten zur Umsetzung von mobilen Applikationen. Jedoch werden hierdurch gleichzeitig die Anforderungen an den Endbenutzer und das Fehlerrisiko erhöht. Die Widget Composition Platform fokussiert sich auf den Datenzugriff, während dieser bei den anderen beiden Sprachen nur rudimentär betrachtet wird.

8.3 Vorgehen bei der Gestaltung einer domänenspezifischen Sprache für mobile ERP-Applikationen

Das Vorgehen bei der Entwicklung einer domänenspezifischen Sprache für mobile ERP-Applikationen in dieser Arbeit orientiert sich am in Kapitel 8.1.2 vorgestellten Vorgehen von Merrnik et al. (2005). In der Phase *Entscheidung* wird zunächst das Potenzial einer domänenspezifischen Sprache für mobile ERP-Applikationen begründet. Anschließend wird in der Phase *Analyse* die Domäne definiert. In einem folgenden Schritt wird die Terminologie der Domäne behandelt und deren Konzepte und Regeln vorgestellt. Darauf aufbauend folgt in der Phase *Design* die formale Design Spezifikation der vorgestellten Konzepte und Regeln. Schließlich wird in der Phase *Implementierung* ein Codegenerator entwickelt. Hierzu werden

eine Reihe von Codeschablonen bereitgestellt, welche vom Codegenerator bei der Transformation der formalen Applikationsspezifikation in Applikationscode überführt werden. In der letzten Phase *Einsatz* wird das domänenspezifische Entwicklungswerkzeug entwickelt. Dieses macht die domänenspezifische Sprache sowie den Codegenerator für Endbenutzer nutzbar. Aufgrund des Umfangs der Beschreibung des Entwicklungswerkzeuges erfolgt dieses in einem separaten Kapitel (Kapitel 9).

8.4 Domänenspezifische Sprache für mobile ERP-Applikationen

Im Folgenden werden die im vorgehende Kapitel (Kapitel 8.3) vorgestellten Phasen zur Entwicklung einer domänenspezifischen Sprache für mobile ERP-Applikationen durchlaufen und beschreiben. Hierbei wird insbesondere in den ersten Phasen auf die Erkenntnisse vorangegangener Kapitel zurückgegriffen.

8.4.1 Entscheidung

Die Entscheidung für eine domänenspezifische Sprache für mobile ERP-Applikationen ist in der Motivation (siehe Kapitel 1.1) und Vision (siehe Kapitel 1.2) der vorliegenden Dissertation begründet. Vision der Arbeit ist es, Endbenutzern die Entwicklung mobiler ERP-Applikationen zu ermöglichen. Hierdurch sollen u.a. Anforderungen besser erfüllt werden und durch den Verzicht auf professionelle Softwareentwickler Kosten eingespart werden. Hierzu soll ein Endbenutzer-fokussiertes Entwicklungswerkzeug konzipiert und prototypisch implementiert werden. Die Analyse anderer domänenspezifischer Sprachen (siehe Kapitel 8.1.2) hat gezeigt, dass ein zweistufiges Vorgehen bei der Entwicklung domänenspezifischer Applikationen vorteilhaft ist. In einer ersten Stufe können die Applikationen implementierungsneutral spezifiziert werden; während ein Codegenerator in einer zweiten Stufe aus der formalen Spezifikation den Programmcode für eine lauffähige mobile ERP-Applikation generiert. Die domänenspezifische Sprache für mobile Applikationen dient bei diesem Vorgehen dazu, die mobilen ERP-Applikationen implementierungsneutral zu spezifizieren.

8.4.2 Analyse

Im Folgenden werden die in Kapitel 8.1.2 erläuterten Aspekte einer Domänenanalyse auf die eigene Arbeit übertragen.

8.4.2.1 Definition der Domäne

Eine Domäne kann nach Kelly und Tolvanen (2008) sowohl vertikal, als auch horizontal abgegrenzt werden. Eine *horizontale Abgrenzung* bezieht sich auf die technischen Charakteristiken, z.B. auf den fokussierten Gerätetyp oder die verwendete Kommunikationstechnologie (Kelly/Tolvanen 2008, 3). Eine *vertikale Abgrenzung* bezieht sich auf die Fachdomäne. Beispiele für eine vertikale Abgrenzung sind die Fokussierung auf bestimmte Branchen, wie beispielsweise auf das Versicherungswesen oder den Handel, oder die Fokussierung auf einen speziellen Funktionsbereich, z.B. auf die Logistik oder das Controlling (Kelly/Tolvanen 2008, 3).

Übertragen auf die eigene Arbeit kann eine horizontale Abgrenzung in Bezug auf den fokussierten Gerätetyp erfolgen. Die fokussierte Geräteklasse der Arbeit sind mobile Endgeräte. Zudem erfolgt eine weitere Einschränkung auf mobile Endgeräte vom Typ Smartphone. Dies ist notwendig, da sich Smartphones u.a. hinsichtlich ihrer Leistungsfähigkeit, ihrer eingebauten Sensorik und Interaktionstechnik von anderen mobilen Endgerätetypen unterscheiden (siehe Kapitel 2.2.4). Derzeit spielt bei der Entwicklung von mobilen Applikationen für Smartphones jedoch auch das zugehörige mobile Betriebssystem eine wichtige Rolle. Aufgrund der Unterschiede derzeitiger mobiler Betriebssysteme erfolgt in Anlehnung an Anforderung 1 (siehe Kapitel 5.2.1) eine Einschränkung auf die mobilen Betriebssysteme iOS und Android. Eine weitere technische Einschränkung basiert auf dem verwendeten ERP-System. Aufgrund der unterschiedlichen Programmierschnittstellen von ERP-Systemen erfolgt eine Einschränkung auf die Programmierschnittstellen eines SAP-ERP-Systems, den BAPIs (siehe Kapitel 2.1.5.2.1).

Horizontal erfolgt eine Fokussierung auf die Fachdomäne „Enterprise Resource Planning“. Dabei sollen die klassischen ERP-Funktionsbereiche, wie beispielsweise Logistik, Personal- oder Finanzwesen (siehe Kapitel 2.1), abgedeckt werden. Eine Einschränkung auf einen speziellen ERP-Funktionsbereich wird nicht vorgenommen. Um die Komplexität zu begrenzen wird jedoch eine Fokussierung auf einen bestimmten Applikationstyp und damit auf bestimmte Funktionalitäten vorgenommen. In Anlehnung an Anforderung 3 (siehe Kapitel 5.2.2) erfolgt eine Fokussierung auf Produktivitätsapplikationen mit den folgenden Funktionalitäten:

- Auflistung aller verfügbaren Business Objekt Typen
- Auflistung aller Business Objekte eines selektierten Business Objekt Typen
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten Business Objektes
- Erzeugung eines neuen Business Objektes
- Änderung eines existierenden Business Objektes
- Löschung eines existierenden Business Objektes

8.4.2.2 Terminologie der Domäne

Prinzipiell ist die in SAP verwendete Terminologie abhängig vom jeweils unterstützten Geschäftsprozess. Jedoch existieren auch Begriffe mit einer Geschäftsprozess-übergreifende Gültigkeit. Aus der Perspektive der Datenverwaltung wird in ERP-Systemen zwischen Stamm- und Transaktionsdaten unterschieden. *Stammdaten* bezeichnen betriebswirtschaftlich relevante Objekte, welche eine geringe Änderungshäufigkeit besitzen und i.d.R. bei der Durchführung unterschiedlicher Geschäftsprozesse verwendet werden. Beispiele für Stammdatensätze sind Kunden- oder Materialdatensätze. Das Gegenstück hierzu bilden sogenannte *Bewegungs-* oder *Transaktionsdaten*, welche stetig bei der Geschäftsprozessausführung erzeugt werden. Beispiele für Transaktionsdaten sind Kontobewegungen oder Kundenaufträge.

Im Zuge der objektorientierten Systemgestaltung des SAP-ERP-Systems hat SAP die Stamm- und Transaktionsdatensätze des ERP-Systems sowie die zugehörigen Methoden zur Manipulation der Datensätze zu Objekten zusammengefasst (siehe Kapitel 2.1.5.2.1). Der Typ eines Datensatzes wird als *Business Object Type* (BOT) bezeichnet; die zugehörigen Instanzen als *Business Objects* (BOs). Ein Beispiel für einen BOT ist „Kunde“. Ein Beispiel für ein BO die Daten des fiktiven Kunden „Max Mustermann“.

Um die Daten der BOs lesen und manipulieren zu können, werden sogenannte *Business Application Programming Interfaces* (BAPIs) genutzt. Ein BAPIs ist immer einem bestimmten BOT zugeordnet. In der Regel werden mindestens die vier CRUD-Methoden (Create, Read, Update, Delete) unterstützt. Abhängig vom jeweiligen BOT ist jedoch auch eine Vielzahl anderer BAPIs möglich.

8.4.2.3 Funktionale Konzepte der Domäne

Im Folgenden werden die identifizierten Konzepte der DSL vorgestellt. Hierbei handelt es sich um Mobile Business Object Typen und zugehörige Mobile Business Objects sowie Attribute und Methoden.

8.4.2.3.1 Mobile Business Object Type und Mobile Business Object

Das Ergebnis der vorherigen Phase “Analyse” hat gezeigt, dass BOTs und BOs aus einer objektorientierten Perspektive die zentralen Elemente der ERP-Domäne sind. Bisherige Ergebnisse dieser Arbeit haben jedoch gezeigt, dass in einer mobilen Applikation selten alle Datenfelder und Funktionalitäten eines BOTs benötigt werden; i.d.R. wird in einem mobilen Szenario nur eine ausgewählte Menge der verfügbaren Datenfelder und Funktionalitäten benötigt (siehe Charakteristik C6 in Kapitel 3.2). Aus diesem Grund wird in Analogie zum Entwicklungswerkzeug SAP Mobile Workspace (siehe Kapitel 6.1.2) der Begriff „Mobile Business Object“ verwendet. Um analog zu BOTs und BOs zwischen einer Spezifikation und einer Instanz zu unterscheiden, werden die Begriffe „Mobile Business Object Type“ (MBOT) und „Mobile Business Object“ (MBO) verwendet.

Das Konzept *MBOT* repräsentiert folglich eine Teilmenge eines BOTs aus einem SAP-ERP-System. Die Datenfelder eines MBOT werden durch Attribute repräsentiert. Die Funktionalitäten werden über zugehörige Methoden repräsentiert. Ähnlich zu einer Klasse in objektorientierten Programmiersprachen repräsentieren MBOT Schablonen. Das Konzept *MBO* repräsentiert die zugehörigen Objektinstanzen. Die Attribute von MBOs sind im Gegensatz zu den Attributen von MBOTs mit konkreten Werten belegt. MBOTs können zueinander in einer „ist Teil von“- Beziehung stehen. Beispielsweise steht ein MBOT „Mitarbeiter“ in einer „ist Teil von“-Beziehung zu einem MBOT „Abteilung“. Die Kardinalitäten der MBOT-Beziehungen sind bereits im ERP-System festgelegt und werden von dort übernommen.

8.4.2.3.2 Attribut

Das Konzept *Attribut* repräsentiert ein Datenfeld eines MBOT. Ein Attribut besitzt immer einen Datentyp sowie einen zugehörigen Wert. Als Datentypen sind sowohl intrinsischen

Datentypen wie beispielsweise Ganzzahlen, Gleitkommazahlen und Zeichenketten erlaubt, als auch andere MBOTs. Im ersten Fall wird jedes Attribut mit genau einem Attribut eines zugehörigen BOT des SAP-ERP-Systems verknüpft. Der zweite Fall realisiert eine „ist Teil von“-Beziehungen zwischen zwei MBOTs.

8.4.2.3.3 Methode

Das Konzept *Methode* repräsentiert die Funktionalität eines MBOT. Beispiele für häufige Funktionalitäten sind das Auflisten MBOs oder das Lesen und Manipulieren von MBO Attributwerten. Ein Methode ist mit immer mit genau einem BAPI des SAP-ERP-Systems verknüpft. Obgleich BOT in einem SAP-ERP-System eine Vielzahl von Methoden bereitstellen sind in nahezu jedem BOT Methoden für das Auflisten der existierenden BOs sowie das Anzeigen, Erzeugen, Bearbeiten und Löschen einzelner BOs vorhanden. Aus diesem Grund fokussiert sich die vorliegende domänenspezifische Sprache auf diese Methoden. Gemäß der Anforderung 3 dieser Arbeit (siehe Kapitel 5.2.2) sind diese die folgenden Methoden:

- Auflistung aller verfügbaren MBOTs
- Auflistung aller MBOs eines selektierten MBOT
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten MBO
- Erzeugung eines neuen MBO
- Änderung eines existierenden MBO
- Löschung eines existierenden MBO

8.4.2.3.4 Zusammenfassende Darstellung

Abbildung 8-5 illustriert die beschriebenen Konzepte und deren Beziehungen untereinander mit Hilfe der Entity-Relationship-Modell-Notation[37] zusammen.

[37] vgl. z.B.. (Ferstl/Sinz 2008, 132 ff.)

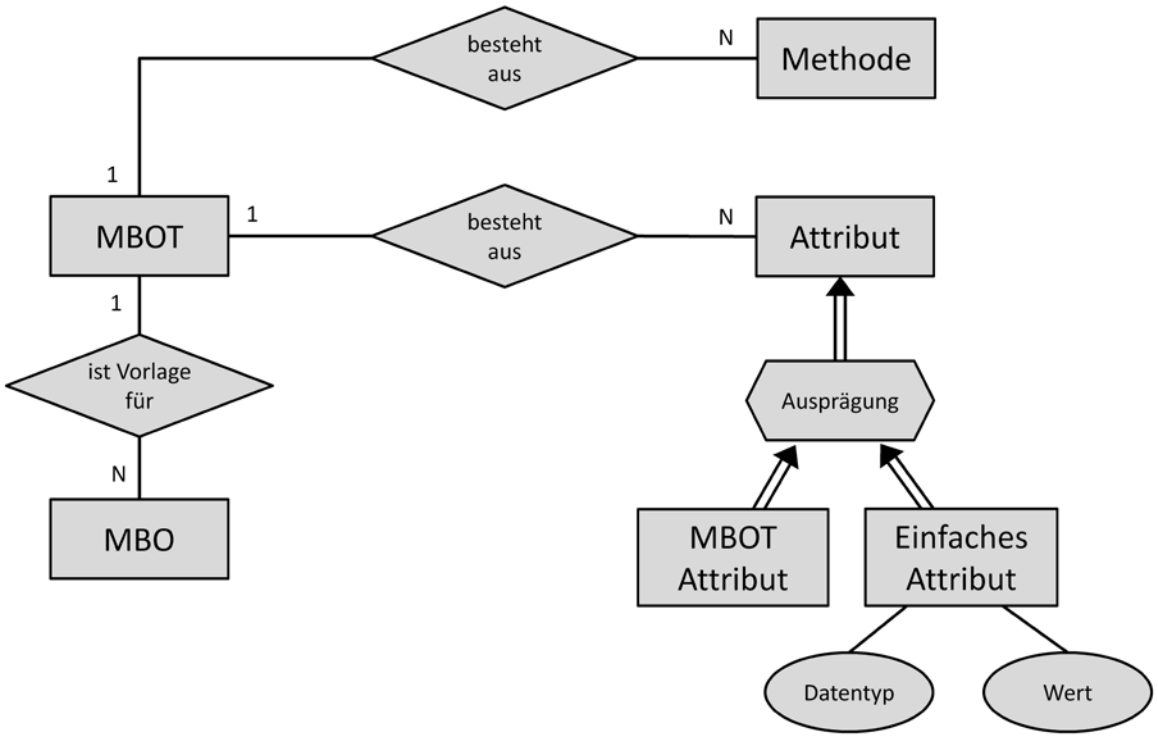

Abbildung 8-5: ERM-Diagramm der DSL-Konzepte
Quelle: Eigene Darstellung

8.4.2.4 Bedienkonzepte der Domäne

Bedienkonzepte beschreiben die Gestaltung der Benutzungsschnittstelle einer Applikation (Pühler 2011, 160). Übertragen auf eine mobile ERP-Applikation umfasst dies die Auswahl und Parametrisierung von Anzeige- und Bedienelementen sowie deren Positionierung auf einer Bildschirmmaske. Die Analyse der Benutzungsschnittstellen existierender mobiler ERP-Applikationen (siehe Kapitel 3.3.2.1; Charakteristik C11) hat gezeigt, dass die untersuchten Applikationen ähnliche Bedienkonzepte verwenden, um bestimmte Funktionalitäten zu repräsentieren. Für diese Arbeit sind diejenigen Bedienkonzepte relevant, welche zur Repräsentation der unter Anforderung 4 (siehe Kapitel 5.2.2) geforderten Funktionalitäten verwendet werden. Um die relevanten Bedienkonzepte zu beschreiben, wird das Format der Entwurfsmuster für Benutzungsschnittstelen (engl. UI-Patterns) verwendet (siehe Kapitel 2.3.3).

8.4.2.4.1 Sortierte Listen-Anzeige

Name: Sortierte Listen-Anzeige

Beschreibung:

Auflistung von MBOTs oder MBOs in einer bestimmten Reihenfolge. Als Sortierreihenfolge werden eine alphabetische Sortierung nach dem Elementnamen sowie eine nummerische Sortierung, beispielsweise nach Auftragspositionen eines Kundenauftrages, unterstützt.

Anwendungsfall:

Das Bedienkonzept „Sortierte Liste Anzeige“ sollte verwendet werden, wenn der Anwender ein bestimmtes Element aus einer Menge von Elementen auswählen soll. Bei Produktivitätsapplikationen sind die beiden primären Anwendungsfälle die Anzeige verfügbarer MBOTs und die Anzeige der MBOs eines ausgewählten MBOT.

Grund:

Der Nutzer soll in der Menge verfügbarer Elemente schnell navigieren und das für ihn interessante Element schnell selektieren können, um sich dessen Details anzuzeigen.

Umsetzung:

Zur Darstellung einer sortierten Liste von MBOTs oder MBOs eignet sich in iOS eine einspaltige Tabelle mit mehreren Zeilen. In iOS wird das zugehörige Anzeigeelement als *Table View* bezeichnet (Apple 2013a, 168 ff.). Die einzelnen Zellen der Tabelle sollten selektierbar sein und bei ihrer Selektion eine Liste der MBOs des jeweiligen MBOT anzeigen. Die Selektierbarkeit einer Zelle wird in iOS durch ein spezielles Symbol am rechten Rand der Zelle gekennzeichnet, den sogenannten „disclosure indicator“ (Apple 2013a, 170). Zusätzlich soll ein optionales Suchfeld über der Liste das schnelle Filtern der Listeneinträge ermöglichen. Nach dem Eintippen einer alphanummerischer Zeichenkette soll die Liste nur noch die Werte anzeigen, welche mit der eingegebenen alphanummerischen Zeichenkette beginnen. In iOS wird ein solches Bedienelement als *Search Bar* bezeichnet (Apple 2013a, 148 f.); in Android als *SearchView Widget* (Open-Handset-Alliance 2013). Der obere Teil der Bildschirmmaske sollte den Titel der aktuellen Anzeige sowie eine optionale Schaltfläche zur Navigation zur vorherigen Bildschirmmaske enthalten. In iOS wird dieses Anzeige- und Bedienelement als *Navigation Bar* bezeichnet (Apple 2013a, 139 ff.); in Android nennt sich das zugehörige Element *Action Bar*. Der untere Teil der Bildschirmmaske sollte ein optionales Copyright-Vermerk und auf der rechten Seite ein optionales Unternehmenslogo enthalten.

Sollte es sich bei der Listenanzeige um eine Auflistung von MBOs handeln, so soll unterhalb der Liste eine optionale Schaltfläche mit einem „+“ Symbol das Erzeugen neuer MBOs ermöglichen.

Beispiel:

Abbildung 8-6 illustriert das Bedienelement „Sortierte Listen-Anzeige“ am Beispiel einer Umsetzung für iOS (linke Bildschirmaufnahme), für Android (mittlere Bildschirmaufnahme) und HTML5 (rechte Bildschirmaufnahme). Die abgebildeten Mock-Ups wurden mit dem Werkzeug Fluid UI[38] umgesetzt.

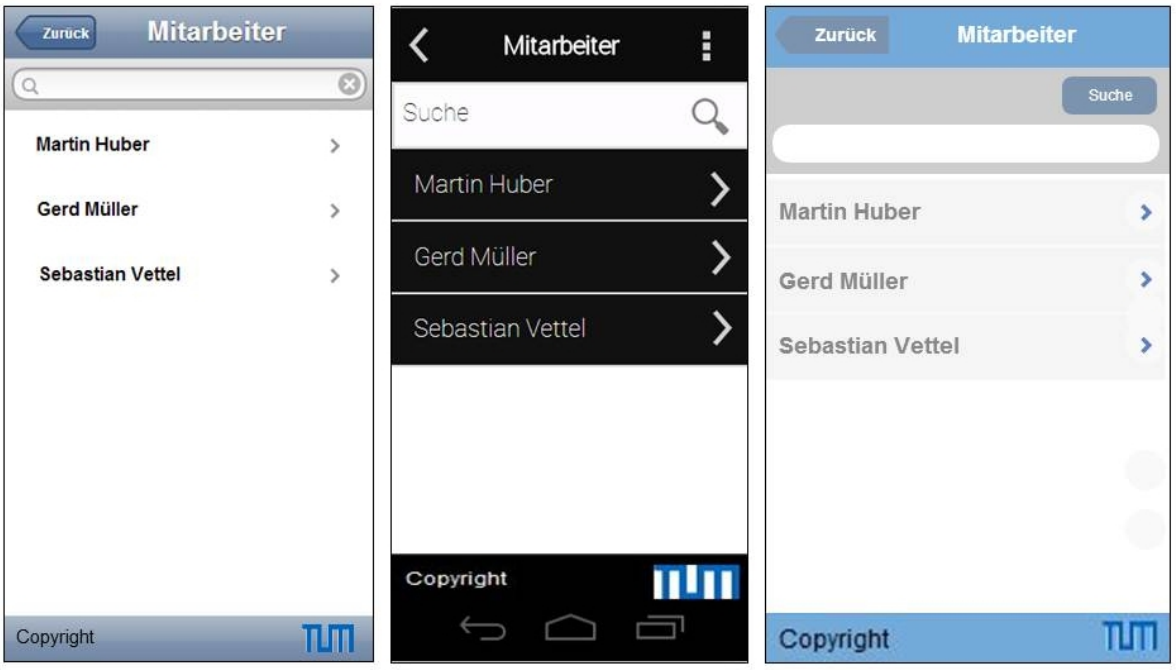

Abbildung 8-6: Bedienelement "Sortierte Liste Anzeige" am von Beispiel iOS, Android und HTML5
Quelle: Eigene Darstellung

8.4.2.4.2 Formularbasierte MBO-Anzeige

Name: Formularbasierte MBO-Anzeige

Beschreibung:

Dieses Bedienkonzept setzt die Anzeige der Attribute eines ausgewählten MBO um. Dabei sollen sowohl der Name des Attributes, als auch dessen Wert angezeigt werden. Zusätzlich soll über diese Anzeige die Möglichkeit angeboten werden, in einen Modus zu wechseln, welcher die Möglichkeit bietet ein ausgewähltes MBO zu löschen oder die Werte seiner Attribute zu ändern.

[38] https://www.fluidui.com, zugegriffen am 9.10.2013

Anwendungsfall:

Das Bedienkonzept „Formularbasierte MBO-Anzeige“ sollte verwendet werden, wenn ein bestimmtes MBO vom Anwender selektiert wurde und dessen Details angezeigt werden sollen.

Grund:

Der Anwender soll alle Attribute und dessen zugehörige Werte eines MBO übersichtlich auf einer Bildschirmmaske einsehen können. Zusätzlich soll er die Möglichkeit erhalten einzelne Attributwerte anzupassen oder das selektierte MBO zu löschen.

Umsetzung:

Die einzelnen Attribute sollen auf der Bildschirmmaske untereinander aufgelistet werden. Für jedes Attribut soll zunächst der Attributname angezeigt werden. Darunter soll sein jeweiliger Wert erscheinen. Um statischen Text anzuzeigen, existiert sowohl in iOS (Apple 2013a, 180 f.) als auch in Android (Open-Handset-Alliance 2013) das Anzeigeelement *Label*. Dies soll für die Anzeige der Attributnamen verwendet werden. Zur Anzeige der zugehörigen Attributwerte soll ein unter dem Attributnamen positioniertes Textfeld verwendet werden. Dieses wird sowohl in iOS (Apple 2013a, 189 f.), als auch in Android (Open-Handset-Alliance 2013) als *Text Field* bezeichnet. Zu beachten ist zusätzlich, dass das Textfeld nicht editierbar sein darf. Zusätzlich sollten zwei optionale Schaltflächen existieren, welche in die Modi „Bearbeiten“ bzw. „Löschen“ des aktuell angezeigten MBO wechseln. Die zugehörige Schaltfläche wird in Android als *Button* (Open-Handset-Alliance 2013) und in iOS als *Rounded Rectangle Button* (Apple 2013a, 185) bezeichnet. Der obere Teil der Bildschirmmaske soll analog zum Bedienelement „Sortierte Listen-Anzeige“ den Titel der aktuellen Anzeige sowie eine optionale Schaltfläche zur Navigation zur vorherigen Bildschirmmaske enthalten. Der untere Teil der Bildschirmmaske sollte ein optionales Copyright-Vermerk und auf der rechten Seite ein optionales Unternehmenslogo enthalten.

Beispiel:

Abbildung 8-7 illustriert das Bedienkonzept „Formularbasierte MBO-Anzeige“ am Beispiel einer Umsetzung für iOS (linke Bildschirmaufnahme), für Android[39] (mittlere Bildschirmaufnahme) und HTML5 (rechte Bildschirmaufnahme). Die abgebildeten Mock-Ups wurden mit dem Werkzeug Fluid UI umgesetzt.

[39] In Android werden die Funktionalitäten zum Bearbeiten und Löschen eines MBO über das Kontextmenü angezeigt, welches nach dem Drücken des Menüpunktes im unteren Teil der Bildschirmmaske dargestellt wird.

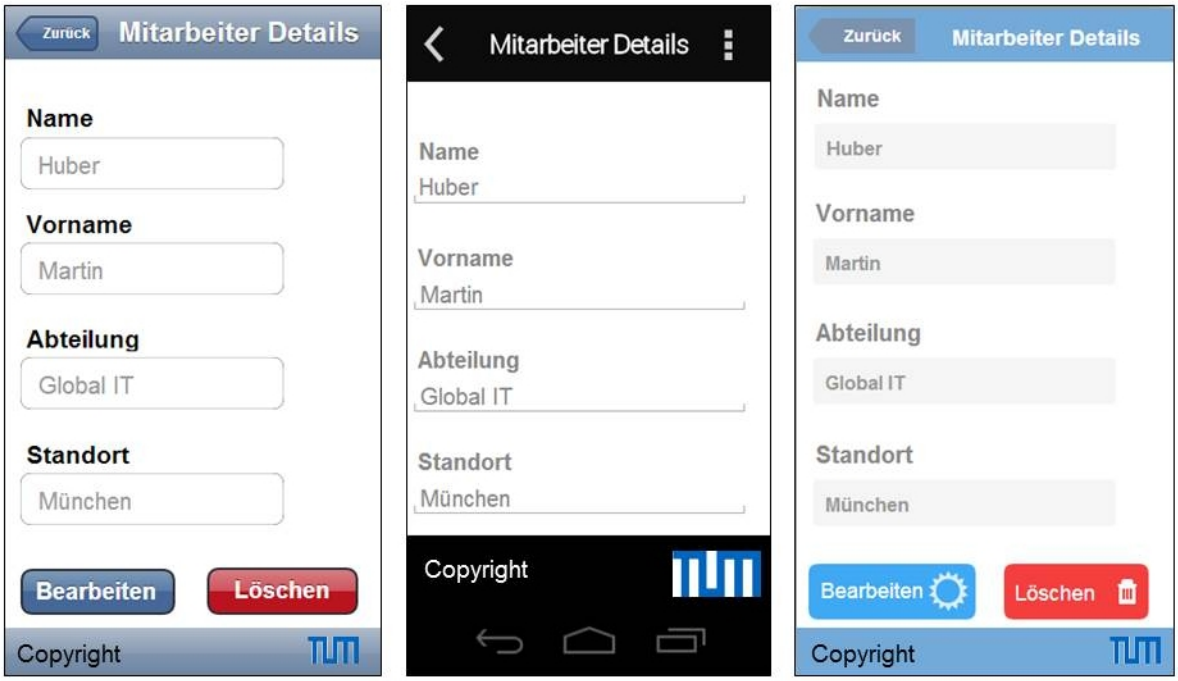

Abbildung 8-7: Bedienelement "Formularbasierte MBO Anzeige" am Beispiel iOS, Android und HTML5
Quelle: Eigene Darstellung

8.4.2.4.3 Formularbasierte MBO-Bearbeitung

Name: Formularbasierte MBO-Bearbeitung

Beschreibung:

Dieses Bedienkonzept setzt die Bearbeitung der Attribute eines ausgewählten MBO um. Dabei sollen sowohl der Name des Attributes, als auch dessen Wert angezeigt werden. Der Wert des Attributs sollte änderbar sein. Nach Abschluss der Bearbeitung sollen die Änderungen gespeichert werden können.

Anwendungsfall:

Das Bedienkonzept „Formularbasierte MBO-Bearbeitung“ sollte verwendet werden, wenn die Änderung von MBO-Attributen ermöglicht werden soll. In diesem Fall muss der Anwender zunächst ein MBO selektieren und anschließend in den Bearbeitungsmodus wechseln.

Grund:

Der Anwender soll alle Attribute und dessen zugehörige Werte eines MBO übersichtlich auf einer Bildschirmmaske einsehen können. Zusätzlich soll er die Möglichkeit erhalten einzelne Attributwerte anzupassen oder das selektierte MBO zu löschen.

Umsetzung:

Analog zum Bedienkonzept „Formularbasierte MBO-Anzeige“ sollen die Attribute auf der Bildschirmmaske untereinander aufgelistet werden. Für jedes Attribut soll zunächst der Attributname gezeigt werden. Darunter soll sein jeweiliger Wert erscheinen. Hierzu soll analog zum Bedienkonzept „Formularbasierte MBO-Anzeige“ ein Textfeld verwendet werden. Allerdings sollte der Wert des Textfeldes angepasst werden können. Falls für ein Attribut ein Wert aus einer Menge vorgegebener Werte ausgewählt werden soll, so ist ein Textfeld nicht geeignet. In diesem Fall sollte ein Auswahlfeld verwendet werden. In iOS (Apple 2013a, 182 f.) und Android (Open-Handset-Alliance 2013) steht hierfür das Bedienelement *Picker* (Apple 2013a, 182 f.) zur Verfügung. Der obere Teil der Bildschirmmaske soll analog zum Bedienelement „Sortierte Listen-Anzeige“ den Titel der aktuellen Anzeige sowie eine optionale Schaltfläche zur Navigation zur vorherigen Bildschirmmaske enthalten. Zusätzlich soll auf der rechten Seite eine Schaltfläche zum Bestätigen der durchgeführten Änderungen existieren. In Android wird dieses durch einen entsprechenden Menüpunkt im Kontextmenü realisiert. Das Betätigen der Schaltfläche bzw. des Menüpunktes soll das Speichern der Änderungen bewirken. Der untere Teil der Bildschirmmaske sollte ein optionales Copyright-Vermerk und auf der rechten Seite ein optionales Unternehmenslogo enthalten.

Beispiel:

Abbildung 8-8 illustriert das Bedienelement „Formularbasierte MBO-Bearbeitung“ am Beispiel einer Umsetzung für iOS (linke Bildschirmaufnahme), für Android (mittlere Bildschirmaufnahme) und HTML5 (rechte Bildschirmaufnahme). Die abgebildeten Mock-Ups wurden mit dem Werkzeug Fluid UI umgesetzt.

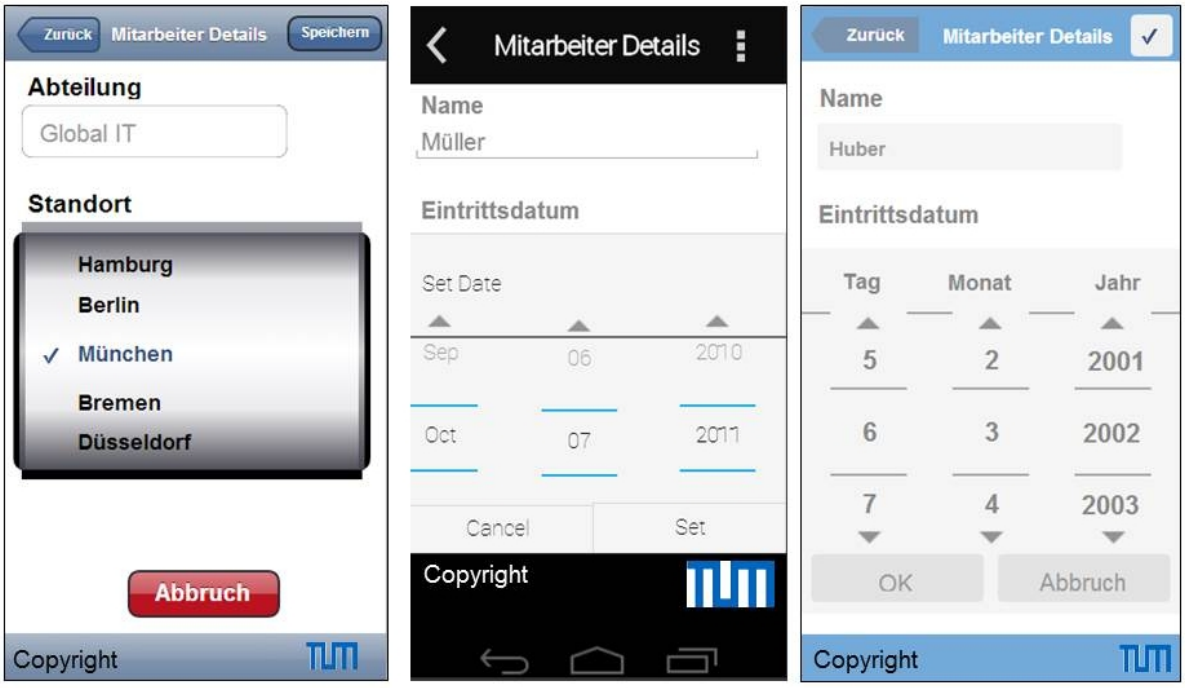

Abbildung 8-8: Bedienelement "Formularbasierte MBO-Bearbeitung" am Beispiel iOS, Android und HTML5

Quelle: Eigene Darstellung

8.4.2.5 Domänenregeln

In Anlehnung an das in Kapitel 8.3 beschriebene Verständnis von Domänenregeln, legen diese die Ausprägungsformen der einzelnen Domänenkonzepte sowie deren Abhängigkeiten fest. Basierend auf den Ergebnissen der Analysephase wird in der domänenspezifischen Sprache für mobile ERP-Applikationen zwischen funktionalen Konzepten und Bedienkonzepten unterschieden. Als funktionale Konzepte konnten MBOTs, MBOs, Attribute und Methoden identifiziert werden. Als Bedienkonzepte wurden die „Sortierte Listen-Anzeige“, die „Formularbasierte MBO-Anzeige“ und die „Formularbasierte MBO-Bearbeitung“ vorgestellt. Im Folgenden werden die Ausprägungsformen dieser Konzepte sowie deren Verknüpfungsregeln beschrieben.

Gemäß der zusammenfassenden Darstellung in Abbildung 8-5 sind Attribute und Methoden Bestandteile eines MBOT. Welche Attribute und Methoden ein MBOT beinhaltet ist von zwei Faktoren abhängig. Der erste Faktor ist das zugehörige BOT des SAP-ERP-Systems. Ein MBOT ist immer mit genau einem BOT des SAP-ERP-Systems verbunden. Daher repräsentiert die Menge an Attributen und Methoden eines BOT die maximale Menge an Attributen und Methoden eines MBOT. Der zweite Faktor ist die zweckmäßige Reduktion der Attribut- und Methodenmenge für die mobile Nutzung. Dieser Faktor leitet sich aus der Charakteristik C6 ab (siehe Kapitel 3.2), wonach in einem mobilen Anwendungsszenario nur ausgewählte Daten und Funktionalitäten benötigt werden.

Die Verknüpfungsregeln der funktionalen Konzepte werden maßgeblich durch das View-Inspect-Act-Interaktionsmuster geprägt (siehe Charakteristik C10 in Kapitel 3.3.2.1). Nach diesem Interaktionsmuster möchte sich ein Anwender zunächst einen Überblick verschaffen (View). Danach schaut macht er sich mit den entsprechenden Details vertraut (Inspect) und führt schließlich seine gewünschte Aktion durch (Act).

Für die Domänenregeln bedeutet dies, dass die MBOT-Methoden nur in einer bestimmten Reihenfolge ausgeführt werden dürfen. Falls eine mobile ERP-Applikation mehrere MBOTs umfasst, dann sollte die erste Methode immer die Auflistung aller verfügbaren MBOTs sein. Dabei handelt es sich um eine statische Liste, für deren Erstellung keine Abfrage an das SAP-ERP-System erfolgen muss. Nach der Selektion eines bestimmten MBOT durch den Anwender, sollte eine Auflistung aller MBOs des selektierten Typs erfolgen. Wurde beispielsweise ein MBOT „Kunde“ selektiert, so sollte in der Folgemethode eine Auflistung der gespeicherten Kunden erfolgen. In diesem Fall muss eine Datenabfrage aus dem SAP-ERP-System durchgeführt werden. Nun sollte der Anwender die Möglichkeit besitzen ein bestimmtes MBO zu selektieren oder ein neues MBO vom Typ des selektierten MBOT zu erzeugen. Beides sind optionale Methoden, welche nicht in jeder mobilen ERP-Applikation benötigt werden.

Wenn der Anwender nun ein bestimmtes MBO auswählt, so sollte in der Folgemethode eine Abfrage der Details des selektierten MBO erfolgen, d.h. dessen Attribute inkl. zugehöriger Werte abgefragt und angezeigt werden. Auch in diesem Fall ist eine Datenabfrage aus dem SAP-ERP-System notwendig. Falls der Anwender ein neues MBO erzeugen möchte, so sollte ein Formular mit den Attributen des MBOT inkl. zugehöriger Felder zur Eingabe entsprechender Attributwerte angezeigt werden. Nach der Speicherung der Eingaben durch den Anwender sollte die entsprechende Programmierschnittstelle zur Speicherung des neuen MBO im SAP-ERP-System aufgerufen werden.

Bei der Anzeige eines bestimmten MBO sollte die optionale Methode zum Löschen des selektierten MBO zur Verfügung stehen. Beim Aufruf dieser Methode sollte ebenfalls die entsprechende Programmierschnittstelle zum Löschen des selektierten MBO im SAP-ERP-System aufgerufen werden. Zusätzlich sollte die optionale Methode zur Änderung des selektierten MBO aufrufbar sein. In diesem Fall sollte ein Formular mit den Attributen und aktuellen Attributwerden angezeigt werden, welche auch geändert werden können. Nach der Bestätigung der getätigten Änderungen sollte wiederrum die entsprechende Programmierschnittstelle zur Speicherung der geänderten Attributwerte im SAP-ERP-System aufgerufen werden. Abbildung 8-9 veranschaulicht die beschriebenen Verknüpfungsregeln von MBOT-Methoden durch ein UML 2 Aktivitätsdiagram.

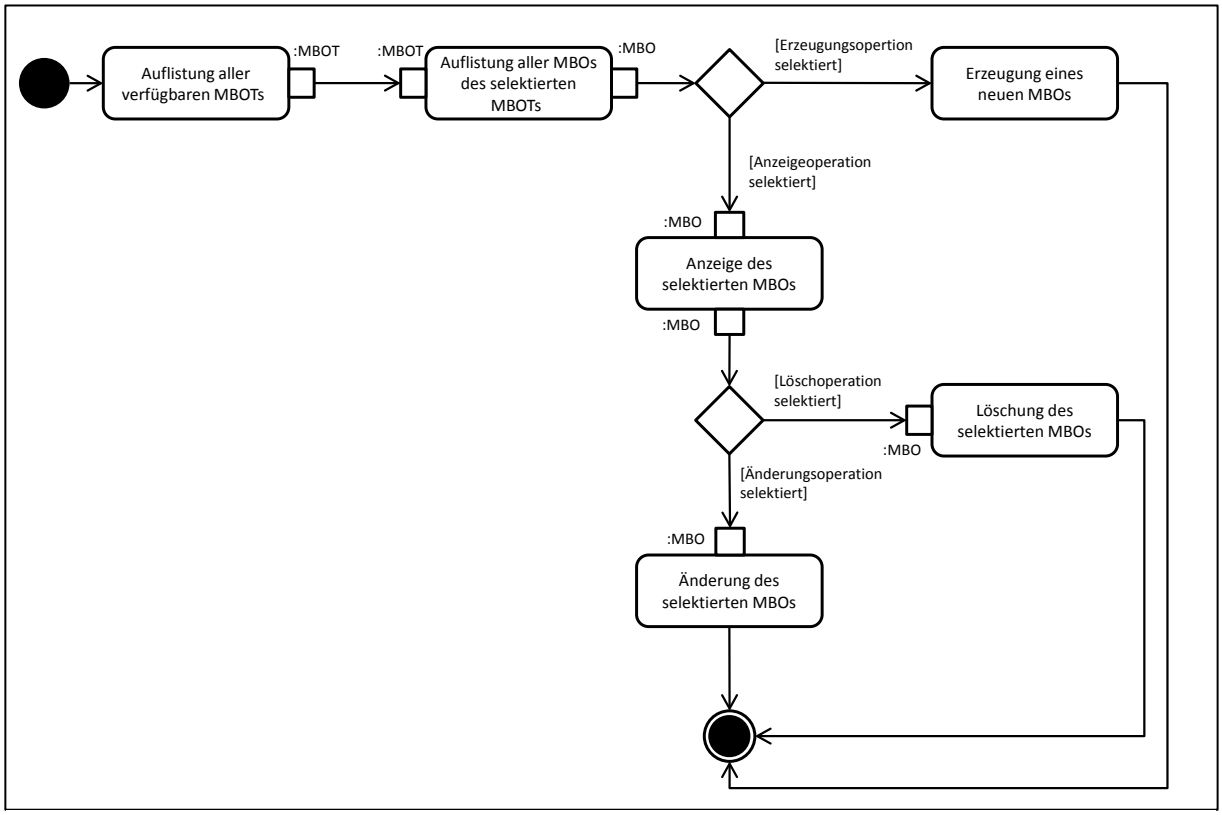

Abbildung 8-9: Verknüpfungsregeln von Methoden

Quelle: Eigene Darstellung

Eine weitere Menge an Verknüpfungsregeln bezieht sich auf die Verknüpfung zwischen den funktionalen Konzepten mit zugehörigen Bedienkonzepten. Die DSL wird so konzipiert, dass die Benutzungsschnittstelle eines funktionalen Konzeptes durch jeweils ein bestimmtes Bedienkonzept umgesetzt wird.

Die beiden funktionalen Konzepte „Auflistung aller verfügbaren MBOTs“ und „Auflistung aller verfügbaren MBOs des selektierten MBOT“ benötigen eine listenartige Repräsentation. Demzufolge werden beide mit dem Bedienkonzept „Sortierte Listen-Anzeige“ verknüpft. Das funktionale Konzept „Anzeige des selektierten MBO“ benötigt eine formularbasierte Benutzungsschnittstelle zur Anzeige der MBO Attribute und deren zugehörige Werte. Daher wird eine Verknüpfung mit dem Bedienkonzept „Formularbasierte MBO-Anzeige“ vorgenommen. Falls auch das funktionale Konzept „Löschung des selektierten MBO“ genutzt wird, so wird lediglich eine zusätzliche Schaltfläche zum Ansteuern des Löschvorganges benötigt. Diese kann bei Bedarf im Bedienkonzept „Formularbasierte MBO-Anzeige“ aktiviert werden. Daher wird das funktionale Konzept „Löschung des selektierten MBO“ mit dem Bedienkonzept „Formularbasierte MBO-Anzeige“ verknüpft. Für das funktionale Konzept „Erzeugung eines neuen MBO“ wird ebenfalls eine formularbasierte Benutzungsschnittstelle zur Eingabe der gewünschten Attributwerte für ein MBO benötigt. In diesem Fall müssen die Bedienelemente für die Eingabe der Attributwerte editierbar sein. Gleiches gilt für die Benutzungsschnittstelle des funktionalen Konzeptes „Änderung des selektierten MBO“. In diesem Fall müssen die Bedienelemente der Attributwerte jedoch mit den aktuell gespeicherten Attributwerten des selektierten MBO vorbelegt werden. Für diese Fälle eignet sich das Bedienkonzept

„Formularbasierte MBO-Bearbeitung“. Daher wird eine Verknüpfung der funktionalen Konzepte „Erzeugung eines neuen MBO“ und „Änderung des selektierten MBO“ mit dem Bedienkonzept „Formularbasierte MBO-Bearbeitung“ durchgeführt. Die beschriebenen Verknüpfungsregeln zwischen den funktionalen- und den zugehörigen Bedienkonzepten sind in Abbildung 8-10 illustriert.

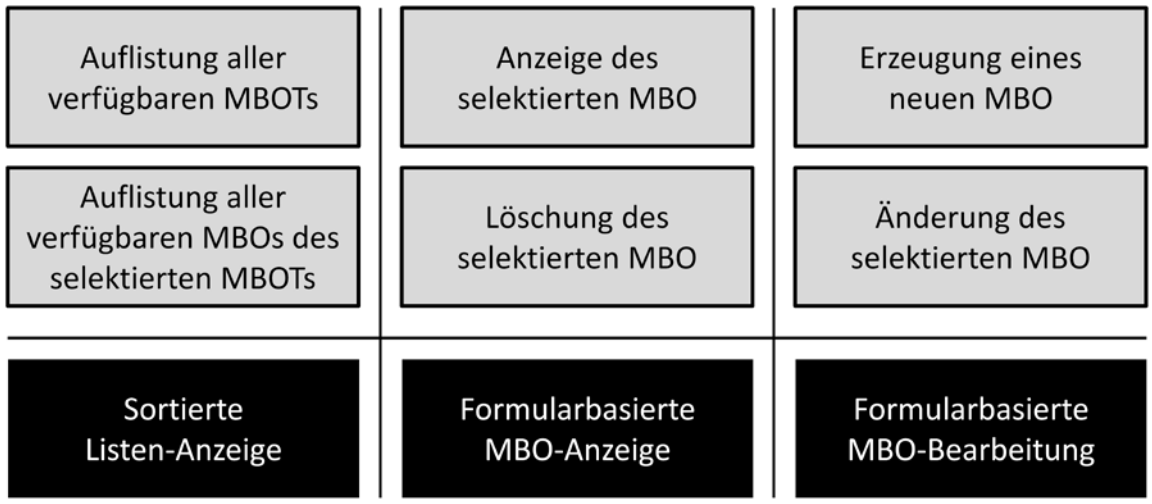

Abbildung 8-10: Verknüpfungsregeln zwischen funktionalen- und Bedienkonzepten
Quelle: Eigene Darstellung

8.4.3 Design

In diesem Kapitel wird die Designphase der zu entwickelnden DSL für mobile ERP-Applikationen behandelt. Hierbei wird zunächst diskutiert, ob für diesen Zweck eine bestehende Sprache erweitert werden kann, oder ob die Gestaltung einer neuen Sprache zweckdienlicher ist. Schließlich wird die konzipierte DSL-Notation für mobile ERP-Applikationen vorgestellt.

8.4.3.1 Spracherweiterung vs. Gestaltung einer neuen Sprache

Gemäß den Ausführungen von Mernik et al. (2005, 326 f.) liegt die zentrale Entscheidung in der Designphase einer DSL in der Wahl zwischen der Option zur Erweiterung einer bestehenden GPL um die notwendigen DSL-Konzepte und –Regeln und der Option zur Erstellung einer neuen Sprache (siehe Kapitel 8.1.2). Eine potenzielle Basissprache für diese Arbeit sollte die Eigenschaft besitzen, dass mit ihrer Hilfe bereits mobile ERP-Applikationen realisiert werden könnten. Hierzu müssten sowohl die gewünschten Benutzungsschnittstellen umsetzbar sein, als auch der Aufruf der benötigten Programmierschnittstellen des SAP-ERP-Systems möglich sein. Eine solche Sprache könnte um die höherwertigen Konstrukte der DSL angereichert werden, um sie hierdurch Endbenutzer-tauglich zu gestalten.

Aktuelle Entwicklungstechnologien für native Applikationen umfassen derzeit mehrere Elemente. Beispielsweise wird für die Entwicklung von iOS-Applikationen die Programmiersprache Objective-C, die Entwicklungsumgebung Xcode und die Programmierbibliothek Cocoa Touch verwendet. Zudem haben iOS-Entwicklungsprojekte einen spezifischen Aufbau und benötigen eine Reihe von Konfigurationsdateien (Homann et al. 2013b, 395 ff.; Rodeweig/Wagner 2013, 32 ff.). Ähnlich ist die Situation bei Android. Hier wird die Programmiersprache Java und die Entwicklungsumgebung Eclipse eingesetzt. Zusätzlich wird das sogenannte Android Development Toolkit verwendet, welches neben einer Android-spezifischen Entwicklungsperspektive für Eclipse auch zugehörige Programmierbibliotheken bereitstellt. Auch bei Android müssen Entwicklungsprojekte einen bestimmten Aufbau sowie eine Reihe von spezifischen Konfigurationsdateien besitzen (Künneth 2012, 29 ff.; Homann et al. 2013b, 359 ff.). Eine DSL auf Basis einer solchen nativen Entwicklungstechnologie wäre durch den Einsatz der geschilderten Technologievielfalt relativ aufwändig umzusetzen und zudem auf ein bestimmtes mobiles Betriebssystem beschränkt.

Eine Alternative bietet die Erweiterung eines HTML5-Frameworks für die Entwicklung mobiler Applikationen. Derartige Frameworks erleichtern die Entwicklung von Webapplikationen für mobile Endgeräte, indem sie u.a. die typischen Anzeige- und Bedienelemente mobiler Applikationen als wiederverwendbare Elemente bereitstellen (Spiering/Haiges 2010, 41). Diese Frameworks nutzen als technologische Basis verbreitete Webtechnologien wie bspw. HTML, CSS und JavaScript (siehe Kapitel 3.2).

Bekannte Vertreter von HTML5-Frameworks für mobile Applikationen sind iWebKit, jQT, Sencha Touch oder jQuery Mobile (Spiering/Haiges 2010, 41 ff.; Bai 2011, 9 ff.). *iWebKit*[40] verzichtet im Gegensatz zu anderen HTML5-Framworks auf JavaScript und Ajax[41] und ist dadurch verhältnismäßig einfach zu nutzen (Spiering/Haiges 2010, 41 ff.). Die primäre Zielgruppe von iWebKit sind unerfahrene Webentwickler (Bai 2011, 15). Während sich iWebKit auf das Aussehen der Anzeige- und Bedienelemente beschränkt, lassen sich mit *jQT*[42] (vormals: jQTouch) auch Animationseffekte von iOS und Android nutzen (Spiering/Haiges 2010, 53 ff.). jQT setzt auf der verbreiteten JavaScript-Bibliothek *jQuery*[43] auf, welche Funktionalitäten zur Manipulation von Webseiten bereitstellt. Im Gegensatz zu jQuery ist jQT auf mobile Webseiten fokussiert und hat Anfänger in der Webprogrammierung als primäre Zielgruppe (Bai 2011, 9 ff.). Vom gleichen Entwicklerteam wie jQT stammt Sencha Touch. *Sencha Touch*[44] hat eine ähnliche Zielsetzung und einen vergleichbaren Funktionsumgang wie jQT, ist jedoch für erfahrenere Webentwickler konzipiert (Bai 2011, 10). Im Gegensatz zu jQT setzt Sencha Touch nicht auf jQuery auf und ist daher weniger ressourcenintensiv (Bai 2011, 11). *jQuery Mobile*[45] basiert anlog zu jQT auf jQuery, wird jedoch direkt vom jQuery Entwicklerteam entwickelt. Dabei handelt es sich analog zu den anderen HTML5-Frameworks um eine Bibliothek für die Entwicklung von Applikationen für mobile

[40] http://snippetspace.com/portfolio/iwebkit, zugegriffen am 03.02.2014
[41] Abkürzung für „Asynchronous JavaScript and XML“
[42] http://jqtjs.com, zugegriffen am 03.02.2014
[43] http://jquery.com, zugegriffen am 03.02.2014
[44] http://www.sencha.com/products/touch, zugegriffen am 03.02.2014
[45] http://jquerymobile.com, zugegriffen am 03.02.2014

Endgeräte mit Gestensteuerung. Im Gegensatz zu den anderen Frameworks hat jQuery Mobile eine Vielzahl von namenhaften Sponsoren (Bai 2011, 16). Obgleich jQuery Mobile auf jQuery basiert, wurde die Paketgröße für die mobile Nutzung stark optimiert (Bai 2011, 19). Während andere HTML5-Frameworks für mobile Applikationen primär auf iOS ausgerichtet sind, unterstützt jQuery Mobile eine Vielzahl von mobilen Endgeräten, mobilen Betriebssystemen und zugehörigen Webbrowsern[46] (Bai 2011, 17 ff.). Ähnlich wie iWebKit ist jQuery Mobile einfach zu nutzen. Insbesondere sind keine JavaScript Programmierkenntnisse erforderlich (Bai 2011, 19). Nichtsdestotrotz ist es jedoch möglich JavaScript Programmcode einzubauen, um beispielsweise eine Programmierschnittstelle eines SAP-ERP-Systems anzusprechen. Resultierend aus den genannten Vorteilen wird jQuery Mobile im Folgenden als repräsentatives HTML5-Framework für die Entwicklung mobiler ERP-Applikationen betrachtet.

8.4.3.1.1 Erweiterung von jQuery Mobile

In jQuery Mobile werden UI-Elemente als sogenannte *Widgets* bezeichnet. Da sich unterschiedliche UI-Elemente in ihrem Aussehen und ihrem Verhalten unterscheiden, werden diese Spezifika über wiederverwendbare Widget-Implementierungen gekapselt (Friberg 2013, 104).

Für eine gewünschte Spracherweiterung stellt jQuery Mobile die Möglichkeit zur Verfügung, eigene Widgets zu entwickeln und in zukünftigen Applikationen wiederzuverwenden. Über die sogenannte *Widget Factory* existiert ein standardisierter Weg dies umzusetzen. Hierzu müssen bestimmte Methoden, bspw. zum Erzeugen oder Löschen des neuen Widgets, implementiert werden. Anschließend kann dieses wie andere UI-Elemente des jQuery Mobile Frameworks verwendet werden (Friberg 2013, 203 ff.). Damit könnten die in der Analysephase (siehe Kapitel 8.4.2.4) vorgestellten Bedienkonzepte umgesetzt werden.

Eine DSL-Konzeption auf Basis von jQuery Mobile schränkt die zu entwickelnden mobilen ERP-Applikationen jedoch auf Webapplikationen ein. Eine zukünftige Erweiterung auf die Entwicklung nativer Applikationen wäre hiermit nicht möglich. Zum anderen ist eine Spracherweiterung auf Basis von jQuery Mobile vor allem dann sinnvoll, wenn die Endbenutzer bereits mit HTML vertraut sind und damit umgehen können. Dies ist notwendig, da auch im Falle eigener Widgets die Grundstruktur einer jQuery Mobile Applikation definiert werden muss. Die Kenntnis von HTML oder die Motivation sich fehlende HTML-Kenntnisse anzueignen kann jedoch bei den fokussierten Endbenutzern dieser Arbeit nicht generell vorausgesetzt werden.

8.4.3.1.2 Neue Sprache auf Basis von XML

XML steht für „Extensible Markup Language“; übersetzt ins Deutsche „erweiterbare Auszeichnungssprache“. Dabei handelt es sich um ein vom World Wide Web Consortium (W3C) standardisiertes Format, um Daten in einer hierarchisch strukturierten, textbasierten Form

[46] Eine Übersicht über die unterstützten Systemkonfigurationen ist unter folgendem Link zu finden: http://jquerymobile.com/gbs, zugegriffen am 03.02.2014

darzustellen. Die einzelnen Textbestandteile werden dabei durch sogenannte *Tags* markiert. Hierdurch wird diesen Textbestandteilen eine semantische Bedeutung zugewiesen (Bach 2000, 25 ff.; Sebestyen 2010, 17 ff.).

Bei XML handelt es sich nicht nur um eine „einfache“ Auszeichnungssprache, sondern um eine „Metasprache“, mit dessen Hilfe eigene Auszeichnungssprachen definiert werden können (Sebestyen 2010, 73). Beispiele für auf XML basierte Auszeichnungssprachen sind die beiden W3C-Standards: SVG[47] (Scalable Vector Graphics) zur Spezifikation von zweidimensionalen Vektorgraphiken und MathML[48] (Mathematical Markup Language) zur Definition von mathematischen Formeln und komplexen Ausdrücken

Aufgrund seiner universellen Einsetzbarkeit hat XML ein breites Anwendungsspektrum. Neben den Anwendungsgebieten der genannten XML-Derivate SVG und MathML wird XML häufig für das plattformübergreifende Publizieren von Informationen, die Definition von plattformunabhängigen Protokollen für den Datenaustausch oder die automatisierte Verarbeitung von übertragener Daten durch Computersysteme eingesetzt (Bach 2000, 20 f.). Zudem wird XML häufig als Dateiformat zur Speicherung von Konfigurationsdaten in Softwarewerkzeugen oder Anwendungsservern verwendet. Auch im Bereich der mobilen Applikationsentwicklung wird XML eingesetzt. Beispielsweise werden die Berechtigungen einer Android Applikation in einem XML-basierten Format[49] konfiguriert. In der iOS-Entwicklung wird u.a. die Gestaltung der Benutzungsschnittstelle in einem XML-basierten Format[50] gespeichert. Die Untersuchung der domänenspezifischen Sprachen in Kapitel 8.1.2 hat auch gezeigt, dass sich XML auch eignet, um eine domänenspezifische Sprache zu definieren.

XML besitzt gegenüber einer anderen Sprachgrundlage, wie beispielweise dem im vorherigen Kapitel diskutierten jQuery Mobile, den Vorteil, dass es einen großen Gestaltungsspielraum besitzt. Zudem hat XML prinzipiell keine Einschränkungen bzgl. des zu erzielenden Ergebnisformates. Durch die Bereitstellung eines entsprechenden Codegenerators kann eine durch XML spezifizierte mobile ERP-Applikation prinzipiell in jede Implementierungsvariante überführt werden. Diesem Vorteil steht der höhere Implementierungsaufwand für die Bereitstellung des Codegenerators gegenüber, da nicht in gleichem Umfang auf bestehende Programmierbibliotheken und existierende Entwicklungswerkzeuge wie bei einer GPL als Sprachgrundlage zurückgegriffen werden kann. Um die Anforderung bzgl. der Erweiterbarkeit des gewünschten Entwicklungswerkzeuges neben iOS und Android auf andere mobile Betriebssysteme zu gewährleisten (siehe Anforderung 12, Kapitel 5.2.1) wird für diese Arbeit eine XML-basierte Notation für die Spezifikation der domänenspezifischen Sprache für mobile ERP-Applikationen gewählt.

[47] http://www.w3.org/TR/2003/REC-SVG11-20030114, zugegriffen am 21.02.2014
[48] http://www.w3.org/TR/MathML3, zugegriffen am 21.02.2014
[49] Die zugehörige Datei nennt sich „AndroidManifest.xml“
[50] Die zugehörigen Dateien nennen sich XIB-Dateien

8.4.3.2 Gestaltung einer XML-basierten Notation für mobile ERP-Applikationen

Ziel der XML-basierten Notation ist es, die funktionalen Konzepte sowie die zugehörigen Bedienkonzepte bei der Spezifikation einer mobilen ERP-Applikation zu berücksichtigen. Um die MBOTs und zugehörigen Attribute zu spezifizieren, werden entsprechende XML-Elemente eingeführt (siehe z.B. (Sebestyen 2010, 37 ff.)). Zur Spezifikation des funktionalen Konzeptes „Mobile Business Object Type“ wird das XML-Element **<mbot>** eingeführt; für die Spezifikation des funktionalen Konzeptes „Attribut“ wird das XML-Element **<attribute>** eingeführt. Das funktionale Konzept „Mobile Business Object“ wird bei der Spezifikation der Sprache nicht direkt berücksichtigt. Grund hierfür ist, dass es sich bei MBOs um Instanzen von MBOT handelt, welche erst bei der Wertabfrage aus dem SAP-ERP-System instanziiert werden. Aus diesem Grund sind sie beim Zeitpunkt der Spezifikation der mobile ERP-Applikation nicht relevant. Ein Ausschnitt aus einer beispielhaften Spezifikation einer mobilen ERP-Applikation ist in Listing 8-1 veranschaulicht.

```
<mbot name="employee" description="Employee">
  <attribute name="familyname" description="Family Name"
             type="string"
             identifier="true" />
  <attribute name="firstname" description="First Name"
             type="string" />
  <attribute name="department" description="Department"
             type="select">
              <item name="DE" description="Sales" />
              <item name="UK" description="Global IT" />
              <item name="US" description="Finance" />
  </attribute>
  <attribute name="email" description="E-Mail"
              type="email" />
</mbot>
```

Listing 8-1: Beispielhafte MBOT Spezifikation

Quelle: Eigene Darstellung

In diesem Beispiel wird ein MBOT „Mitarbeiter“ mit den Attributen Vor- und Nachname, Abteilung und E-Mail Adresse spezifiziert. In diesem Beispiel ist auch zu sehen, dass das Element **<attribute>** als Bestandteil des Elementes **<mbot>** definiert wurde, um die 1:N-Relation zwischen diesen beiden funktionalen Konzepten abzubilden. Zudem besitzen beide XML-Elemente entsprechende XML-Attribute, um die Eigenschaften der XML-Elemente zu beschreiben (siehe z.B. (Sebestyen 2010, 47 ff.)).

Das Element **<mbot>** besitzt ein XML-Attribut „name“, welches den eindeutigen Namen des MBOT enthält. Zudem gibt es ein XML-Attribut „description“, welches beispielsweise für eine Hilfefunktion der Benutzungsschnittstelle zur Erläuterung des Attributes verwendet werden kann.

Das Element **<attribute>** besitzt ebenfalls die beiden XML-Attribut „name“ und „description“. Diese dienen ebenso wie beim Element **<mbot>** dazu, das MBOT-Attribut eindeutig zu identifizieren inkl. einer zusätzlichen Beschreibung. Zusätzlich wird das Attribut „type“ definiert, welches den Datentyp des XML-Attributes festlegt. Möglich sind u.a. die Datentypen „int“ für ganzzahlige Werte, „string“ für Zeichenketten oder „email“ für die

Angabe von E-Mail Adressen. Zudem kann über den Typ „select“ eine Wertemenge vorgegeben werden. Zusätzlich wird über das XML-Attribut „identifier“ dasjenige Attribut festgelegt, nach welchem in einer listenförmigen Darstellung sortiert werden soll und welches gleichzeitig zur primären Identifizierung des MBOT gesondert hervorgehoben werden soll.

Um die gewünschte 1:N-Realtion zwischen MBOTs zu ermöglichen wird das Attribut „type“ eingeführt. Falls dieses den Wert „relation“ besitzt, so liegt eine 1:N-Relation zu einem anderen MBOT vor, d.h. beim vorliegenden Attribut handelt es sich wiederum um einen MBOT. Um die Relation zum zugehörigen MBOT zu spezifizieren wird das XML-Element **<relation>** spezifiziert, welches über sein Attribut „entity“ den Wert des Attributes „name“ des verknüpften MBOT enthält. Ein Auszug aus einer beispielhaften Spezifikation einer 1:N-Relation zwischen den MBOTs „Abteilung“ und „Mitarbeiter“ ist in Listing 8-2 veranschaulicht.

```
<mbot name="department" description="Department" >
  <attribute name="department-id"
             description="Department ID"/>
  <attribute name="department-name"
             description="Department Name"
             identifier="true" />
  <attribute name="postalcode"
             description="Postalcode" />
  <attribute name="street"
             description="Street" />
  <attribute name="street-number"
             description="Street Number" />
</mbot>

<mbot name="employee" description="Employee">
  <attribute name="employee-id"
             description="Employee ID" />
  <attribute name="department-id"
             description="Department"
             type="relation">
             <relation entity="department" />
  </attribute>
  <attribute name="email"
             description="E-Mail"
             type="email" />
  <attribute name="firstname"
             description="Firstname" list="true" />
  <attribute name="lastname"
             description="Lastname"
             identifier="true" />
</mbot>
```

Listing 8-2: Beispielhafte MBOT Spezifikation mit MBOT-Attributen

Quelle: Eigene Darstellung

Nachdem die Spezifikation der funktionalen Konzepte “Mobile Business Object Types” und “Attribute” beschrieben wurde, wird nun die Verwendung des funktionalen Konzeptes “Methode“ sowie die dazugehörigen Bedienkonzepte erläutert. Für die verwendeten Methoden wird ein spezifischer Abschnitt in der XML-basierten Spezifikation geschaffen, welcher über das XML-Element **<Methods>** gekennzeichnet wird. In diesem Kapitel wird jede verwendete Methode über ein XML-Element **<Method>** beschrieben. Dieses enthält wiederrum

Attribute, um die Parameter der Methoden festzulegen. Mit dem Attribut „mbot“ wird auf die Verknüpfung zum zugehören MBOT spezifiziert. Um die Verknüpfung zum zugehörigen Bedienkonzept herzustellen, wird das Attribut „type“ eingeführt. Die möglichen Werte dieses Attributes repräsentieren jeweils eines der drei eingeführten Bedienkonzepte. Mögliche Werte sind „sorted-list“, „mbo-formbased-view“ sowie „mbo-formbased-edit“. Bei Verwendung des Bedienkonzeptes “Sortierte Listen-Anzeige” kann zudem über das XML-Attribut “sortorder” die Sortierreihenfolge bestimmt werden. Mögliche Werte sind „ascending“ oder „descending“. Zudem wird über das Attribut “handler” die Verknüpfung zu derjenigen Softwarekomponente hergestellt, welche den Aufruf der zugehörigen Programmierschnittstelle im SAP-ERP-System bewerkstelligt. Ein beispielhafter Auszug aus einer Spezifikation von vier Methoden ist in Listing 8-3 dargestellt.

```
<Methods>
   <Method type="sorted-list"
              sortorder="ascending"
              mbot="customer"
              handler="CustomerListHandler" />
   <Method type="mbo-formbased-view"
              mbot="customer"
              handler="CustomerDetailHandler" />
   <Method type="sorted-list"
              sortorder="descending"
              mbot="employee"
              handler="EmployeeListHandler" />
   <Methode type="mbo-formbased-view"
             mbot="employee"
              handler="EmployeeDetailHandler" />
</Methods>
```

Listing 8-3: Beispielhafter Methodenblock

Quelle: Eigene Darstellung

8.4.4 Implementierung und Einsatz

Gemäß den Ausführungen in Kapitel 8.1.2 umfasst die Phase Implementierung die Transformation der spezifizierten Applikation in die Umsetzungsebene. Hierzu muss ein geeigneter Codegenerator entwickelt werden, welcher die vorgestellte, XML-basierte Notation über geeignete Transformationsregeln in eine ausführbare mobile ERP-Applikation überführt. Unter der Phase Einsatz wird die Nutzung der domänenspezifischen Sprache verstanden. Im Fall der vorliegenden Arbeit wird hierfür eine prototypische Benutzungsschnittstelle entwickelt, um den Codegenerator zu bedienen. Hierdurch soll die Umsetzbarkeit der vorgestellten Konzepte evaluiert werden. Aufgrund des Beschreibungsumfangs wird die Implementierung des Codegenerators und der zugehörigen Benutzungsschnittstelle in Kapitel 9 behandelt.

Da die auf Basis der domänenspezifischen Sprache spezifizierten XML-Dokumente als Eingabe für den Codegenerator dienen, ist die *Gültigkeit* der XML-Dateien wichtig. Unter einem gültigen XML-Dokument wird in diesem Kontext verstanden, das die richtigen Elemente und Attribute in der richtigen Reihenfolge verwendet werden (Sebestyen 2010, 73). Um die Gültigkeit eines XML-Dokumentes zu prüfen existieren im XML-Umfeld sogenannte DTDs (Document Type Definition) und XML-Schemas (Bach 2000, 41 f.; Sebestyen 2010, 73 ff.). Während sich DTDs auf die strukturelle Gültigkeit beschränken, können mit Hilfe von

XML-Schemas auch die Datentypen der Elemente und Attribute vorgegeben werden (Bach 2000, 41 f.). Mit Hilfe eines XML-Schemas wäre es somit möglich, die Gültigkeit spezifizierter mobiler ERP-Applikationen vor der Eingabe in den Codegenerator zu prüfen. Es ist jedoch fraglich, ob Endbenutzer mit der Erstellung einer XML-basierten Textdatei zurechtkommen würden. Aus Befragung II ging hervor, dass ein geeignetes Entwicklungswerkzeug für Endbenutzer einfach bedienbar und Fehlerquellen bestmöglich vermeiden sollte (siehe Kapitel 4.3). Diese Ansprüche können beim Einsatz eines Texteditors nicht erfüllt werden. Aus diesem Grund wird in dieser Arbeit keine Möglichkeit für Endbenutzer bereitgestellt, XML-basierte Applikationsspezifikationen direkt zu erstellen. Stattdessen wird der Ansatz verfolgt, dass eine geeignete Benutzungsschnittstelle für Endbenutzer bereitgestellt wird, über welche letztlich die XML-basierte Spezifikation generiert wird. Die XML-basierten Spezifikationen sind somit für die Endbenutzer nicht sichtbar und somit irrelevant. Sie dienen aus Perspektive der Architektur jedoch dem Zweck, dass die Applikationen plattformübergreifenden spezifiziert werden können. Da die Gültigkeit der XML-basierten Spezifikation in der gewählten Umsetzungsvariante jedoch direkt über das Entwicklungswerkzeug sichergestellt werden kann, ist die Erstellung eines XML-Schemas nicht notwendig.

8.5 Fazit und Diskussion

In diesem Kapitel wurden zunächst die Idee sowie die Vorteiler einer domänenspezifischen Sprache beschrieben. Zudem wurde ein Vorgehen aus der Literatur vorgestellt, um neue domänenspezifische Sprachen zu entwickeln. Anschließend wurden ausgewählte domänenspezifische Sprachen aus drei Forschungsarbeiten vorgestellt. Dabei hat sich bestätigt, dass eine Fokussierung einer domänenspezifischen Sprache auf ausgewählte Anwendungsszenarien sinnvoll ist. Zudem verwenden zwei der untersuchten Forschungsarbeiten eine XML-basierte Notationsform für ihre domänenspezifischen Sprachen. Es wurde auch deutlich, dass eine geeignete Benutzungsschnittstelle für Endbenutzer entscheidend für den Erfolg der domänenspezifischen Sprache ist.

Auf Basis dieser Erkenntnisse wurde eine eigene domänenspezifische Sprache gemäß der vorgestellten Vorgehensweise von Merrnik (2005, 320 ff.) (siehe Kapitel 8.3) entwickelt. Grundlage für verwendeten Sprachelemente waren die identifizierten Charakteristiken mobiler ERP-Applikationen aus Kapitel 3. Als resultierende Sprachelemente wurden eine Reihe von funktionalen Konzepten und Bedienkonzepten vorgestellt.

Aufgrund der geringeren Flexibilität und der nicht voraussetzbaren HTML5-Kenntnisse der Endbenutzer wurde entschieden, die domänenspezifische Sprache nicht auf Basis einer GPL-Spracherweiterung zu entwickeln. Stattdessen wurde eine eigenständige Sprache entwickelt, welche eine XML-basierte Notationsform besitzt. Der höheren Flexibilität steht jedoch eine höhere Komplexität des benötigen Codegenerators gegenüber. Die Nutzung von XML als Notationsform hat den Vorteil, dass es sich hierbei um ein verbreitetes Format handelt. Für die Verarbeitung von XML-Dokumenten stehen eine Vielzahl von Werkzeugen und Bibliotheken für gängige Programmiersprachen zur Verfügung. Dies hilft dabei, den Aufwand zu reduzieren. Zudem wurde der Sprachumfang auf Produktivitätsapplikationen mit bestimmten

Funktionalitäten begrenzt. Als Folge konnte die domänenspezifische Sprache auf vier funktionale Konzepte und drei Bedienkonzepte beschränkt werden. Die Analyse in Kapitel 3.3 hat jedoch gezeigt, dass dies die häufigsten Anwendungsfälle abdeckt. Dies führt zu einer weiteren Einschränkung der Komplexität des zu entwickelnden Codegenerators. Die Architektur des Codegenerators, dessen technologische Basis sowie die zugehörige Benutzungsschnittstelle werden im nächsten Kapitel beschrieben.

9 Architektur und prototypische Implementierung des Codegenerators

In diesem Kapitel wird die Konzeption des Codegenerators beschrieben, welcher aus einer spezifizierten mobilen ERP-Applikation eine lauffähige Applikation generieren soll. Mit dem Ziel die Umsetzbarkeit des vorgestellten Konzeptes zu demonstrieren wird der Codegenerator zudem prototypisch implementiert. Dieses Kapitel beschreibt die hierbei verwendete technische Umgebung sowie die verwendete Implementierungsvariante. Ein größerer Teil der Ergebnisse wurde bereits im Rahmen des Veröffentlichungsprozesses dieser Dissertation in folgendem Beitrag publiziert: (Homann et al. 2014b).

9.1 Konzeption der Architektur

Die erste Designentscheidung bei der Gestaltung des Codegenerators ist das Zielformat. Hierbei muss gemäß Anforderung 1 (siehe Kapitel 5.2.1) berücksichtigt werden, dass die generierten Applikationen sowohl auf iOS, als auch auf Android lauffähig sein müssen. Ferner müssen die notwendigen Bedienkonzepte (siehe Kapitel 8.4.2.4) mit dem gewählten Zielformat umsetzbar sein.

9.1.1 Zielformat für die Benutzungsschnittstelle der mobilen ERP-Applikationen

Um eine mobile Applikation zu generieren, welche sowohl auf iOS, als auch auf Android lauffähig ist, eignet sich ein HTML5-Framework für mobile Applikationen. In Kapitel 8.4.3.1 wurden die am weitesten verbreiteten Vertreter iWebKit, jQT, Sencha Touch und jQuery Mobile vorgestellt und diskutiert. Dabei wurden auch die Vorteile von jQuery Mobile für die Aufgabenstellungen dieser Arbeit beschrieben. Da jQuery Mobile auch alle für die Umsetzung der geforderten Bedienkonzepte notwendigen Anzeige- und Bedienelemente bereitstellt, eignet es sich als Zielformat für die Umsetzung der Benutzungsschnittstelle. Zudem handelt es beim Quellcode für Applikationen auf Basis von jQuery Mobile ausschließlich um ein textbasiertes Format, welches in vielen Fällen nur aus einer einzigen HTML-Datei besteht. Dies reduziert die Komplexität für die Umsetzung der Transformationsschritte des Codegenerators.

9.1.2 Aufruf der Programmierschnittstellen des SAP-ERP-Systems

Wie in Kapitel 2.1.5.2.1 dargestellt, erfolgt der Datenaustausch mit dem SAP-ERP-System über sogenannte BAPIs. Für die Nutzung der BAPIs stehen unterschiedliche Alternativen zur Auswahl. Die gängigste Alternative ist die Nutzung des proprietären SAP Netzwerkprotokolls RFC (siehe Kapitel 2.1.5.2.2). Um dieses Netzwerkprotokoll zu nutzen hat SAP für viele Programmiersprachen entsprechende Programmierbibliotheken bereitgestellt. Eine weitere Alternative ist die Nutzung von SOAP-basierten Webservices. In SAP-ERP-Systemen ist es möglich, aus BAPIs eine entsprechende WSDL-Spezifikation für einen zugehörigen Webservice zu generieren, welcher anschließend über das SOAP-Protokoll angesprochen werden kann (siehe Kapitel 2.1.5.2.2). Eine dritte Alternative ist die Nutzung von RESTful-Webservices. Hierzu muss allerdings eine zusätzliche SAP-Komponente, der sogenannte SAP NetWeaver Gateway, installiert und konfiguriert werden (siehe Kapitel 2.1.5.2.2).

Die primäre Programmiersprache in jQuery Mobile ist JavaScript. Da mit JavaScript die Nutzung von SOAP möglich ist, wird in der prototypischen Implementierung der Aufruf der SAP-ERP Programmierschnittstellen über SOAP-basierte Webservices gewählt.

9.1.3 Transformationswerkzeug

Nach Festlegung des Quell- und Zielformates besteht die wesentliche Aufgabe bei der Gestaltung des Codegenerators darin, den Transformationsprozess zwischen Quell- und Zielformat umzusetzen. Beim Quellformat handelt es sich um ein XML-Dokument. Beim Zielformat um ein HTML-Dokument. Da es sich bei beiden Formaten um textbasierte Formate handelt, ist die Komplexität des Transformationsprozesses geringer als beispielsweise bei Text-zu-Binär-Transformationen (vgl. z.B. (Sebestyen 2010, 291 ff.)). Gesucht wird ein Werkzeug, mit welchem sich Transformationsregeln von einem textbasierten Format in ein anderes textbasiertes Format definieren lassen.

Um das XML-basierte Quellformat einzulesen und daraus das gewünschte HTML-basierte Zielformat zu erzeugen, ist eine Variante die Verwendung von sogenannten XML-Parsern bzw. XML-Prozessoren[51] in gängigen Programmiersprachen. Ein XML-Parser stellt eine API für das Einlesen von XML-Dokumenten und dem anschließenden Zugriff auf ihre Struktur und ihren Inhalt zur Verfügung (Sebestyen 2010, 58). Dabei wird zwischen DOM-basierten und SAX-basierten Parsern unterschieden (Ullenboom 2012, 1150). DOM steht für „Document Object Model“ und ist ein Standard des W3C, welcher die Datenstruktur eines eingelesenen XML-Dokumentes als Baumstruktur im Hauptspeicher eines Computers ablegt (Bach 2000, 47 f.). Zusätzlich enthält eine DOM-API Funktionalitäten zum traversieren der Baumstruktur im Hauptspeicher. Damit kann zu den einzelnen Elementen navigiert werden und beim Auftreten bestimmter Muster ein entsprechender HTML-Code in eine Textdatei geschrieben werden.

[51] Die Begriffe XML-Parser und XML-Prozessor werden als Synonyme verwendet (Sebestyen 2010, 59)

Die zweite Parser-Variante sind sogenannte SAX-Parser. SAX steht für „Simple API for XML Parsing“. SAX wurde von David Meggison entwickelt und steht zur freien Verwendung zur Verfügung (Ullenboom 2012, 1150). Bei SAX steht die Reduktion von Hauptspeicherressourcen und die schnelle Verarbeitung von XML-Dokumenten im Vordergrund (Ullenboom 2012, 1150). Im Gegensatz zu DOM-Parsern arbeiten SAX-Parser ereignisorientiert und verzichten auf das vollständige Einlesen eines XML-Dokumentes in den Hauptspeicher. Bei den Ereignissen handelt es sich um das Auftreten eines bestimmten XML-Elementes. Falls ein solches Ereignis eintritt, können entsprechende Programmroutinen ausgeführt werden. Im Kontext des Codegenerators ist es durch diese Programmroutinen möglich, beim Auftreten bestimmter XML-Elemente einen zugehörigen HTML-Code in eine Textdatei zu schreiben. Damit wäre es mit beiden Parser-Varianten möglich, die benötigten Transformationsregeln in einer Programmiersprache umzusetzen, für welche entsprechende XML-Parser zur Verfügung stehen.

Eine Alternative zur Verwendung von XML-Parsern in Programmiersprachen bietet die Nutzung der *Extensible Stylesheet Language Transformation*, kurz *XSLT*. XSLT ist eine Programmiersprache zur Transformation von XML-Dokumenten und wird häufig verwendet, um XML-Dokumente von einem XML-Format in ein anderes XML- oder Textformat zu transformieren (Bach 2000, 119; Fitzgerald 2004, 1; Sebestyen 2010, 195). XSLT ist Teil der *Extensible Stylesheet Language* Sprachfamilie, kurz *XSL*. Die verschiedenen XSL-Sprachen werden für die Formatierung von XML-Dokumenten für unterschiedliche Ausgabemedien, wie bspw. für den Druck oder Bildschirmausgaben, verwendet (Sebestyen 2010, 195). Für diese Aufgabe sind i.d.R. zwei Schritte notwendig: in einem ersten Schritt wird die Struktur des XML-Dokumentes angepasst, in einem zweiten Schritt erfolgt die eigentliche Formatierung. Der erste Schritt wird üblicherweise unter Nutzung von XSLT durchgeführt. Für den zweiten Schritt wurde XSL-FO (XSL Formatting Objects) entwickelt (Bach 2000, 116 f.; Sebestyen 2010, 195 f.).

Bei XSLT handelt es sich um eine *deklarative Programmiersprache*. Im Gegensatz zu imperativen Programmiersprachen, wie bspw. Java oder JavaScript, in welchen der Programmierer jeden auszuführenden Schritt festlegt, genügt bei XSLT die Beschreibung der Transformationsregel. Die eigentliche Ausführung der beschriebenen Transformationsregeln wird durch einen sogenannten XSLT-Prozessor bewerkstelligt (Sebestyen 2010, 202).

Bei XSLT-Prozessoren handelt es sich um Softwareanwendungen, welche i.d.R. über die Kommandozeile eines Betriebssystems angesprochen werden können. Zudem haben gängige Webbrowser[52] oftmals einen integrierten XSLT-Prozessor (Sebestyen 2010, 197). Zudem stellen XSLT-Prozessoren i.d.R. APIs für gängige Programmiersprachen bereit. Beispiele für XSLT-Prozessoren sind Xalan[53] der Apache Software Foundation, MSXML[54] von Microsoft oder Saxon[55] von Michael Kay.

[52] Beispielsweise wird XSLT von den Webbrowsern Internet Explorer, Mozilla Firefox, Opera und Safari unterstützt (Sebestyen 2010, 200)
[53] http://xalan.apache.org, zugegriffen am 10.02.2014

Die Implementierung von Transformationsregeln in XSLT erfolgt durch die Entwicklung sogenannter XSLT-Stylesheets (Bach 2000, 119). Stylesheets werden selbst in XML geschrieben (Bach 2000, 119; Sebestyen 2010, 203). In einem XSLT-Stylesheet werden Transformationsregeln über das XML-Element **`<xsl:template>`** definiert. In diesem Element wird über ein Muster (engl. pattern) festgelegt, in welchem Fall eine Transformationsregel angewendet wird, z.B. wenn ein bestimmtes XML-Element auftritt oder ein XML-Attribut einen bestimmten Wert besitzt (Bach 2000, 123 ff.; Fitzgerald 2004, 84 f.). Findet eine solche Übereinstimmung zwischen dem Muster und einem Fragment des eingelesenen XML-Dokumentes statt, so wird von einer Übereinstimmung (engl. match) gesprochen (Bach 2000, 124).

Zu Beginn des XSLT-Transformationsprozesses parst der XSLT-Prozessor sowohl das Quelldokument, als auch das XSLT-Stylesheet und speichert beide in einer DOM-Struktur. Anschließend wird das Quelldokument ausgehend von seinem Wurzelknoten traversiert. Hierbei wird bei jedem Element geprüft, ob eine Übereinstimmung mit einem im XSLT-Stylesheet definierten Muster vorliegt. Liegt eine solche Übereinstimmung vor, so werden die in der Transformationsregel festgelegten Anweisungen ausgeführt (Fitzgerald 2004, 119 ff.). Abbildung 9-1 illustriert den beschriebenen XSLT-Transformationsprozess.

Die Nutzung von XSLT ist prinzipiell ähnlich zur beschriebenen Alternative, der Implementierung der Transformationsregeln in einer Programmiersprache wie Java oder C#. Der Vorteil von XSLT ist jedoch, dass XSLT-Prozessoren für unterschiedliche Betriebssysteme existieren und der Transformationsprozess sowohl über eine Kommandozeile, als auch über eine Programmiersprache oder einen Webbrowser durchgeführt werden kann. Da für das geplante Vorhaben keine Nachteile von XSLT gegenüber einer Implementierung der Transformation durch eine Programmiersprache erkennbar sind, wird im Folgenden XSLT zur Umsetzung des Transformationsprozesses verwendet.

[54] http://msdn.microsoft.com/en-us/library/ms763742(v=vs.85).aspx, zugegriffen am 10.02.2014
[55] http://saxon.sourceforge.net, zugegriffen am 10.02.2014

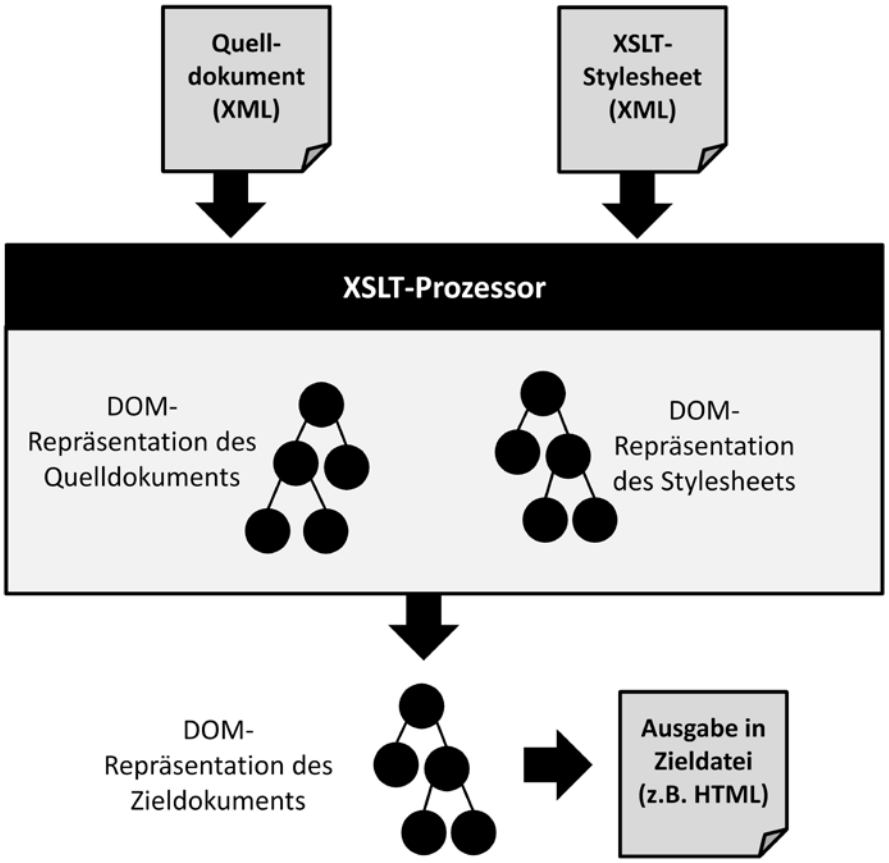

Abbildung 9-1: XSLT-Transformationsprozess
Quelle: In Anlehnung an (Bach 2000, 120)

9.1.4 Architektur der mobilen ERP-Applikationen

Um den Transformationsprozess umsetzen zu können, müssen weitere Details zum gewünschten Transformationsergebnis, in diesem Fall den mobilen ERP-Applikationen, erarbeitet werden. Hierzu ist es notwendig, die Architektur der mobilen ERP-Applikationen zu konzipieren.

Um eine möglichst übersichtliche und ausbaufähige Architektur für die mobilen ERP-Applikationen zu konzipieren, sollte die Datenabfrage vom SAP-ERP-System und die interne Repräsentation der abgefragten Daten von der Benutzungsschnittstelle der mobilen ERP-Applikation abgekoppelt werden. Hierdurch wäre es möglich mit einer geringen Anzahl von wiederverwendbaren Bausteine für die Benutzungsschnittstelle eine vergleichsweise hohe Anzahl an mobilen ERP-Applikationen umsetzen zu können. Durch eine solche Abkopplung kann bei der Umsetzung eine flexible Verknüpfung zwischen der Datenschnittstelle und Benutzungsschnittstelle realisiert werden. Aufgrund der identifizierten Ähnlichkeit der Benutzungsschnittstellen mobiler ERP-Applikationen (siehe Kapitel 3.3.2.1) erscheint dieses

Vorgehen sinnvoll. Für den Zweck der Abkopplung zwischen Datenschnittstelle und Benutzungsschnittstelle ist das *Model-View-Controller-Entwurfsmuster*, kurz *MVC*, geeignet.

Das MVC-Entwurfsmuster enthält die folgenden drei Bestandteile (Fowler et al. 2002, 330 ff.; Lahres/Rayman 2009, 511 ff.):

- **Model:** Das Model repräsentiert die Datenschnittstelle. Es enthält alle Datensätze, welche über die Benutzungsschnittstellen angezeigt und manipuliert werden können (Fowler et al. 2002, 330). In der Regel repräsentiert ein Model einen bestimmten Datenausschnitt eines Backendsystems, z.B. einer Datenbank. Das Model kann folglich als eine zusätzliche Abstraktionsschickt verstanden werden, welche vom eigentlichen Datenzugriff auf das Backendsystem abstrahiert.
- **View:** Der View repräsentiert die Benutzungsschnittstelle einer Applikation (Fowler et al. 2002, 330; Lahres/Rayman 2009, 514). Im Falle einer grafischen Benutzungsschnittstelle stellt er hierfür entsprechende Anzeige- und Bedienelemente bereit (Krasner/Pope 1988, 28; Fowler et al. 2002, 330).
- **Controller:** Der Controller agiert als Vermittler zwischen Model und View und steuert die Interaktion mit dem Nutzer (Fowler et al. 2002, 330; Lahres/Rayman 2009, 514). In dieser Funktion sendet er bspw. Informationen über eine gewünschte Datenänderung oder –anzeige des Nutzers an das Model oder sorgt bei einer Datenänderung für eine Aktualisierung des Views (Fowler et al. 2002, 330).

Abbildung 9-2 illustriert die einzelnen Bestandteile des MVC-Entwurfsmusters sowie deren Aufgabenteilung.

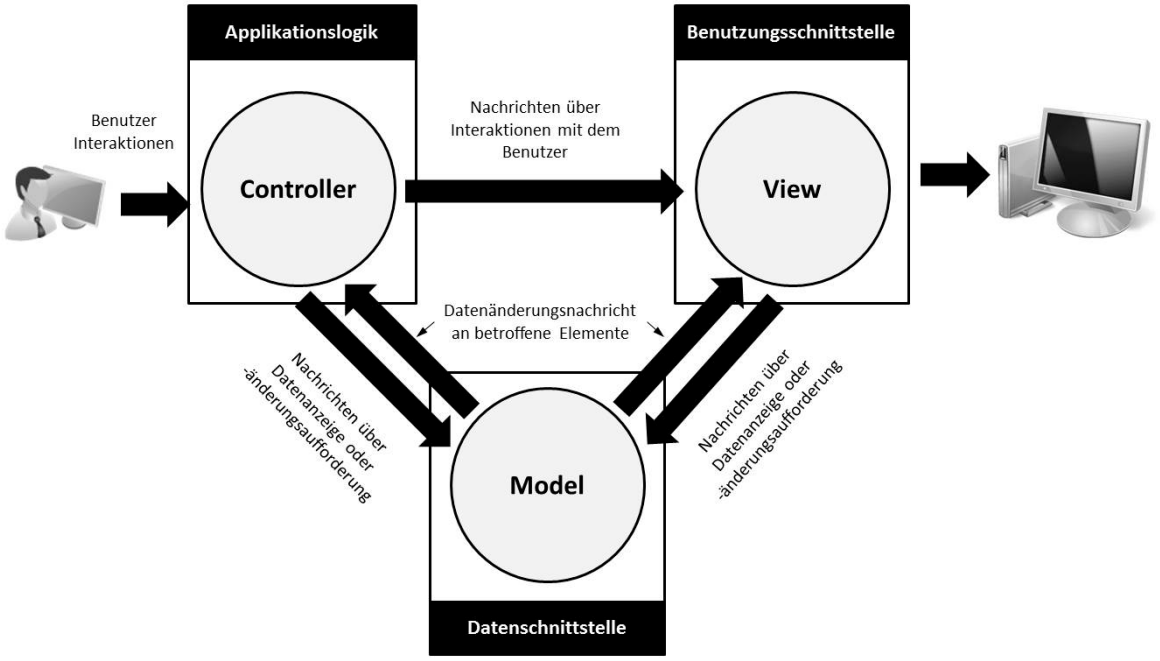

Abbildung 9-2: Model-View-Controller-Entwurfsmuster
Quelle: Eigene Darstellung in Anlehnung an (Krasner/Pope 1988, 30)

Im Fall der vorliegenden Arbeit wird ein SAP-ERP-System als Backendsystem verwendet. Bei der zughörigen Datenschnittstelle handelt es sich um einen BAPI aus einem SAP-ERP-System. Demzufolge repräsentiert das *Model* einer mobilen ERP-Applikation einen abstrahierten Datenzugriff auf einen BAPI eines SAP-ERP-Systems.

Der *View* der mobilen ERP-Applikationen wird durch eine Kombination der gewünschten Bedienkonzepte (siehe Kapitel 8.4.2.4) umgesetzt. Er wird durch zugehörige HTML5-Seiten implementiert, welche auf Basis des jQuery Mobile Frameworks erstellt wurden.

Die Applikationslogik einer mobilen ERP-Applikation wird durch den *Controller* bereitgestellt. Dies umfasst im Wesentlichen die Umsetzung der unter Kapitel 8.4.2.3.3 vorgestellten MBOT-Methoden. Zusätzlich werden Prüfroutinen durch den Controller umgesetzt, um beispielsweise die Eingabe fehlerhafter Datensätze zu verhindern.

Abbildung 9-3 illustriert die Anwendung des MVC-Entwurfsmusters zur Gestaltung der Architektur mobiler ERP-Applikationen im Kontext der vorliegenden Arbeit.

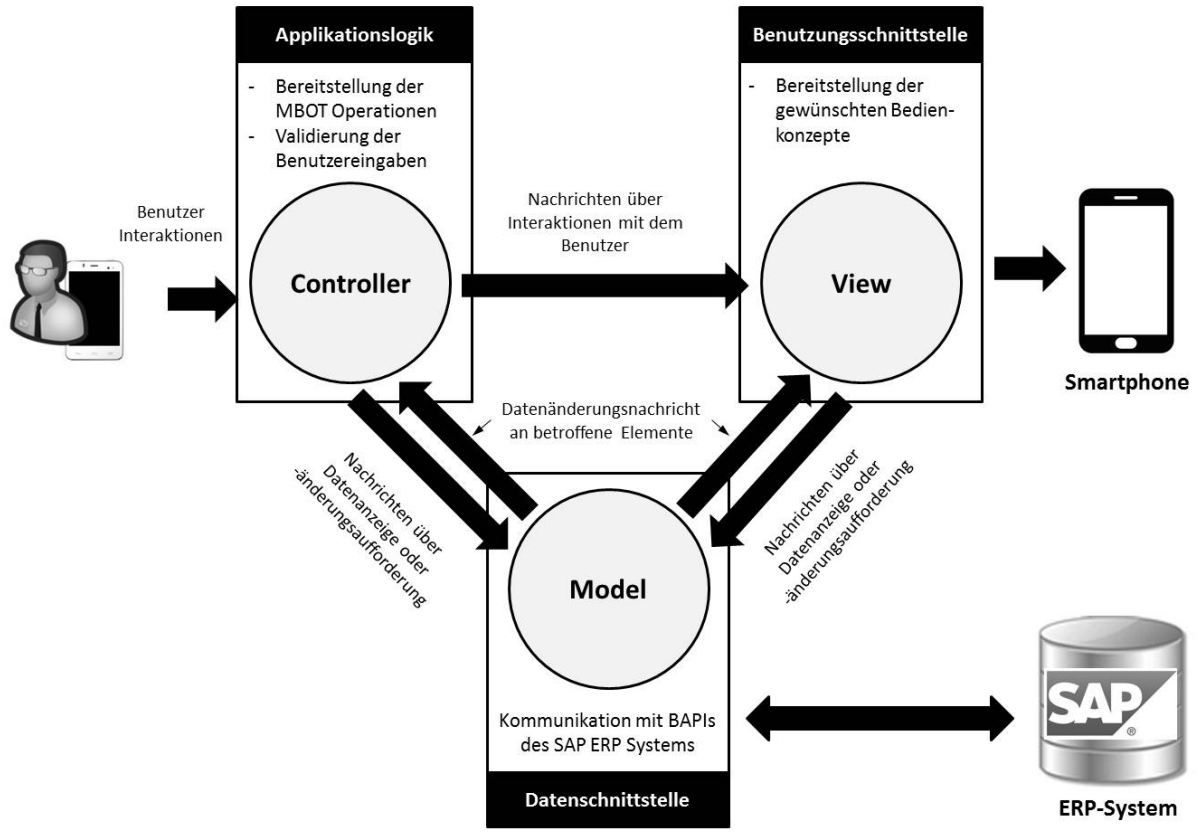

Abbildung 9-3: Anwendung des MVC-Entwurfsmusters für mobile ERP-Applikationen
Quelle: Eigene Darstellung in Anlehnung an (Krasner/Pope 1988, 30)

9.1.5 Architekturentwurf des Entwicklungswerkzeuges

Bei der Architekturgestaltung des Entwicklungswerkzeuges müssen mehrere Designentscheidungen getroffen werden. Diese werden im Folgenden diskutiert. Daraus wird schließlich ein Lösungsvorschlag abgeleitet.

Die erste Designentscheidung betrifft die Kommunikationsbeziehung mit dem SAP-ERP-System. Wie bereits in früheren Kapiteln erläutert (siehe z.B. Kapitel 2.1.5.2.1) stellen SAP-ERP-Systeme Programmierschnittstellen in Form von BAPIs zur Verfügung. Diese sind aktuell über folgende Varianten ansprechbar:

(1) Über das proprietäre SAP-Netzwerkprotokoll RFC

(2) Über das HTTP-basierte, standardisierte Netzwerkprotokoll SOAP

(3) Über HTTP-basierte RESTful Webservices

Für Variante (1) ist eine spezielle Programmierbibliothek notwendig, welche eine entsprechende API zur Verwendung des RFC-Netzwerkprotokolls bereitstellt. Für diese Zwecke existieren bspw. Programmierbibliotheken für die Programmiersprachen Java (Wegelin/Englbrecht 2011, 279 ff.) und C (Wegelin/Englbrecht 2011, 163 ff.). Für Variante (3) ist hingegen die Installation einer speziellen SAP Komponente als Erweiterung im SAP-ERP-System, das sogenannte SAP NetWeaver Gateway, notwendig (vgl. z.B. (Homann et al. 2013b, 44 ff.)). Einzig bei Variante (2) ist keine zusätzliche Komponente notwendig. Zusätzlich handelt es sich bei SOAP um ein standardisiertes, XML-basiertes Netzwerkprotokoll, welches von einem Großteil verfügbarer Programmiersprachen unterstützt wird. Zur Nutzung von SOAP muss lediglich ein existierender BAPI über eine Exportfunktion in eine WSDL-Beschreibung exportiert werden. Anschließend ist der zugehörige BAPI als Webservice über SOAP ansprechbar. Aufgrund der vergleichbar geringen Abhängigkeiten und der guten Unterstützung durch verschiedene Programmiersprachen wird die Kommunikation mit dem SAP-ERP-System im Folgenden über SOAP durchgeführt.

Eine weitere Designentscheidung betrifft den Typ der Benutzungsschnittstelle des Entwicklungswerkzeuges. Hierbei wäre prinzipiell eine Desktop Anwendung für ein spezielles Betriebssystem, eine betriebssystemübergreifende Desktop Anwendung oder eine Webanwendung möglich. Unter einer Desktop Anwendung für ein spezielles Betriebssystem wird beispielsweise eine Anwendung für Mac OS X oder Windows verstanden. Dieser Anwendungstyp ist vergleichbar zu nativen Applikationen auf einem mobilen Betriebssystem und ermöglicht den größten Funktionsumfang und die komfortabelste Bedienung Derartige Anwendungen werden mit herstellerspezifischen Entwicklungswerkzeugen wie Microsoft Visual Studio oder XCode entwickelt. Ein Beispiel für betriebssystemübergreifenden Desktop Anwendungen sind Java Anwendungen. Hierzu muss eine sogenannte Java Virtual Machine (JVM) installiert werden, welche von den jeweiligen Spezifika eines Betriebssystems abstrahiert. Daher sind derartige Anwendungen prinzipiell auf allen Betriebssystemen lauffähig, für welche eine entsprechende JVM bereitsteht. Die dritte Variante ist eine Webanwendung. Diese wird mit Webtechnologien wie HTML, CSS und JavaScript erstellt und über einen Webserver bereitgestellt. Der Zugriff erfolgt über einen Webbrowser. Neben der Betriebssystemunabhängigkeit ist ein weiterer Vorteil dieser Variante die gute Administrierbarkeit. Während die Softwareanwendungen in den beiden anderen Varianten lokal auf den Computern der Endbenutzer installiert werden müssen, was die Wartung erschwert, werden Webanwendungen zentral auf einem Webserver bereitgestellt und können dadurch auch zentral überwacht und ggf. optimiert werden. Da die gewünschte Benutzungsschnittstelle für das Entwicklungswerkzeug mit Webtechnologien realisierbar ist und auch aufgrund der Entscheidung für jQuery Mobile für die Benutzungsschnittstelle der mobilen ERP-Applikationen der Einsatz eines Webservers bereits notwendig ist, wird die Benutzungsschnittstelle des Entwicklungswerkzeug als Webanwendung realisiert.

Um die Komplexität eines verteilten Anwendungssystems, wie dem vorliegenden, zu reduzieren hat sich eine sogenannte *Middleware* bewährt (Ferstl/Sinz 2008, 409 ff.; Krcmar 2010, 348). Durch den Einsatz einer Middleware können unter anderem die Kommunikationsverbindungen in einem verteilten Anwendungssystem vereinfacht werden oder eine Laufzeitumgebung für Komponenten des verteilten Anwendungssystems bereitgestellt werden. Derartige

Komponenten können spezifische Aufgaben übernehmen, z.B. den Zugriff auf das Backendsystem (Ferstl/Sinz 2008, 409 f.). Im vorliegenden Fall sollte eine Middleware-Lösung zum einen den SOAP-basierten Zugriff auf das SAP-ERP-System umsetzen. Zudem anderen sollte eine geeignete Middleware-Lösung Funktionalitäten zur Bereitstellung von Webapplikationen bieten. Dadurch fungiert sie als Laufzeitumgebung für die webbasierten mobilen ERP-Applikationen, als auch als Laufzeitumgebung für die webbasierte Benutzungsschnittstelle des Entwicklungswerkzeuges.

Abbildung 9-4 illustriert den beschriebenen Architekturentwurf für das Entwicklungswerkzeuges.

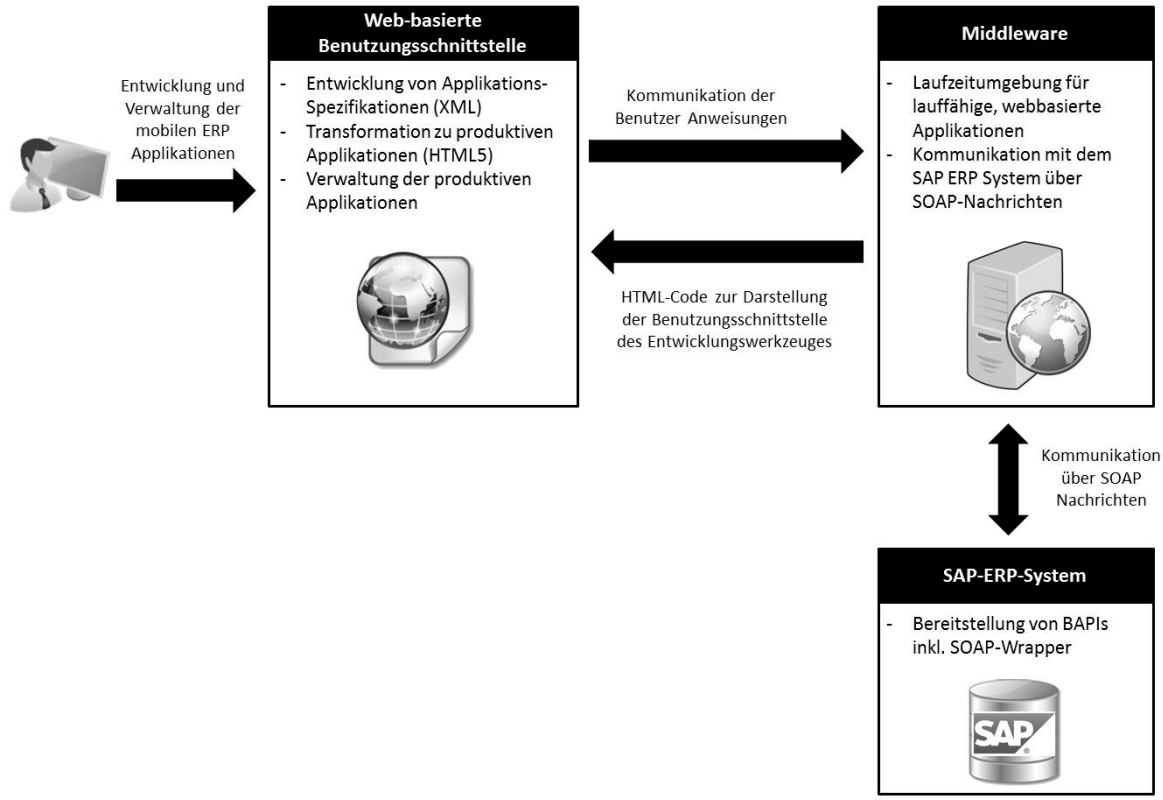

Abbildung 9-4: Architekturentwurf für das Entwicklungswerkzeug
Quelle: Eigene Darstellung

9.2 Prototypische Umsetzung

Nachdem im vorherigen Kapitel der Architekturentwurf für das Entwicklungswerkzeug skizziert wurde, wird in diesem Kapitel dessen prototypische Umsetzung beschrieben. Bestandteile der Beschreibung sind die eingesetzten Technologien, das implementierte XSLT-Stylesheet zur Transformation der XML-basierten Applikationsspezifikation in eine ausführbare Applikation sowie die implementierte Benutzungsschnittstelle zur Nutzung des Codege-

nerators. Die prototypische Umsetzung des Codegenerators wurde in größeren Teilen gemeinsam mit dem damaligen Studenten Malte Sieb (2012) als Teil seiner Bachelorarbeit durchgeführt.

9.2.1 Technologische Basis und Aufbau der Testumgebung

Um den skizzierten Architekturentwurf aus Kapitel 9.1.5 umzusetzen, ist eine geeignete technologische Basis für die prototypische Umsetzung der Middleware notwendig. Eine geeignete technologische Basis muss eine Laufzeitumgebung für webbasierte Applikationen bereitstellen. Hierzu eignet sich ein sogenannte *Webserver*. Bei einem Webserver handelt es sich um eine Software- und/oder Hardware-Lösung zur Verwaltung von Webinhalten, welche von Clients, i.d.R. Webbrowsern, abgerufen werden (Prevezanos 2012, 746). Verbreitete Webserver sind bspw. der Internet Information Services (IIS) Server von Microsoft oder der Apache HTTP Server der Apache Foundation. Während es sich beim IIS um eine kommerzielle Softwarelösung handelt, ist der Apache HTTP Server eine quelloffene und freiverfügbare Softwarelösung.

Neben der Bereitstellung einer Laufzeitumgebung für Webapplikation muss eine geeignete technologische Basis zudem Funktionalitäten bereitstellen, um eine SOAP-basierte Kommunikation mit einem SAP-ERP-System zu ermöglichen. Die Architektur des Apache HTTP Servers ist modular aufgebaut. Zusätzlich benötigte Funktionalitäten können über entsprechende Module hinzugefügt werden (Coar/Bowen 2003, xi). Dies ermöglicht es beispielsweise Module zur Verarbeitung von XML-Dokumenten hinzuzufügen. Zudem existieren für den Apache HTTP Server Module für den Einsatz serverseitige Skriptsprachen wie beispielsweise PHP, Perl, Python oder Ruby. Durch den Einsatz einer serverseitigen Skriptsprache können dynamische Webinhalte erzeugt werden und Anwendungsfunktionalitäten implementiert werden. Durch den Einsatz einer serverseitigen Skriptsprache ist zudem die SOAP-basierte Kommunikation mit einem SAP-ERP-System umsetzbar. Aufgrund seiner freien Verfügbarkeit, seiner weiten Verbreitung und modulbasierten Erweiterbarkeit wird der Apache HTTP Server als Middleware-Lösung für die prototypische Implementierung dieser Arbeit verwendet.

Um eine funktionstüchtige Installation des Apache HTTP Servers mit einer serverseitigen Skriptsprache durchzuführen, sind eine Reihe von Installations- und Konfigurationsschritten notwendig. Eine komfortable Installationsvariante wird durch das XAMPP[56] Projekt der gemeinnützigen Online-Projektgruppe „Apache Friends“ angeboten. Der Name XAMPP steht für die Anfangsbuchstaben der Hauptkomponenten: dem Apache HTTP Server, dem Datenbankserver MySQL und der beiden serverseitigen Skriptsprachen Perl und PHP. Das „X“ symbolisiert einen Platzhalter und soll ausdrücken, dass XAMPP für unterschiedliche Betriebssysteme verfügbar ist. XAMPP ermöglicht eine komfortable Installation und Konfiguration der enthaltenen Komponenten und dient zum Aufbau einer Umgebung für die Bereitstellung dynamischer Webseiten (Brinzarea-Iamandi et al. 2009, 280 ff.). Für den Aufbau der Testumgebung dieser Arbeit wurde XAMPP in der Version 1.8.2 für Microsoft Windows

[56] http://www.apachefriends.org/de/index.html, zugegriffen am 10.12.2013

verwendet, welche im Oktober 2013 die aktuelle Version war. Diese wurde auf einem Windows Server 2008 R2 Betriebssystem installiert. Die XAMPP Version 1.8.2 beinhaltet die für diese Arbeit notwendigen Komponenten in folgenden Versionen:

- Apache HTTP Server in Version 2.4.4
- MySQL Datenbankserver in der Version 5.5.32
- PHP Modul in der Version 5.4.22

Um die Verarbeitung der XML-basierten Applikationsspezifikationen zu ermöglichen wird zusätzlich die Programmierbibliothek *libxml*[57] in der Version 2.5.4 verwendet. Um eine XSLT-Verarbeitung zu ermöglichen wird die Bibliothek *libxslt*[58] in der Version 1.1.28 verwendet.

Als ERP-System wird ein SAP-ERP-System mit nachfolgender Ausprägung verwendet:

- Komponentenversion: SAP Enterprise Core Component (ECC) in Version 6.0
- Datenbank: IBM DB2 Enterprise Server Edition (ESE) Version in 9.7
- Betriebssystem: AIX Version 6.1 Technology Level (TL) 7

Aufgrund der großen Anzahl von Attributen existierender BOTs in einem SAP-ERP-System ist die Parametrisierung der zugehörigen BAPIs für den Zweck der prototypischen Implementierung aufwändig. Aus diesem Grund wurden im verwendeten SAP-ERP-System eigene BOTs mit zugehörigen BAPIs implementiert, um die Parametrisierung zu vereinfachen. Hierzu wurden BOTs zur Repräsentation von Kunden, Kundenaufträgen, Abteilungen und Mitarbeitern mit beispielhaften Attributen im SAP-ERP-System angelegt. Zudem wurden für jedes BOT entsprechende BAPIs zum Erzeugen, Löschen, Ändern, Lesen, Suchen und Auslesen zugehöriger BOs implementiert. Anschließend wurde für jedes BAPI über den Web Service Wizard der ABAP Workbench eine WSDL-Beschreibung sowie ein zugehöriger Web Service Proxy erzeugt. Dies ermöglicht es, die bereitgestellten BAPIs über SOAP anzusprechen.

Um die Benutzungsschnittstelle des implementierten Entwicklungswerkzeuges zu testen wurde der Webbrowser Firefox in der Version 16.0 verwendet. Für das Testen der entwickelten mobilen ERP-Applikationen wurde ein iPhone 4S mit der installierten iOS Version 6.1.5 und dem integrierten Safari Webbrowser verwendet. Abbildung 9-5 illustriert den beschriebenen Aufbau der Testumgebung.

[57] http://www.xmlsoft.org, zugegriffen am 10.12.2013
[58] http://xmlsoft.org/XSLT, zugegriffen am 10.12.2013

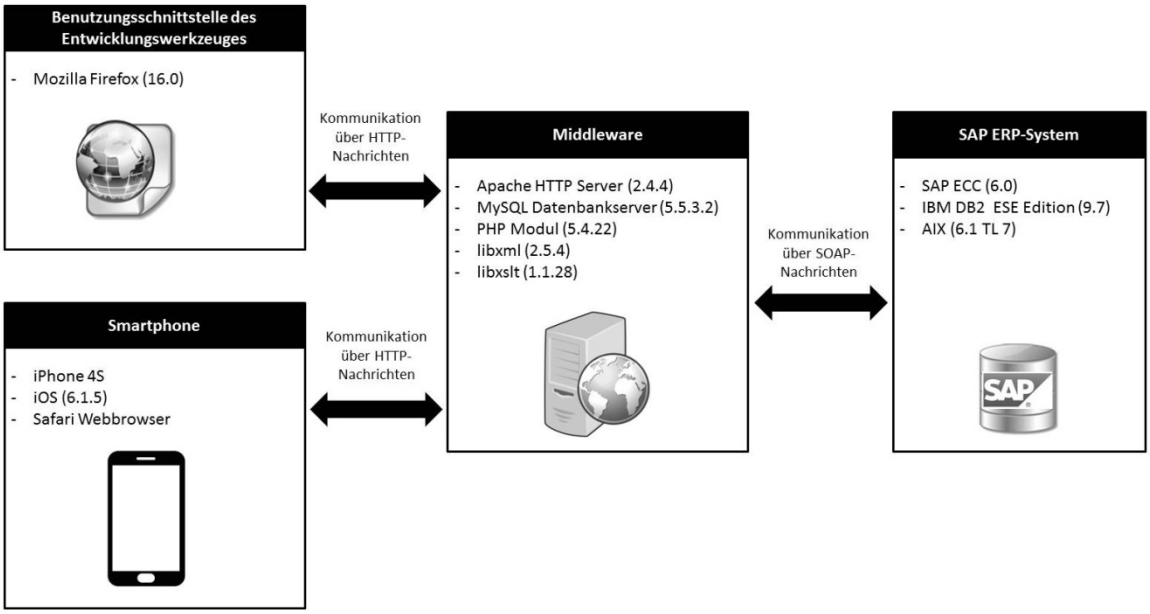

Abbildung 9-5: Aufbau der Testumgebung zur Evaluation des Codegenerators
Quelle: Eigene Darstellung

9.2.2 Bestandteile und Vorgang der Transformation

Um die in Kapitel 9.1.4 beschriebene Architektur für mobile ERP-Applikationen umzusetzen müssen die drei Elemente der MVC-Architektur umgesetzt werden. Im Folgenden wird die gewählte Umsetzungsvariante für die prototypische Implementierung beschrieben. Im Anschluss daran wird der Transformationsvorgang erläutert.

Die prototypische Umsetzung des *Model* besteht aus zwei Bestandteilen: einem PHP-Skript für den Aufruf der BAPIs über SOAP-Nachrichten und einer JavaScript-Klasse zur Repräsentation der MBOs als JavaScript-Objekte. Das PHP-Script stellt für jedes BAPI eine Funktion mit zugehörigen Übergabe- und Rückgabewerten zur Verfügung. Die Definition der Übergabe- und Rückgabewerte erfolgt in Form der *JavaScript Object Notation*, kurz *JSON*. Dabei handelt es sich um ein kompaktes, textbasierte Format (ECMA 2013, ii). JSON ist einfacher gestaltet als XML und ist typisiert. JSON unterstützt nur grundlegende Datentypen (Zahlen, Zeichenketten, Arrays, Objekte, boolesche Werte oder null) (ECMA 2013, 2), welche jedoch für den Zweck der vorliegenden Arbeit ausreichend sind. Um eine einfache Handhabung der MBOs zu ermöglichen, wird für jedes MBOT eine JavaScript-Klasse implementiert. Diese enthält neben den Attributen des MBOT auch zugehörige Getter- und Setter-Methoden zum Auslesen bzw. Ändern der Attributwerte.

Die prototypische Implementierung des *Controllers* dieser Arbeit besteht aus einer JavaScript-Klasse, dem sogenannten Handler. Der Handler einer mobilen ERP-Applikation wird in der XML-basierten Applikationsspezifikation definiert. Er bewerkstelligt die Kommunikation zwischen Model und View. Im Falle der prototypischen Implementierung ruft er die

benötigten Funktionen aus dem PHP-Skript des Modells auf und Instanziiert die MBOs in Form von Instanzen der zugehörigen JavaScript-Klasse des Models. Zusätzlich enthält er Validierungsfunktionalitäten für Benutzereingaben.

Zur prototypischen Implementierung des *Views* wird das HTML5-Framework jQuery Mobile verwendet. Die einzelnen Bildschirmmasken der mobilen ERP-Applikationen werden in jQuery Mobile durch ** ** Bereiche mit der data-role =“page“ repräsentiert. Die Umsetzung der einzelnen Bildschirmmasken orientiert sich an den vorgestellten Bedienkonzepten (siehe Kapitel 8.4.2.4). Hierfür wurden für jedes Benutzungskonzept Codeschablonen im XSLT-Stylesheet implementiert. Die variablen Bestandteile eines Benutzungskonzeptes, wie beispielsweise der Name der genutzten MBOTs oder die MBOT-Attribute werden beim Transformationsvorgang berücksichtigt.

Bei der Entwicklung einer mobilen ERP-Applikation spezifiziert der Endbenutzer über eine bereitgestellte webbasierte Benutzungsschnittstelle die Inhalte der Applikation. Hierbei kann er verfügbare MBOTs auswählen und anschließend die gewünschten Methoden und anzuzeigenden MBOT-Attribute selektieren. Aus diesen Angaben wird eine XML-basierte Spezifikation gemäß der vorgestellten domänenspezifischen Sprachdefinition aus Kapitel 8.4.3.1.2 generiert. Diese wird als Datei mit der Dateiendung „.xml“ und dem Namen der Applikation als Dateiname im Dateiverzeichnis des Apache HTTP Servers gespeichert. In einem zweiten Schritt wird aus der XML-basierten Applikationsspezifikation eine HTML-Datei auf Basis von jQuery Mobile erzeugt. Hierzu wurde ein XSLT-Stylesheet entwickelt, welches die Codeschablonen für die notwendigen Bedienkonzepte enthält. Im Rahmen der Transformation wird der XSLT-Prozessor aus der verwendeten libxslt Programmierbibliothek aufgerufen und die Applikationsspezifikation übergeben. Der XSLT-Prozessor generiert auf Basis der Applikationsspezifikation und des XSL-Stylesheets eine Datei mit der Dateiendung „.html“ und dem Namen der Applikation als Dateinamen. Der beschriebene Spezifikations- und Transformationsvorgang ist in Abbildung 9-6 illustriert.

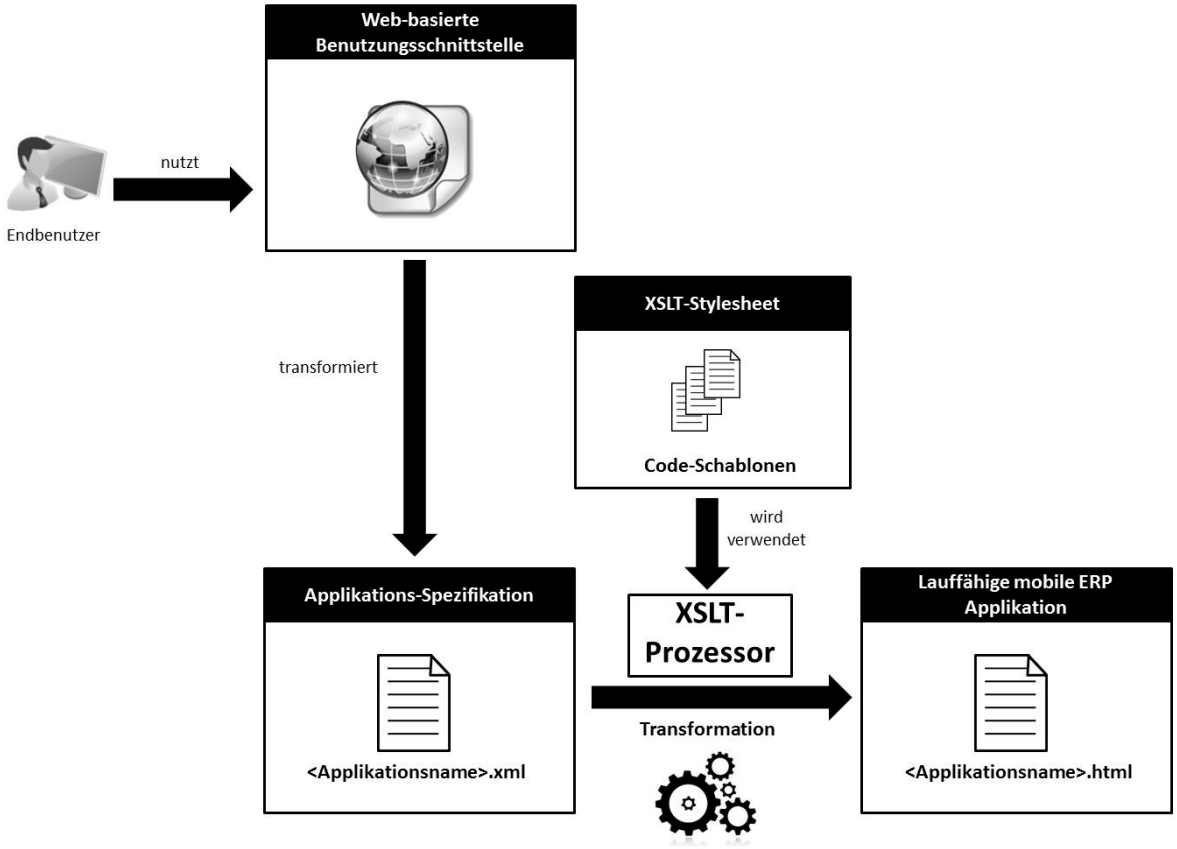

Abbildung 9-6: Spezifikations- und Transformationsvorgang
Quelle: Eigene Darstellung

9.2.3 Benutzungsschnittstelle

Um das Testen der prototypischen Werkzeugimplementierung zu ermöglichen, wurde eine webbasierte Benutzungsschnittstelle implementiert. Hierdurch ist es möglich, die Spezifikation einer mobilen ERP-Applikation komfortabler durzuführen, als dies über eine direkte Eingabe der XML-basierten Spezifikation über einen Texteditor möglich wäre. Im Folgenden wird der Aufbau der prototypisch umgesetzten Benutzungsschnittstelle beschrieben.

Abbildung 9-7 illustriert eine Bildschirmaufnahme der Einstiegsbildschirmmaske in die prototypisch implementierte Benutzungsschnittstelle. Im Reiter „Spezifikation“ kann eine neue mobile ERP-Applikation spezifiziert werden. Hierzu ist zunächst der Name der neuen Applikation zu spezifizieren. Anschließend können über die Schaltfläche „MBOT hinzufügen“ neue MBOTs hinzufügt werden. Anschließend können die anzuzeigenden MBOT-Attribute ausgewählt werden. Für jedes Attribut muss ein Datentyp spezifiziert werden. Dieser bestimmt beim Transformationsvorgang das zu nutzende Anzeige- bzw. Bedienelement und legt die Validierungsfunktion bei der Prüfung von Benutzereingaben fest. Zudem kann für jedes Attribut spezifiziert werden, ob dieses im Bedienkonzept „Sortierte Listen-Anzeige“ als Identifikator genutzt wird oder ob ein Attribut im Bedienkonzept „Formularba-

sierte MBO-Bearbeitung“ editierbar ist. Über die Schaltfläche „Speichere XML“ wird die durchgeführte Spezifikation als XML-Datei im Dateisystem des Apache HTTP Servers gespeichert. Anschließend kann der Transformationsprozess über die Schaltfläche „Generiere Applikation“ gestartet werden.

Neben diesen beiden Hauptfunktionalitäten ist es mit der prototypisch Implementierten Benutzungsschnittstelle auch möglich, bereits bestehende Applikations-Spezifikationen zu laden und anschließend zu bearbeiten. Hierzu wird der Reiter „Upload von XML“ verwendet. Zudem ist es möglich eine XML Spezifikation direkt als Text einzugeben. Hierzu wird der Reiter „Lade von XML“ verwendet. Beide Funktionalitäten eigneten sich für Testzwecke während der Implementierungsphase des Werkzeuges.

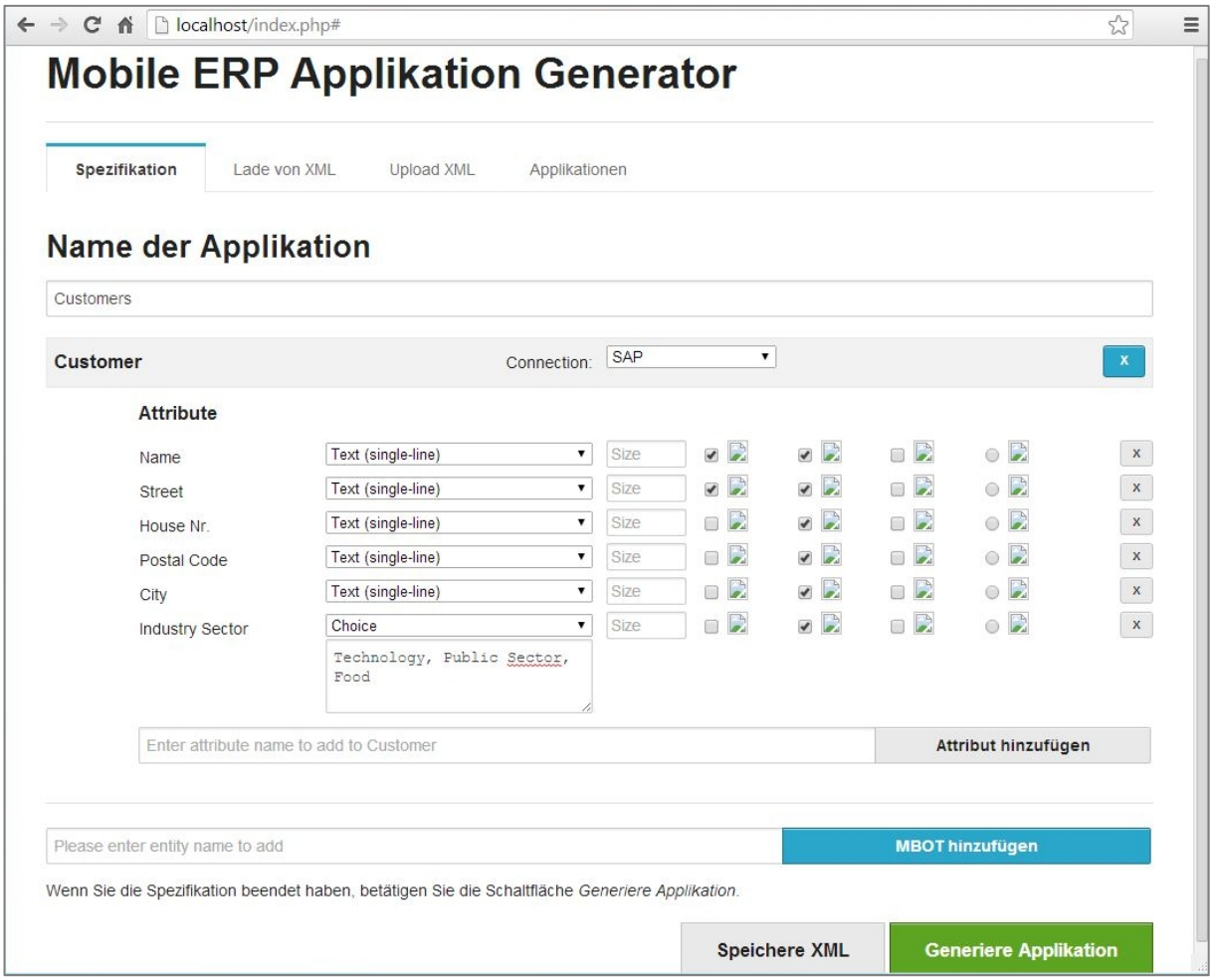

Abbildung 9-7: Einstiegsbildschirm in die webbasierte Benutzungsschnittstelle
Quelle: Eigene Darstellung

Die generierten mobilen ERP-Applikationen werden im Dateisystem des Apache HTTP Servers abgelegt und sind über den Reiter „Applikationen“ einsehbar. Abbildung 9-8 illustriert eine Bildschirmaufnahme des Werkzeuges mit dem geöffneten Reiter „Applikationen“.

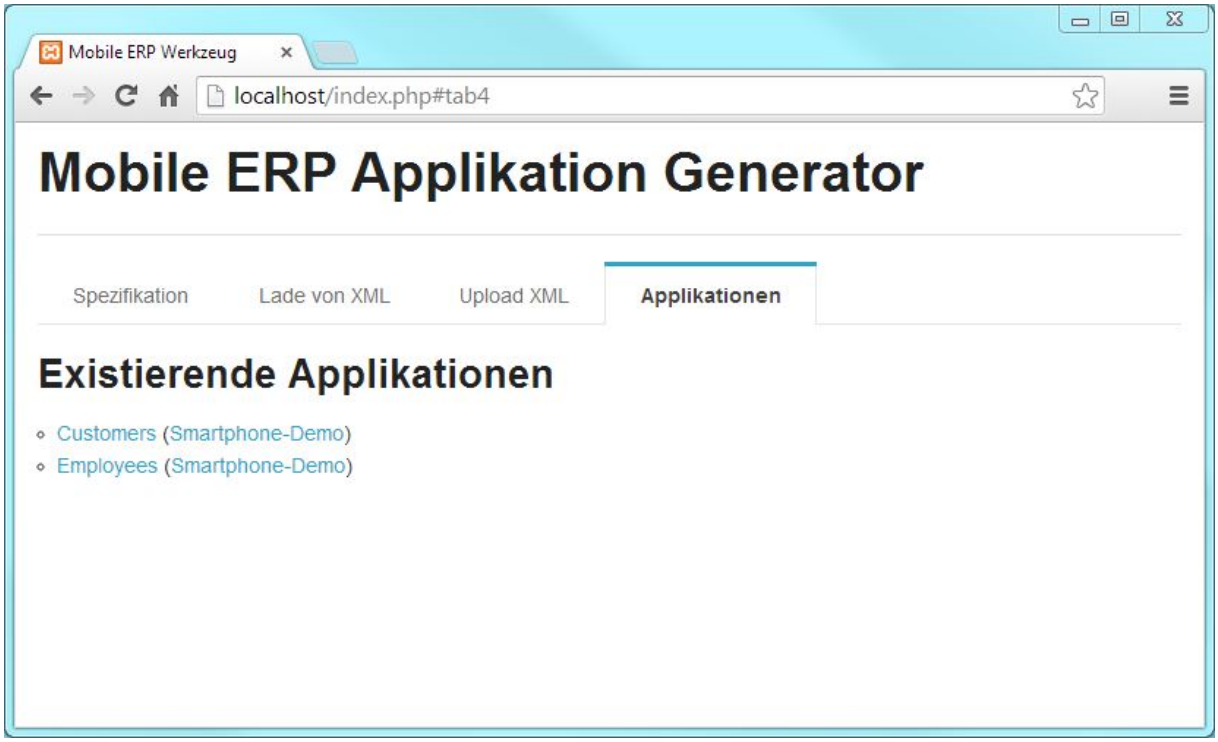

Abbildung 9-8: Applikationsliste in der webbasierten Benutzungsschnittstelle
Quelle: Eigene Darstellung

Jede existierende mobile ERP-Applikation ist über eine URL ansprechbar. In der angezeigten Liste können die einzelnen Applikationen direkt selektiert werden, woraufhin sich ein neuer Reiter im Webbrowser öffnet und die selektierte mobile ERP-Applikation anzeigt. Um den Test der Applikationen auch auf dem Desktop-PC realistischer erscheinen zu lassen, wurde eine zusätzliche Ansicht geschaffen, welche eine iPhone Grafik als Hintergrundbild verwendet und den Inhalt der mobilen ERP-Applikation im Displaybereich der iPhone Grafik anzeigt. Diese Ansicht ist über den Link „Smartphone-Demo“ hinter dem Applikationsnamen in der Applikationsliste aufrufbar. Abbildung 9-9 illustriert eine Bildschirmaufnahme im soeben beschriebenen Smartphone-Testmodus.

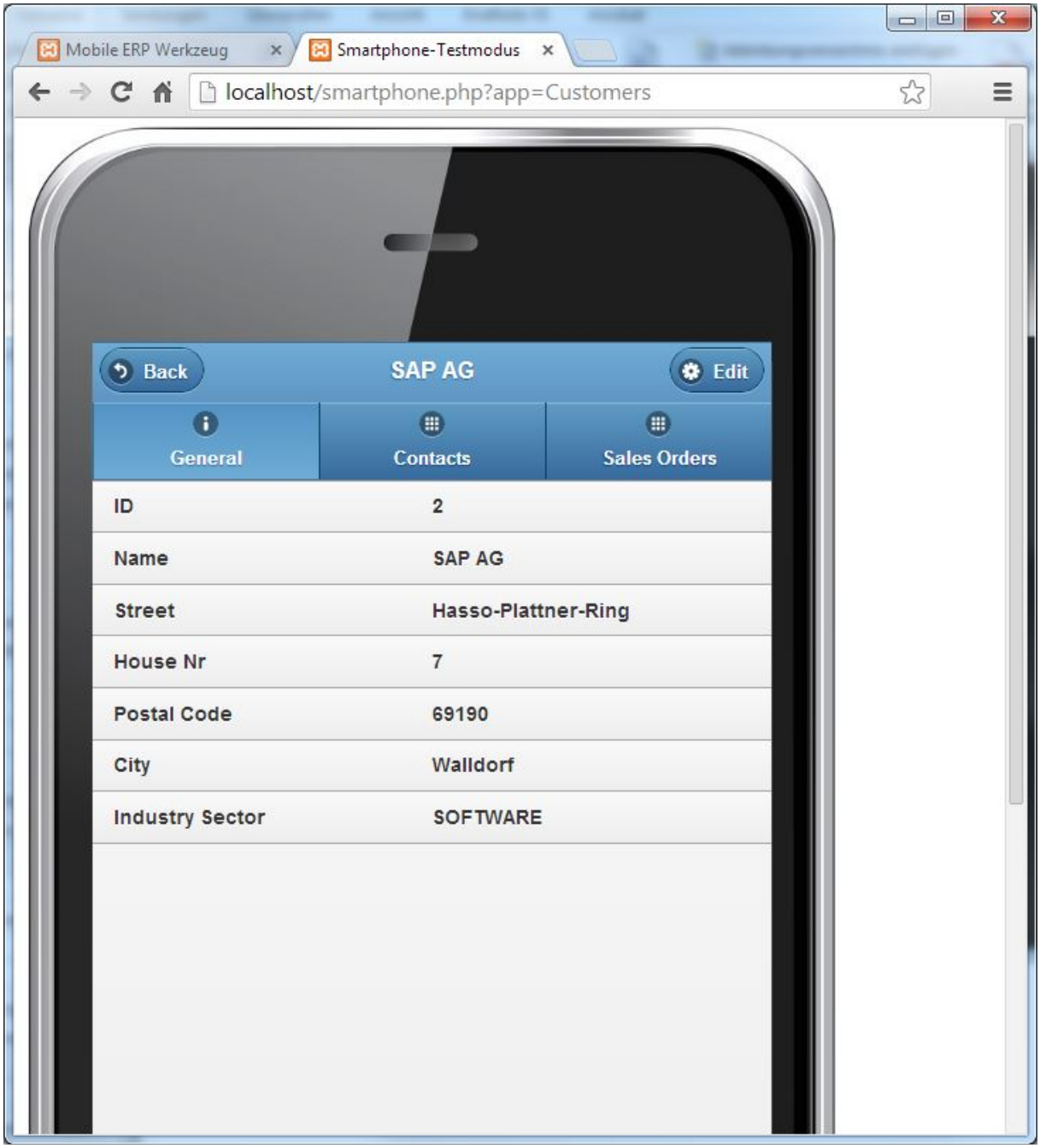

Abbildung 9-9: Benutzungsschnittstelle im Smartphone-Testmodus
Quelle: Eigene Darstellung

9.3 Evaluation

Mit Hilfe der beschriebenen prototypischen Implementierung ist es bereits möglich, einen Teil der Evaluationsaspekte (EA) aus Tabelle 7-2 zu prüfen. Deren Evaluation wird im Folgenden beschrieben. Dies betrifft die Evaluationsaspekte: EA1, EA2, EA3, EA10 und

EA12. Die Gesichtspunkte der restlichen Evaluationsaspekte werden in den folgenden beiden Designzyklen konzipiert und prototypisch umgesetzt und anschließend evaluiert.

9.3.1 Funktionaler Test zur Prüfung von EA1, EA2 und EA3

Um EA1, EA2 und EA3 zu prüfen wurde eine mobile ERP-Applikation zur Verwaltung eines MBOT „Customer“ über die prototypisch implementierte Benutzungsstelle spezifiziert. Diese mobile ERP-Applikation wurde gemäß des unter Anforderung 3 geforderten Funktionsumfanges wie folgt spezifiziert:

- Auflistung des einzigen, verfügbaren MBOT „Customer“
- Auflistung aller MBOs des selektierten MBOT „Customer“
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten „Customer“ MBO
- Erzeugung eines neuen „Customer“ MBO
- Änderung eines existierenden „Customer“ MBO
- Löschung eines existierenden „Customer“ MBO

Nach der Spezifikation der Applikation wurde die ausführbare Applikation generiert. Anschließend wurde Sie sowohl auf einem iPhone, als auch auf einem Android Smartphone getestet, um EA1 zu prüfen. Das verwendete iPhone Testgerät war wie folgt ausgestattet:

- Geräteversion: iPhone 5s
- Betriebssystemversion: iOS 7.0.6
- Webbrowser: Integrierter Webbrowser des iOS Betriebssystems (Safari Mobile)

Zur Evaluation der erstellten Applikation auf einem Android Gerät wurde der sogenannte Android Virtual Device (AVD) Manager eingesetzt. Mit Hilfe des AVD Managers können unterschiedliche Hardwareparameter und Betriebssystemversionen konfiguriert und emuliert werden können. Hierbei wurde folgende Konfiguration verwendet:

- Geräteversion: Nexus S 4.0, 4 Zoll Display
- Betriebssystem: Android 4.2.2
- Webbrowser: Integrierter Webbrowser des Android Betriebssystems

Abbildung 9-10 illustriert die Konfigurationsparameter des virtuellen Testgerätes im AVD Manager.

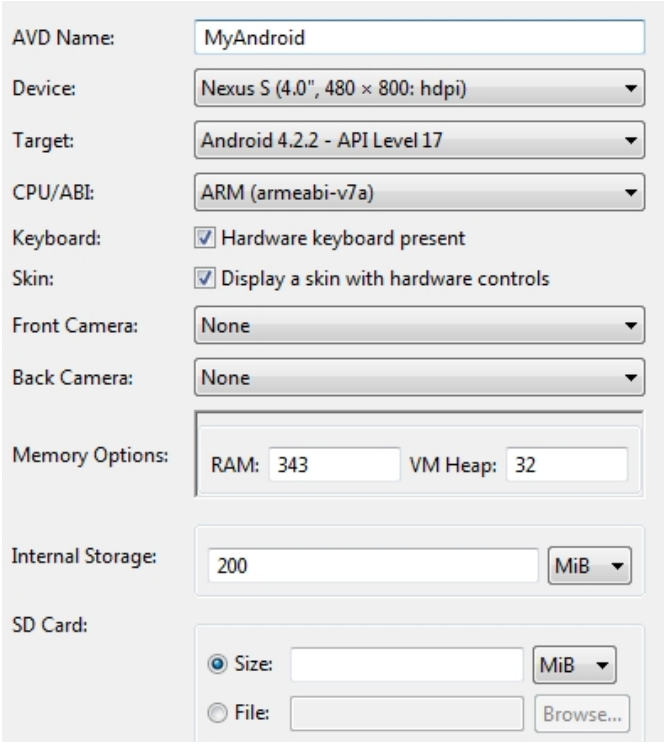

Abbildung 9-10: Konfiguration des virtuellen Android Testgerätes
Quelle: Eigene Darstellung

Abbildung 9-11 illustriert zwei beispielhafte Bildschirmaufnahmen aus dem beschriebenen funktionalen Test. Die rechte Bildschirmaufnahme zeigt die Funktionalität zum Auflisten der „Customer"-MBOs auf dem iPhone; die linke Bildschirmaufnahme zeigt die Funktionalität zur Änderung eines existierenden „Customer" MBO auf dem virtuellen Android Gerät.

Um EA2 zu prüfen, greift die erstellte Applikation auf die in Kapitel 2.1.5.2.1 genannten BAPIs über SOAP-basierte Webservices zu. Das Ergebnis des funktionalen Tests hat gezeigt, dass die geprüften Evaluationsaspekte (EA1, EA2 und EA3) durch die prototypische Implementierung erfüllt werden. Dabei ist jedoch anzumerken, dass die geforderten Bedienkonzepte in dieser prototypischen Implementierung noch nicht 1:1 umgesetzt wurden. Dies kann als ein Verbesserungsaspekt festgehalten werden. Ein weiterer Verbesserungsaspekt betrifft die Geschwindigkeit der Datenabfrage vom SAP-ERP-System. Die Nutzung der erstellten Testapplikation hat gezeigt, dass die Datenabfrage trotzt der geringen Anzahl an MBO-Datensätzen teilweise über 10 Sekunden dauert. Obgleich die Anforderungen keine Details zum Antwortverhalten der erstellten Applikationen beinhalten, kann dies die Akzeptanz der erstellten Applikationen einschränken. Aus diesem Grund wird die Verbesserung des Antwortverhaltens der Datenabfrage in den erstellten Applikationen ebenfalls als Verbesserungsaspekt aufgenommen werden.

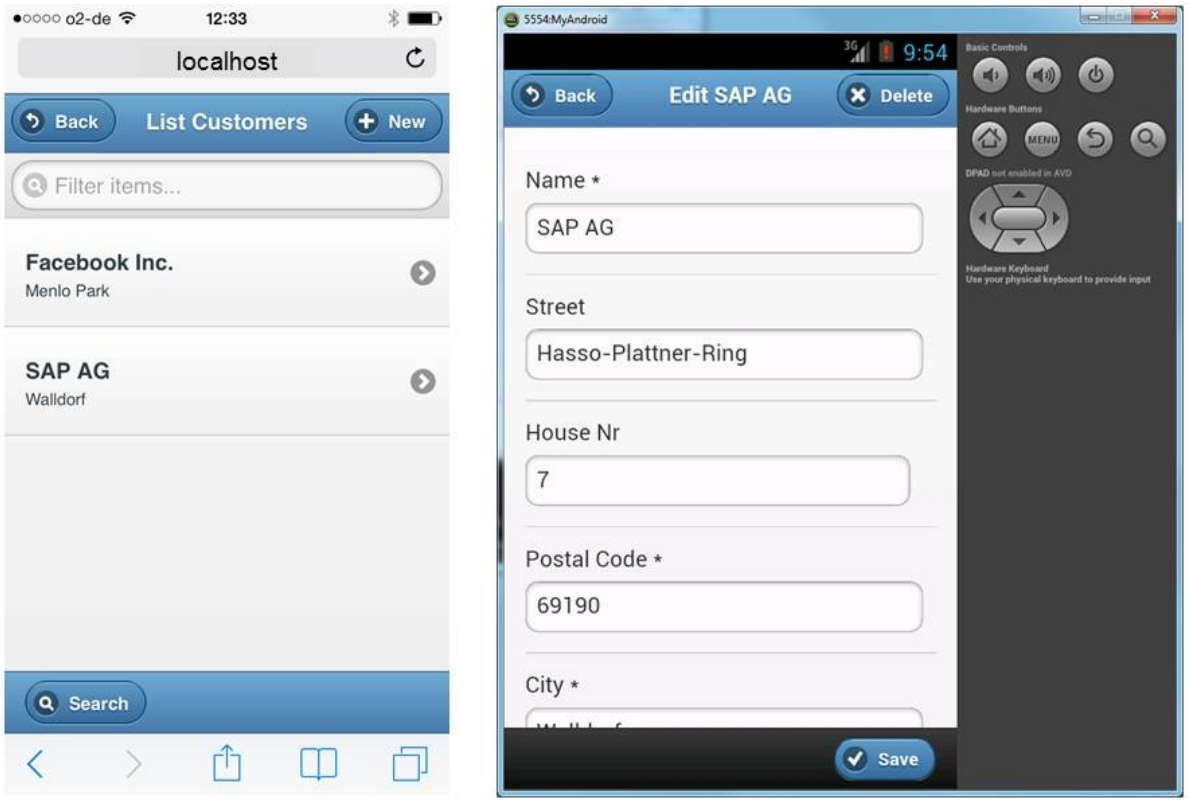

Abbildung 9-11: Funktionaler Test bzgl. EA1, EA2 und EA3 auf iPhone und virtuellem Android Smartphone

Quelle: Eigene Darstellung

9.3.2 Statische Analyse zur Prüfung von EA10

Die mit Evaluationsaspekt 10 verknüpfte Anforderung 10 verlangt die Verwendung wiederverwendbarer Codeschablonen. Beim lauffähigen Code für die entwickelten mobilen ERP-Applikationen handelt es sich im vorgestellten Architekturkonzept um ein HTML-Dokument, welches durch eine Transformation aus einer XML-basierten Applikationsspezifikation erzeugt wird. Zur Transformation wird ein XSLT-Prozessor verwendet. Die zugehörigen Transformationsregeln werden durch ein XSLT-Stylesheet bereitgestellt. Das XSLT-Stylesheet basiert auf einer Menge sogenannter XSLT-Templates. Diese definieren Codeschablonen, welche beim Auftreten eines bestimmten Musters (engl. pattern) in der Applikationsspezifikation angewendet werden. Die primären Codeschablonen des XSLT-Stylesheets sind die bereitgestellten Bedienkonzepte (siehe Kapitel 8.4.2.4). Hierdurch kann die Benutzungsschnittstelle unterschiedlicher mobiler ERP-Applikationen auf Basis der bereitgestellten XSLT-Templates realisiert werden. Aus diesem Grund wird EA10 durch den XSLT-basierten Transformationsvorgang auf Basis wiederverwendbarer XSLT-Templates sichergestellt.

9.3.3 Statische Analyse zur Prüfung von EA12

Die mit Evaluationsaspekt 12 verknüpfte Anforderung 12 fordert die prinzipielle Erweiterbarkeit des Entwicklungswerkzeuges, zur zukünftigen Unterstützung weiterer mobiler Betriebssysteme (neben iOS und Android). Im vorgestellten Architekturkonzept für das Entwicklungswerkzeug wird der Entwicklungsvorgang in zwei Schritte aufgeteilt. In einem ersten Schritt wird eine von einem mobilen Betriebssystem unabhängige Applikationsspezifikation erzeugt. In einem zweiten Schritt wird diese Applikationsspezifikation in eine HTML5-Datei, basierend auf dem jQuery Mobile Framework, überführt. Diese HTML5-Datei kann im Webbrowser eines iPhone oder Android Smartphones ausgeführt werden und stellt die Benutzungsschnittstelle der mobilen ERP-Applikation dar.

In Bezug auf die zukünftige Erweiterbarkeit auf andere mobile Betriebssysteme hat dieses Vorgehen zwei wesentliche Vorteile. Der erste Vorteil betrifft die Trennung von Spezifikation und ausführbarem Code der Applikationen. Um ein weiteres mobiles Betriebssystem zu unterstützen, müsste letztlich nur eine Implementierung eines entsprechenden Codegenerators bereitgestellt werden. Wenn es sich um ein textbasiertes Zielformat handelt, wäre sogar nur die Erweiterung des existierenden XSLT-Stylesheets notwendig. Durch den Einsatz anderer Transformationsimplementierungen wäre es zudem auch prinzipiell möglich, die notwendigen Dateien für native Applikationen zu generieren.

Der zweite Vorteil wird durch die Verwendung von jQuery Mobile erzielt. Bei jQuery Mobile handelt es sich um ein HTML5-Framework, welches die Unterstützung einer Vielzahl mobiler Betriebssystemen als Ziel verfolgt[59]. Falls zukünftig neue mobile Betriebssysteme entstehen oder existierende eine höhere Relevanz erhalten, wird dies voraussichtlich auch von den jQuery Mobile Entwicklern berücksichtigt und das Framework entsprechend erweitert. Insgesamt wird EA12 somit sowohl durch den zweistufigen Entwicklungsvorgang, als auch durch die Verwendung des etablierten HTML5-Frameworks jQuery Mobile sichergestellt.

9.4 Zusammenfassung und Diskussion

In diesem Kapitel wurde ein Architekturkonzept für einen Codegenerator und dessen Zusammenspiel mit anderen Komponenten vorgestellt. Der Codegenerator hat die Aufgabe, die auf Basis der domänenspezifischen Sprache spezifizierten Applikationen in lauffähige Applikationen zu überführen. Bei der Beschreibung des Architekturkonzeptes wurden unterschiedliche Designalternativen vorgestellt und diskutiert. Insbesondere das Zielformat der Benutzungsschnittstelle der mobilen ERP-Applikationen, der Aufruf der Programmierschnittstellen des SAP-ERP-Systems, das Transformationswerkzeug und die Architektur der mobilen ERP-Applikationen waren Schwerpunkt der Betrachtung.

[59] Eine Übersicht über die unterstützten mobilen Betriebssysteme ist unter folgendem Link zu finden: http://jquerymobile.com/gbs, zugegriffen am 03.02.2014

Um die Umsetzbarkeit des vorgestellten Architekturkonzeptes zu demonstrieren, wurde eine prototypische Implementierung durchgeführt und beschrieben. Bei der Beschreibung wurden u.a. die eingesetzten Technologien und Softwarekomponenten sowie der Aufbau der Testumgebung erläutert. Zudem wurde eine prototypische Implementierung einer Benutzungsschnittstelle für das Entwicklungswerkzeugs vorgestellt, welche die Evaluation einer Teilmenge der vorgestellten Evaluationsaspekte (siehe Kapitel 7.3) ermöglichte. Die bereitgestellte Benutzungsschnittstelle diente jedoch ausschließlich der komfortableren Durchführung der funktionalen Tests und erhebt keinen Anspruch auf Endbenutzertauglichkeit.

Durch die vorgestellte prototypische Implementierung war es möglich, die Evaluationsaspekte EA1, EA2 und EA3 durch einen funktionalen Test zu prüfen und nach dem Test positiv zu beantworten. Weiterhin wurden die Evaluationsaspekte EA10 und EA12 durch eine statische Analyse positiv beantwortet. Neben diesen positiven Evaluationsergebnissen hat der funktionale Test eine unzureichende Geschwindigkeit in der Datenkommunikation mit dem SAP-ERP gezeigt. Die vermutete Ursache ist die SOAP-basierte Datenkommunikation. Dieser Aspekt wird in einer verbesserten Implementierung berücksichtigt. Zudem wurden die vorgestellten Bedienkonzepte (siehe Kapitel 8.4.2.4) nicht 1:1 umgesetzt. Auch dies soll Gegenstand der verbesserten Implementierung sein.

10 Architektur und prototypische Implementierung des Entwicklungswerkzeuges

Nachdem die domänenspezifische Sprache und der zugehörige Codegenerator in den vorherigen Schritten konzipiert und prototypisch umgesetzt wurden, besteht der nächste Schritt in der Konzeption eines zugehörigen Entwicklungswerkzeuges. Fokus hierbei ist eine geeignete Benutzungsschnittstelle, welche eine Bedienung durch Endbenutzer ermöglicht. Schließlich müssen die vorgestellten Komponenten noch miteinander integriert werden, um ein kohärentes Entwicklungswerkzeug für die Endbenutzer-Entwicklung mobiler ERP-Applikationen vorstellen zu können.

Um eine geeignete Benutzungsschnittstelle zu konzipieren, wird zunächst ein low-fidelity Prototyp entwickelt und evaluiert. Die Entwicklung und Evaluation des low-fidelty Prototypen erfolgte gemeinsam mit der damaligen Studentin Andrea Maria Cuno (2013) als Teil ihrer Masterarbeit. Die resultierenden Evaluationsergebnisse werden anschließend verwendet, um eine verbesserte Benutzungsschnittstelle zu konzipieren. Diese wird anschließend in Form eines high-fidelity Prototypen implementiert und ebenfalls evaluiert. Die Entwicklung und Evaluation des high-fidelity Prototypen erfolgte gemeinsam mit dem damaligen Studenten Mateus Hubert Gawelek (2014) als Teil seiner Masterarbeit.

10.1 Konzeption der Benutzungsschnittstelle

Bei der angestrebten Spezifikation einer mobilen ERP-Applikation handelt es sich im Wesentlichen um Selektionsschritte. Hierbei können verfügbare MBOTs und die anzuzeigenden Attribute sowie die in der mobilen ERP-Applikation bereitgestellten Methoden selektiert werden. Für eine solche Selektion erscheint die Interaktionstechnik „formularbasierte Programmierung“ (siehe Kapitel 2.4.3.1.3) geeignet. In einer konkreten Umsetzung könnten die verfügbaren MBOTs als Auswahlfelder in einem Formular angeboten werden. Nach Selektion eines gewünschten MBOT könnten die Attribute des MBOT zur Auswahl angeboten werden.

Der Vorteil dieser Variante ist, dass der Endbenutzer zur Bedienung des Entwicklungswerkzeuges vorab keine Syntax erlernen muss. Damit kann der Trainingsaufwand für die Nutzung des Entwicklungswerkzeuges reduziert werden (siehe Anforderung 4). Zudem wäre hierdurch ein schrittweiser Entwicklungsprozess umsetzbar (siehe Anforderung 5). Der Entwicklungsprozess wäre in diesem Fall durch die Auswahl eines MBOT in einem ersten Schritt und durch die Auswahl der Attribute des MBOT und der unterstützten Methoden in einem folgenden Schritt charakterisiert. Diese Schritte würden sich anschließend für weitere in der Applikation verwendeten MBOTs wiederholen. Da sich der Entwicklungsprozess in diesem Fall im Wesentlichen auf Selektionsaktivitäten beschränkt wäre es möglich potentielle Fehlerquellen deutlich zu reduzieren und damit einen kurzen Entwicklungsvorgang zu erreichen (siehe Anforderung 7).

Eine alternative Umsetzungsvariante wäre der Einsatz der Interaktionstechnik „visuellen Programmierung“ (siehe Kapitel 2.4.3.1.1). In diesem Fall könnten die MBOT-Methoden (siehe Kapitel 8.4.2.3.3) durch visuelle Elemente repräsentiert werden. Hierbei würde sich eine ähnliche Umsetzungsvariante wie im SAP Mobile Workspace (siehe Kapitel 6.1.2) oder in der ASML Workbench (siehe Kapitel 8.2.1) anbieten. Die Entwicklungsaktivität bestünde in diesem Fall aus dem Drag & Drop benötigter MBOT-Methoden in ein Entwicklungsdiagramm sowie der Konfiguration bereitgestellter Parameter. Zudem könnten die einzelnen MBOT-Methoden durch Verbindungslinien verknüpft werden, um den Kontrollfluss der Applikation zu gestalten. Die visuellen Programmierung als mögliche Umsetzungsvariante für diese Arbeit ist schematisch in Abbildung 10-1 skizziert. Die dort beispielhafte skizzierte mobile ERP-Applikation umfasst die beiden MBOTs „Kunde“ und „Mitarbeiter“. Für beide MBOTs wurden die Methoden: 1) Auflistung aller MBOs und 2) Anzeige der Attribute und zugehörigen Werte eines ausgewählten MBO sowie 3) Änderung bzw. Bearbeitung der Attributwerte eines ausgewählten MBO spezifiziert.

Die beschriebene Umsetzungsvariante der Interaktionstechnik „visuelle Programmierung“ besitzt gegenüber der Interaktionstechnik „formularbasierte Programmierung“ den Vorteil, dass der Kontrollfluss der mobilen ERP-Applikation visuell dargestellt wird und somit für einen Endbenutzer möglicherweise leichter verständlich ist. Zudem besitzt die visuelle Programmierung mehr Freiheitsgrade sowohl hinsichtlich der Kombinationsmöglichkeiten der Applikationsbausteine, als auch hinsichtlich des Entwicklungsprozesses. Der Vorteil des höheren Freiheitsgrades geht jedoch einher mit einer höheren Fehlerwahrscheinlichkeit und einer höheren Einstiegshürde. In Bezug auf den Freiheitsgrad bei der Applikationsspezifikati-

on hat sich gezeigt, dass zweckmäßige Kombinationsmöglichkeiten eingeschränkt sind. Dies zeigt u.a. das VIA-Interaktionsmuster, welches die Bearbeitungsreihenfolge „Selektion → Präsentation → Manipulation“ empfiehlt (siehe Kapitel 3.3.2.1). Am skizzierten Beispiel in Abbildung 10-1 bedeutet dies, dass der Baustein „Kunden-Details“ als zweckmäßigen Vorgänger den Baustein „Kundenliste“ besitzt. Basierend auf dieser Erkenntnis kann der Aspekt „Freiheitsgrad bei der Entwicklung“ als weniger relevant eingestuft werden. Demzufolge erscheint die „formularbasierte Programmierung“ als geeignete Interaktionstechnik für die Benutzungsschnittstelle des Entwicklungswerkzeuges.

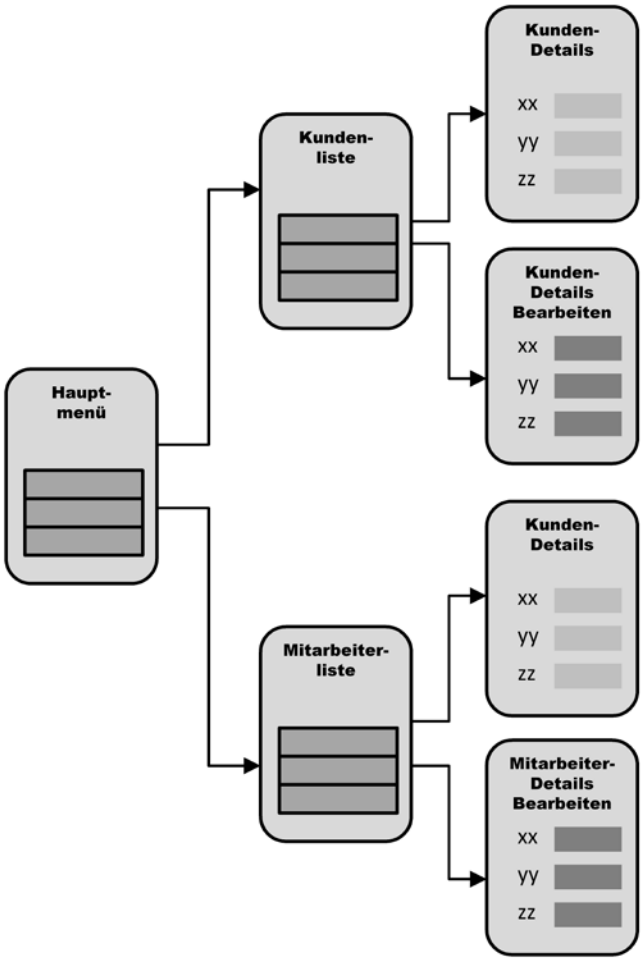

Abbildung 10-1: Visuelle Programmierung mobiler ERP-Applikationen

Um eine erste Evaluation der formularbasierten Interaktionstechnik zu erhalten wurde ein low-fidelity Prototyp auf Basis von Microsoft Powerpoint entworfen. Dieser repräsentiert einen ersten Entwurf der grafischen Benutzungsoberfläche. Im Folgenden werden exemplarisch zwei Bildschirmmasken[60] des Prototyps skizziert.

[60] Die einzelnen Bildschirmmasken werden im Folgenden als Formulare bezeichnet.

Abbildung 10-2 illustriert den Entwurf der Benutzungsoberfläche zum Spezifizieren des Bedienkonzeptes „Sortierte Listen-Anzeige“. In dieser Formularseite kann der Titel der Bildschirmseite eingegeben werden sowie das fokussierte MBOT ausgewählt werden. Zudem können die angezeigten Attribute der gelisteten MBOs selektiert werden. Im abgebildeten Entwurf ist dabei die Selektion von bis zu vier Attributen möglich, wobei für die Sortierung in der Liste automatisch das erste Attribut (Attribut 1) verwendet wird. Um den Endbenutzer zu unterstützten, enthält das Formular einen erklärenden Text bzgl. der durchzuführenden Schritte sowie deren Einschränkungen. Neben diesem Text ist für jedes Formular eine ausführlichere textuelle Beschreibung über die Betätigung der Schaltfläche „Hilfe“ abrufbar. Zusätzlich ist eine generische Abbildung des soeben fokussierten Bedienkonzeptes auf der Formularseite enthalten. Diese soll den Endbenutzer beim Verständnis der durchzuführenden Schritte unterstützen.

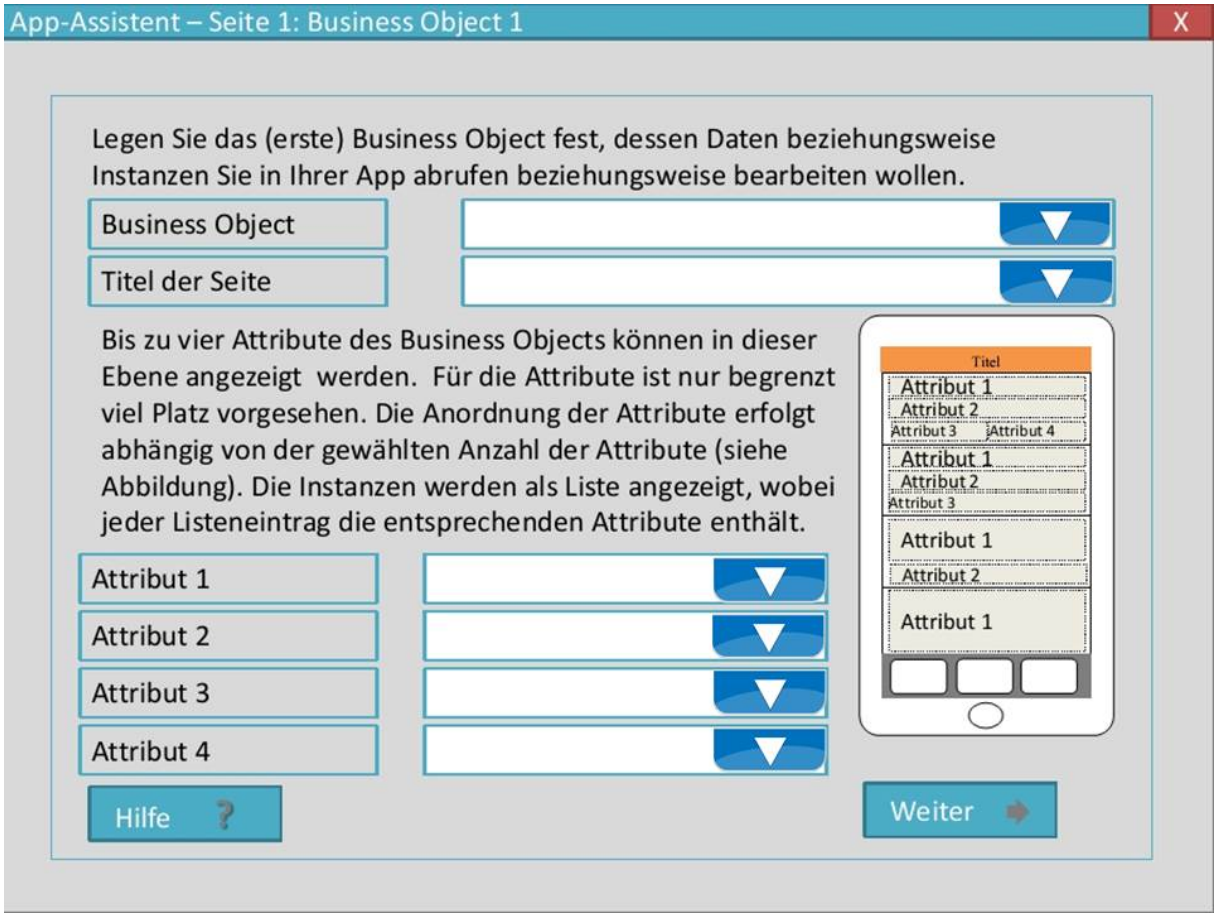

Abbildung 10-2: Entwurf der Benutzungsschnittstelle zur Spezifikation des Bedienkonzeptes "Sortierte Listen-Anzeige"

Quelle: Eigene Darstellung in Anlehnung an (Cuno 2013, 20)

Abbildung 10-3 illustriert den Entwurf der Benutzungsschnittstelle zur Spezifikation des Bedienkonzeptes „Formularbasierte MBO-Anzeige“. In diesem Formular können die anzuzeigenden Attribute des selektierten MBO ausgewählt werden. Für jedes Attribut kann eine optionale Beschreibung eingegeben werden. Diese legt den Text vor dem Attributwert fest.

Wird keine Beschreibung angegeben, so wird der Name des jeweiligen Attributes verwendet. Welches Anzeigeelement für die Darstellung des Attributwertes verwendet wird, kann durch den Endbenutzer nicht bestimmt werden. Es wird stattdessen direkt aus dem Datentyp des Attributes abgeleitet. Diese Umsetzungsvariante wurde aus Anforderung 9 abgeleitet, welche einen geringen Gestaltungsspielraum bei der Entwicklung der Benutzungsschnittstelle fordert. Zusätzlich enthält das Formular einen beschreibenden Text, eine Schaltfläche zur Navigation

Abbildung 10-3: Entwurf der Benutzungsschnittstelle zur Spezifikation des Bedienkonzeptes "Formular-basierte MBO-Anzeige"

Quelle: Eigene Darstellung in Anlehung an (Cuno 2013, 20)

zu einem ausführlicheren Hilfetext sowie ein generische Abbildung des Bedienkonzeptes.

10.2 Zwischen-Evaluation

Mit dem Ziel den ersten Entwurf der Benutzungsschnittstelle zu testen, wurde eine Zwischen-Evaluation durchgeführt. Hierbei wurden die Evaluationsaspekte EA4 und EA9 geprüft. Zudem sollten Gestaltungsideen für eine erweiterte- und optimierte Benutzungsschnittstelle gewonnen werden.

10.2.1 Nutzerinterviews zur Zwischen-Evaluation von EA4

Evaluationsaspekt 4 erfordert die Prüfung des Trainingsaufwandes für die Nutzung des Entwicklungswerkzeuges. Da die Entwicklung lauffähiger mobiler ERP-Applikationen mit dem low-fidelity Prototypen nicht möglich ist, wurde als Vorabevaluation ein semi-strukturiertes Interview mit potenziellen Endbenutzern durchgeführt. Ziel dabei war es, ein erstes Feedback für die Benutzungsschnittstelle zu erhalten und die Verständlichkeit der Benutzungsschnittstelle ohne vorherige Einweisung zu prüfen. Zudem sollten Gestaltungsvorschläge für die Erweiterung und Optimierung der Benutzungsschnittstelle identifiziert werden.

10.2.1.1 Aufbau des Interviews

Das Interview gliedert sich insgesamt die drei folgenden Phasen:

(1) Ideenfindung zu einer mobilen ERP-Applikation

(2) Vorstellung und Bewertung der einzelnen Formulare des low-fidelity Prototypen

(3) Fragen zum low-fidelity Prototypen

Phase (1) diente zur Vorbereitung auf das Interview. Die Probanden sollten sich vorab zwei zu entwickelnde mobile ERP-Applikation überlegen und die einzelnen Bildschirmmasken der Applikation auf einem Blatt Papier skizzieren. Der Anwendungsfälle für die mobile ERP-Applikationen sollte sich aus dem Arbeitsumfeld der Probanden ergeben. Eine Einschränkung hinsichtlich des eigenen Bedarfes oder eines Fremdbedarfes wurde nicht getroffen.

In Phase (2) wurden den Probanden die einzelnen Formulare des low-fidelity Prototypen gezeigt. Die Aufgabe der Probanden war die Nutzung der angebotenen Formulare im Hinblick auf die Erstellung ihrer in Phase (1) entwickelten Applikationsidee zu bewerten. Die Kommentare der Probanden wurden dabei vom Interviewleiter protokolliert.

In Phase (3) wurden den Probanden übergreifende Fragen zum vorgestellten low-fidelity Prototypen gestellt. Zur Strukturierung der Befragung wurde hierbei ein Fragebogen verwendet, welcher in Anhang C zu finden ist. Der Fragebogen umfasst Fragen zur Bedienung, zur Hilfestellung sowie zum Funktionsumfang des Werkzeuges.

Zur Teilnahme am beschriebenen Experiment konnte 11 potenzielle Endbenutzer als Probanden gewonnen werden. Die Probanden hatten 1 – 9 Jahre Erfahrung im ERP-Umfeld. Zudem waren alle 11 Probanden Besitzer eines Smartphones und hatten mehrere Jahre Erfahrung in der Bedienung von Smartphones.

10.2.1.2 Applikationsideen

Der größte Teil der von den Probanden beschriebenen Applikationsideen hatte als Zielgruppe Mitarbeiter oder Vorgesetzte eines Unternehmens. Nur zwei Applikationsideen hatten Kun-

den als Zielgruppe. Eine Applikationsidee hatte das Monitoring von IT-Systemen als Anwendungsfall und damit IT-Administratoren als Zielgruppe.

Die Applikationsideen für Mitarbeiter oder Vorgesetzte eines Unternehmens können dabei den folgenden Bereichen zugeordnet werden:

- Bearbeitung von Anfragen, insbesondere Genehmigungen
- Anzeige von Geschäftsdaten
- Erfassung oder Bearbeitung von Geschäftsdaten

Die am häufigsten skizzierten Funktionalitäten auf den einzelnen Bildschirmmasken waren: 1) Übersichtsseiten als Einstiegspunkt in die Navigation der Applikation, 2) Detailseiten einzelner MBOs, 3) die Bearbeitung von MBOs und 4) die Erstellung von MBOs. Neben diesen häufig genannten Funktionalitäten der Applikationen wurden u.a. auch vereinzelt folgende Funktionalitäten skizziert:

- Grafische Visualisierung von Daten
- Terminplanung
- Routenplanung
- Speicherung von Daten auf dem Smartphone

10.2.1.3 Interviewergebnisse und Auswertung des Fragebogens

In Bezug auf die *Bedienung* des Werkzeuges beurteilten die meisten Probanden die gewählte formularbasierte Interaktionsform des Werkzeuges als intuitiv und einfach erlernbar. Einige Teilnehmer stellten die Ähnlichkeit zu Assistenten in ihnen bekannten Softwarewerkzeugen fest, weshalb ihnen die gewählte Interaktionsform vertraut war. Es hat sich gezeigt, dass die Probanden das Ziel der einzelnen Formulare schnell erkannt haben und fähig waren, die benötigten Eingaben zu tätigen. Der Großteil der Probanden sah sich in der Lage eine mobile ERP-Applikation mit dem Werkzeug zu erstellen. Als Verbesserungsvorschläge wurden u.a. folgende Aspekte genannt:

- Höhere Flexibilität des Werkzeuges
- Höherer Funktionsumfang
- Größere Freiheiten bei der Layoutgestaltung

Die *Hilfefunktion* wurde nur von wenigen Probanden als hilfreich bewertet. Es hat sich gezeigt, dass die Probanden eine Trial & Error Vorgehensweise gegenüber dem Lesen von Hilfetexten bevorzugen. Einige Probanden nannten integrierte Videoanleitungen als sinnvolle Erweiterung der Hilfefunktion.

Die in Kapitel 10.2.1.2 aufgelisteten Funktionen der Applikationsideen zeigen bereits, dass ein Großteil der gewünschten Funktionalitäten mit dem Werkzeug realisiert werden kann.

Neben diesen häufig genannten Funktionalitäten wurden vereinzelt weitere Funktionalitäten gewünscht. Dabei wurden u.a. die folgenden Funktionalitäten genannt:

- Berechnungsfunktionalitäten
- Integration von Gerätehardware (z.B. GPS-Sensor oder Kamera)
- Nutzung von Daten, welche nicht als Attribute von MBOs verfügbar sind
- Auswahl unterschiedlicher Layoutvorlagen

10.2.2 Statische Analyse zur Prüfung von EA9

Der Evaluationsaspekt 9 erfordert eine Prüfung des Gestaltungsspielraumes bei der Entwicklung der Benutzungsschnittstelle der mobilen ERP-Applikationen. Im vorgestellten low-fidelity Prototypen können nur wenige Parameterwerte über ein Textfeld festgelegt werden. Dazu zählen beispielsweise der Name der Applikation oder individuelle Beschriftungen von Attributen. Die restlichen Entwicklungsaktivitäten beschränken sich auf die Selektion von Werten aus einem Auswahlfeld und der Reihenfolge der Auswahl. Über diese Möglichkeiten hinaus bestehen keine weiteren Gestaltungsfreiheiten. Eine individuelle Positionierung von Anzeige- oder Bedienelementen ist somit nicht möglich. Zusammenfassend lässt sich somit feststellen, dass EA9 durch die gewählte formularbasierte Interaktionstechnik gewährleistet werden kann.

10.2.3 Schlussfolgerungen der Zwischen-Evaluation

In Kapitel 10.1 wurde ein erster Entwurf einer Benutzungsschnittstelle für das gewünschte Entwicklungswerkzeug entwickelt. Aufgrund des gewünschten, geringen Gestaltungsspielraumes bei der Entwicklung der Benutzungsschnittstelle (siehe Anforderung 9) und der erwarteten, intuitiven Bedienbarkeit wurde eine formularbasierte Interaktionstechnik gewählt. Bei der gewählten Umsetzungsvariante beschränkt sich die Entwicklung im Wesentlichen auf die Auswahl von Werten aus verfügbaren Auswahlfeldern.

Die Evaluation mit potenziellen Endbenutzern hat gezeigt, dass der Fokus auf Produktivitätsapplikationen (siehe Anforderung 3) und die dabei verwendeten Bedienkonzepte (siehe Kapitel 8.4.2.4) einen Großteil der gewünschten Funktionalitäten und Bildschirmmasken gewünschter mobiler ERP-Applikationen abdecken. Daneben existieren jedoch noch weitere Anforderungen bzgl. gewünschter Funktionalitäten und Bildschirmmasken, welche durch das Werkzeug aktuell nicht abgedeckt werden. Beispiele sind die Integration mit Gerätehardware und Berechnungsfunktionalitäten.

In Bezug auf die gewählte Interaktionstechnik hat die Evaluation gezeigt, dass Endbenutzer diese zügig verstehen und nutzen können. Dies verstärkt die Erwartung, dass für die Nutzung des Werkzeuges kein Trainingsaufwand notwendig ist (siehe Anforderung 4). Dies kann jedoch erst bei Vorliegen einer prototypischen Implementierung des Werkzeuges in einem kontrollierten Experiment evaluiert werden. Einige Endbenutzer empfinden den geringen

Gestaltungsspielraum jedoch als ungeeignet. Sie fordern stattdessen Werkzeuge, welche eine feingranulare Positionierung von Anzeige- und Bedienelementen über Drag & Drop erlauben. Die Hilfefunktion wurde insgesamt kaum verwendet. Stattdessen hat sich gezeigt, dass Endbenutzer eine Trial & Error Vorgehensweise dem Lesen von Hilfetexten vorziehen. Daher sollte die Benutzungsschnittstelle des Werkzeuges ein „probierendes“ Vorgehen mit hilfreichen Hinweisen im Fehlerfall umsetzen.

10.3 Erweiterung und Optimierung der Benutzungsschnittstelle

Die Evaluation der konzipierten Benutzungsschnittstelle im vorherigen Kapitel hat gezeigt, dass die formularbasierte Interaktionstechnik von potenziellen Endbenutzern verstanden wird. Zudem können hiermit die gewünschten Entwicklungsaktivitäten abgedeckt werden. Um Anforderung 11 zu erfüllen, muss die Benutzungsschnittstelle jedoch auch auf einem Smartphone sinnvoll einsetzbar sein. Aufgrund der kleineren Displaygröße ist die in Kapitel 10.1 vorgestellte Benutzungsschnittstelle nicht geeignet.

Die gewählte formularbasierte Interaktionstechnik erscheint jedoch weiterhin zweckdienlich. Gegenüber der visuellen oder textuellen Programmierung sind bei einer formularbasierten Programmierung verhältnismäßig wenige Eingaben erforderlich. Da Drag & Drop Interaktionen und Texteingaben auf berührungsempfindlichen Bildschirmen (engl. touch screens) aufwändig und fehleranfällig sind (Dahm 2006, 205 f.), reduziert die selektionsbasierte Vorgehensweise bei der formularbasierten Programmierung die Texteingaben und Interaktionen mit dem berührungsempfindlichen Bildschirm. Daher sollte die vorgestellte Benutzungsschnittstelle des Entwicklungswerkzeuges auf die für ein Smartphone geeignete Bildschirmgröße transformiert werden, um Anforderung 11 zu erfüllen.

Um die in Kapitel 10.1 vorgestellte Benutzungsschnittstelle auf einen Smartphone-Bildschirm zu transformieren ist es notwendig, die auf den Formularen gezeigten Inhalte zu reduzieren. Da die Hilfefunktion in der Zwischen-Evaluation kaum beachtet wurde, wird diese auf der Smartphone-Benutzungsschnittstelle nicht angeboten. Da die Vorschau-Grafik als hilfreich eingestuft wurde, wird diese weiterhin Bestandteil der einzelnen Formulare sein. Hingegen wird die textbasierte Hilfestellung reduziert. Zudem soll eine Trial & Error Vorgehensweise besser unterstützt werden. Hierzu werden die getätigten Eingaben validiert; im Fehlerfall wird der Übergang zum nächsten Formular verhindert und ein Fehlerhinweis angezeigt. Um den Fehlerhinweis sofort erkennen zu können, wird dieser in roter Schrift angezeigt. Um den Fehlerhinweis zuordnen zu können, wird dieser in unmittelbarer Nähe vom jeweiligen Eingabefeld anzeigt. Ein weiterer Anpassungsbedarf betrifft die Überführung der Anzeige- und Bedienelemente der Desktop-Benutzungsschnittstelle auf geeignete Anzeige- und Bedienelemente einer Smartphone-Benutzungsschnittstelle.

Durch die Nutzung eines vergleichsweise kleinen Smartphonebildschirmes müssen die Inhalte der Applikationsverwaltung und –entwicklung auf mehrere Bildschirmmasken verteilt werden. Hierzu wird die Navigationsstruktur in drei Navigationsblöcke (NB) aufgeteilt. NB1 dient als Einstiegspunkt in das Entwicklungswerkzeug. Hier kann sich der Endbenutzer mit

einem existierenden Benutzerkonto anmelden oder ein neues Benutzerkonto erstellen. Nach erfolgreicher Anmeldung gelangt der Endbenutzer in die Applikationsverwaltung (NB2). Hier sieht der Endbenutzer seine bereits entwickelten oder zugewiesenen Applikationen. Diese kann er durch eine Selektion nutzen. Zudem hat er die Möglichkeit die angezeigten Applikationen zu bearbeiten oder zu löschen. Weiterhin hat der Endbenutzer die Möglichkeit eine neue Applikation zu entwickeln. Hierbei gelangt er zum Entwicklungsassistenten (NB3), welcher ihn Schritt-für-Schritt durch den Entwicklungsvorgang leitet. Am Ende des Entwicklungsvorganges hat der Endbenutzer zudem die Möglichkeit seine Applikation an andere, existierende Benutzerkonten zu verteilen. Abbildung 10-4 skizziert die beschriebenen Navigationsblöcke des Entwicklungswerkzeuges.

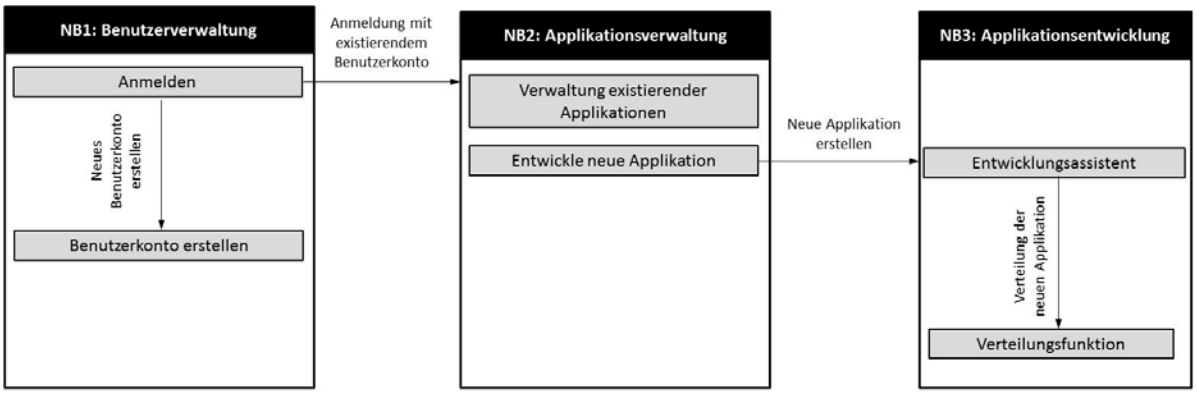

Abbildung 10-4: Navigationsblöcke der Benutzungsschnittstelle des Entwicklungswerkzeuges
Quelle: Eigene Darstellung

Zur Umsetzung von Anforderung 5, welche eine Schritt-für-Schritt Entwicklung fordert, wir die Benutzungsschnittstelle des Entwicklungsassistenten in mehrere Bildschirmmasken (BM) aufgeteilt. Hierbei sollte eine Vorwärts- und Rückwärtsnavigation zwischen den einzelnen Bildschirmmasken möglich sein. In der ersten Bildschirmmaske (BM1) wird die Eingabe des Applikationsnamens gefordert. Bevor eine Navigation zur nächsten Bildschirmmaske möglich ist, muss die Gültigkeit des eingegebenen Namens geprüft werden. Gültige Namen enthalten keine Ziffern und Sonderzeichen. In der folgenden Bildschirmmaske (BM2) werden die verfügbaren Layouts angezeigt. Nach Auswahl eines Layouts werden die verfügbaren MBOTs aufgelistet (BM3). Der Endbenutzer hat nun die Möglichkeit eines oder mehrere MBOTs zu selektieren. Nach der Auswahl navigiert der Entwicklungsassistent zur Spezifikation des ersten selektieren MBOT. Hierbei wird zunächst das Bedienkonzept „formularbasierte Listen-Anzeige“ spezifiziert (siehe Kapitel 8.4.2.4.1) (BM4). Hierbei wählt der Endbenutzer ein identifizierendes Attribut aus, welches zur Identifikation und Sortierung der MBOs in der listenartigen Darstellung verwendet wird. Anschließend wird das Bedienkonzept „formularbasierte MBO-Anzeige“ (siehe Kapitel 8.4.2.4.2) spezifiziert (BM5). Dabei werden die in der Detailansicht eines selektierten MBO anzuzeigenden Attribute ausgewählt. Anschließend können weitere gewünschte MBOT-Methoden gewählt werden (BM6).

Sollten mehrere MBOTs selektiert worden sein, so wird die selektierte Liste sukzessive abgearbeitet und jeweils zur Spezifikation des nächsten MBOTs zur Bildschirmmaske BM4

navigiert. Sobald die Liste abgearbeitet ist, hat der Endbenutzer die Möglichkeit die spezifizierte Applikation zu generieren (BM7) und anschließend an andere Endbenutzer zu verteilen (BM8). Während des gesamten Entwicklungsvorganges werden die spezifizierten Werte des Endbenutzers im Hauptspeicher abgelegt. Sobald der Endbenutzer die Generierungsfunktion betätigt, werden die spezifizierten Werte in die XML-basierte Applikationsspezifikation überführt und anschließend der Transformationsvorgang zu einer HTML5-Applikation gestartet. Abbildung 10-5 illustriert die erläuterte Navigationsstruktur bei der Applikationsentwicklung.

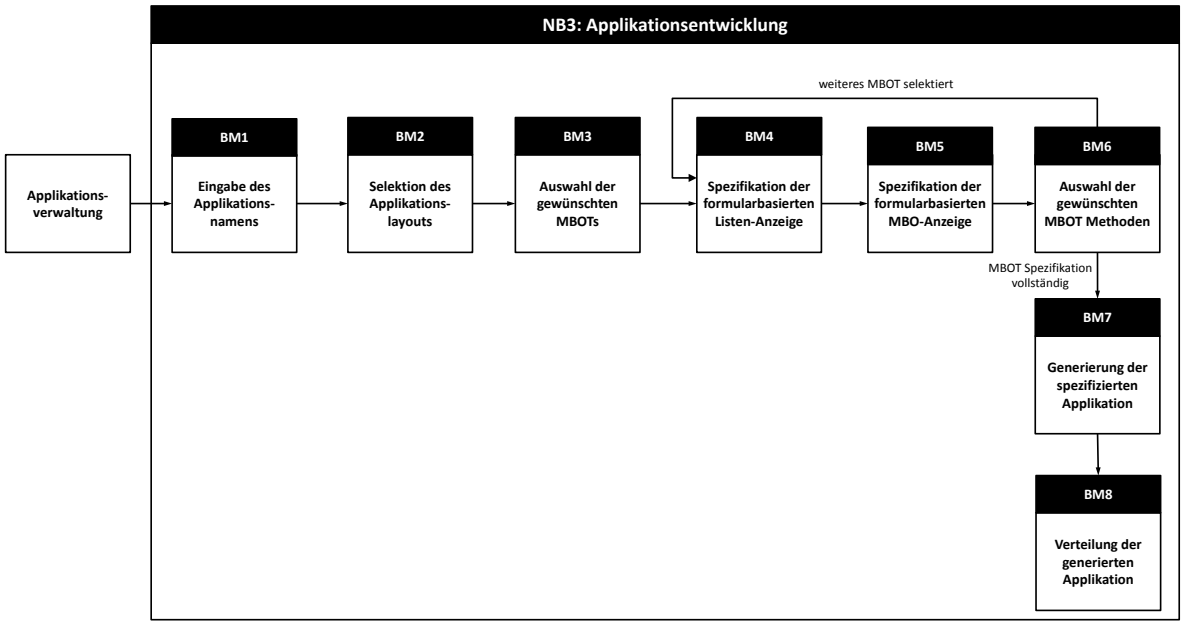

Abbildung 10-5: Navigationsstruktur der Applikationsentwicklung
Quelle: Eigene Darstellung

Im Folgenden wird der überarbeitete Gestaltungsvorschlag für die Benutzungsschnittstelle des Entwicklungswerkzeuges anhand von zwei beispielhaften Formularen erläutert. Beide Mock-Ups wurden mit dem Werkzeug Fluid UI[61] erstellt. Abbildung 10-6 illustriert das Einstiegsformular bei der Erstellung einer neuen mobilen ERP-Applikation. Der einzige, festzulegenden Wert in diesem Formular ist der Name der neuen mobilen ERP-Applikation Dieser wird später zur Identifikation der neuen Applikation verwendet. Für den Applikationsnamen sind keine Zahlen, Sonderzeichen und Leerzeichen zulässig. In Abbildung 10-6 wird ein Fehlerfall veranschaulicht. In diesem Fall erscheint ein roter Schriftzug mit einem entsprechenden Fehlerhinweis unter dem Textfeld. Durch zwei Schaltflächen im unteren Bildschirmbereich kann der Endbenutzer im Entwicklungsprozess vorwärts und rückwärts navigieren. In der Vorzeigegrafik wird die Verwendung des Applikationsnamens als Verständnishilfe grafische veranschaulicht.

[61] https://www.fluidui.com, zugegriffen am 20.11.2013

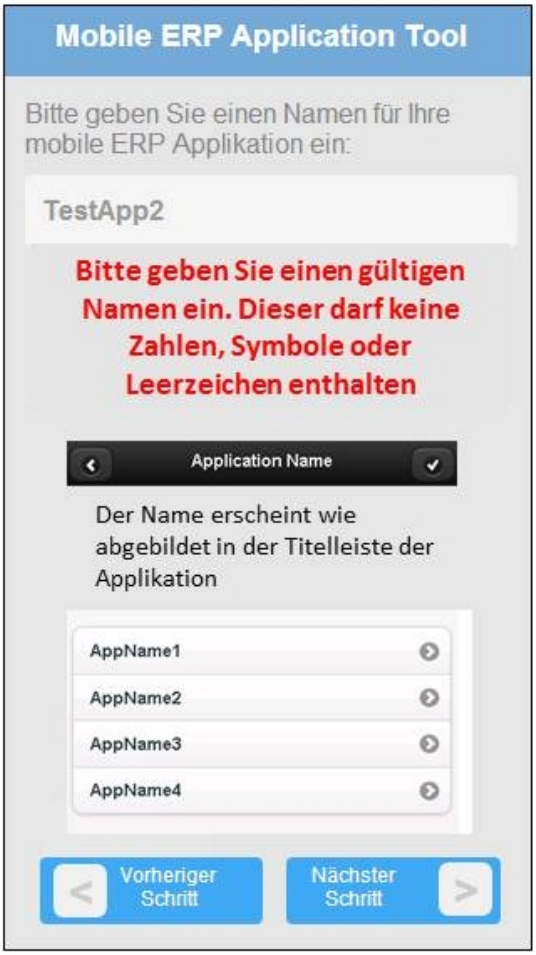

Abbildung 10-6: Entwurf der Bildschirmmaske BM1
Quelle: Eigene Darstellung

Abbildung 10-7 illustriert das Formular zur Konfiguration des Bedienkonzeptes „formularbasierte MBO-Anzeige“, also der Detailansicht eines selektierten MBO. Wie im Einstiegsformular werden auch in diesem Formular Schaltflächen zur Vorwärts- und Rückwärtsnavigation im Entwicklungsprozess bereitgestellt. Im illustrierten Beispiel wird dies für ein MBO „Kunde“ durchgeführt. Hierbei können anzuzeigende Attribute des MBO durch ein Aktivieren bzw. Deaktivieren des Kontrollbox vor dem zugehörigen Attribut in die Anzeige aufgenommen werden oder ausgeblendet werden.

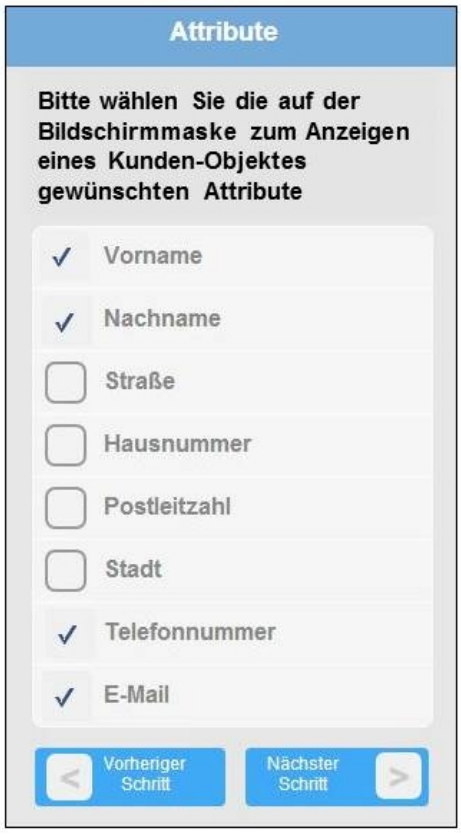

Abbildung 10-7: Entwurf der Bildschirmmaske BM5
Quelle: Eigene Darstellung

Zur Umsetzung der Benutzungsschnittstelle des Entwicklungswerkzeuges sind ähnlich wie bei der Umsetzung Benutzungsschnittstelle für die mobilen ERP-Applikationen, prinzipiell die vier Implementierungsvarianten: native-, Web-, Hybrid- oder Container-Applikationen möglich. Analog zu den mobilen ERP-Applikationen selbst wird eine Lauffähigkeit des Entwicklungswerkzeuges sowohl auf dem iPhone als auch auf einem Android-Smartphone gefordert (siehe Anforderung 11). Daher ist eine Umsetzung als native Applikation nicht geeignet. Hybrid- und Container-Applikationen besitzen den Vorteil, einer besseren Unterstützung für den Zugriff auf die Gerätehardware des Smartphones. Da jedoch keine Anforderung an das Entwicklungswerkzeug bzgl. eines Zugriffs auf die Gerätehardware existiert, erscheint eine Umsetzung als Webapplikation zweckdienlich.

10.4 Verbesserung des Antwortverhaltens

Im Rahmen der Evaluation des Codegenerators (siehe Kapitel 9.3) hat sich gezeigt, dass die Datenabfrage vom SAP-ERP-System teilweise über 10 Sekunden dauert. Als Ursache wurde die Verwendung des SOAP-Protokolls vermutet, welches durch seine generische, XML-basierte Form im Vergleich zu proprietären Netzwerkprotokollen weniger performant ist. Um

eine Verbesserung des Antwortverhaltens zu erreichen soll bei der Implementierung des high-fidelity Prototypen das proprietäre Netzwerkprotokoll des SAP-ERP-Systems verwendet werden, das sogenannte RFC Protokoll (siehe Kapitel 2.1.5.2).

Bei der Implementierung der Benutzungsschnittstelle zum Testen des Codegenerators wurde die serverseitige Skriptsprache PHP verwendet (siehe Kapitel 9.2.3). Für PHP existiert ein Erweiterungsmodul SAPRFC, welches die Netzwerkkommunikation über das RFC-Protokoll ermöglicht (Sourceforge o.J.). Dies ermöglicht den Aufruf von BAPIs über das RFC-Protokoll aus einem PHP-Skript. Obgleich die Schritte aus der Dokumentation zur Installation von SAPRFC befolgt wurden, waren die Tests anschließend erfolglos. Eine mögliche Ursache könnte sein, dass die letzte Versionsaktualisierung des Erweiterungsmoduls im Jahr 2005 vorgenommen wurde und das Modul mit neueren Windows Versionen evtl. nicht kompatibel ist. Aufgrund der erfolglosen Tests von SAPRFC musste eine andere Möglichkeit zur Nutzung des RFC-Protokolls gefunden werden.

Eine alternative Möglichkeit besteht in der Nutzung des *SAP Java Connectors*, kurz *JCo*. Dabei handelt es sich um eine Programmierschnittstelle zur Nutzung des RFC-Protokolls in der Programmiersprache Java. JCo unterstützt eine bidirektionale Kommunikation zwischen einer Java Applikation und dem SAP-ERP-System. Dies bedeutet, dass die Programmierschnittstellen des SAP-ERP-Systems einerseits aus der Java Applikation aufgerufen werden können. Andererseits ist es auch möglich die Programmierschnittstellen der Java Applikation aus dem SAP-ERP-System anzusprechen (SAP o.J.-c). Wenn eine Java Applikation ein SAP BAPI aufrufen möchte, muss ein Aufruf der JCo Java API aus dem Java Quellcode erfolgen. Dieser wird anschließend über eine Middleware Schnittstelle in die RFC Middleware Schnittstelle überführt. Anschließend erfolgt eine Übersetzung des Aufrufs über eine Java Native Interface (JNI) Schicht in einen RFC-Aufruf (SAP o.J.-c). Abbildung 10-8 illustriert die beschriebene JCo-Kommunikationsstruktur.

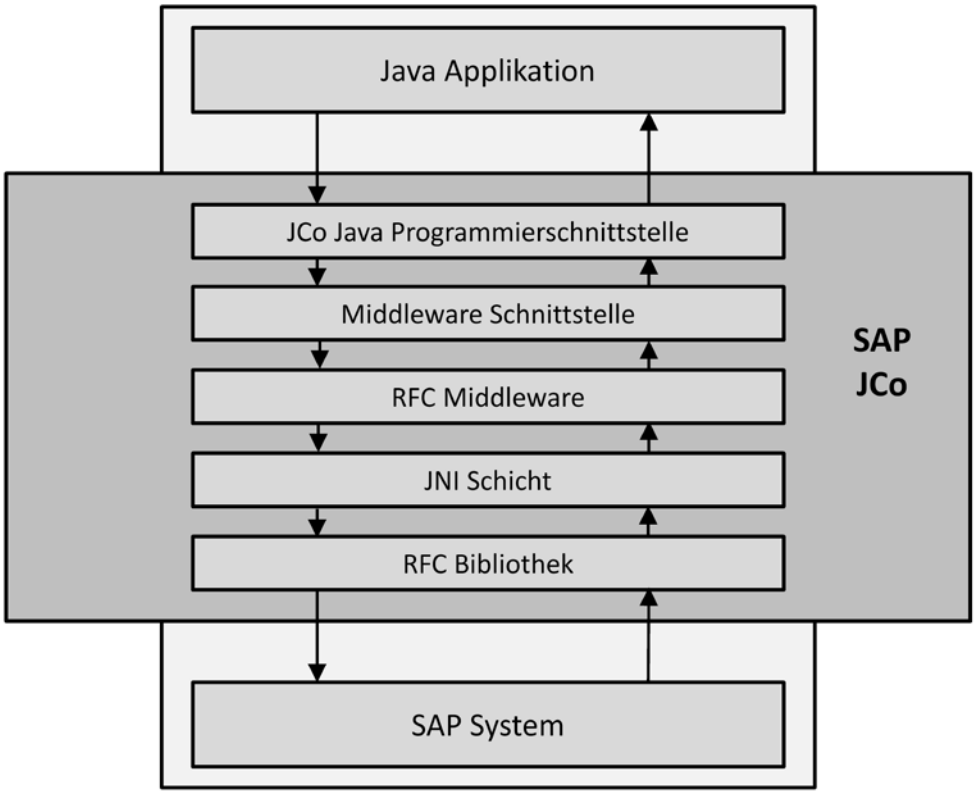

Abbildung 10-8: SAP JCo Kommunikationsstruktur
Quelle: Eigene Darstellung in Anlehnung an (SAP o.J.-c)

Die Nutzung von JCo erfordert den Einsatz der Programmiersprache Java. Um den skizzierten Architekturentwurf des Entwicklungswerkzeuges aus Kapitel 9.1.5 umsetzen zu können, muss eine geeignete Java-basierte serverseitige Webtechnologie gefunden werden. Für die vorliegenden Anforderungen erscheinen *Java Servlets* als geeignet. Dabei handelt es sich um Java-Klassen, deren Instanzen innerhalb eines Webservers Anfragen von Clients, wie bspw. Webbrowsern, beantworten. Java Servlets sind Teil der Java Enterprise Edition (EE) und benötigen einen sogenannten *Servlet-Container* als Laufzeitumgebung (Perry 2004, 1 ff.).

10.5 Erweiterung des Funktionsumfangs

In den bisherigen Ausführungen wurde ein Teil der Anforderungen nicht berücksichtigt. Dies betrifft die Anforderungen einer Verteilungsfunktion entwickelter Applikationen an andere Endbenutzer (siehe Anforderung 13) und die Nutzung von Layoutschablonen zur Umsetzung einer Corporate Identity (siehe Anforderung 6). Die Umsetzung dieser Anforderungen ist Bestandteil der folgenden Ausführungen. Bevor eine Verteilungsfunktion konzipiert werden kann, muss zunächst die generelle Mehrbenutzerfähigkeit des Entwicklungswerkzeuges sowie eine Verwaltungsfunktionalität der entwickelten Applikationen umgesetzt werden. Daher werden die Konzeption der Mehrbenutzerfähigkeit und die Verwaltungsfunktionalität im folgenden Kapitel beschrieben. Anschließend folgen die Erläuterungen zur Umsetzung von Anforderung 13 und Anforderung 6.

10.5.1 Mehrbenutzerbetrieb und Applikationsverwaltung

Unter *Mehrbenutzerbetrieb* des Entwicklungswerkzeugs wird in diesem Kontext verstanden, dass unterschiedliche Endbenutzer das Werkzeug verwenden können. Die von einem Endbenutzer entwickelten Applikationen sollen zudem für andere Endbenutzer nicht sichtbar sein. Eine Ausnahme ist die explizite Nutzung der Verteilungsfunktion (siehe Kapitel 10.5.2), welche die entwickelte Applikation anderen Endbenutzern zur Verfügung stellen kann.

Unter *Applikationsverwaltung* wird in diesem Kontext verstanden, dass ein Endbenutzer seine bereits entwickelten mobilen ERP-Applikationen einsehen und ausführen kann. Er soll jedoch auch die Möglichkeit besitzen seine bereits erstellten mobilen ERP-Applikationen zu bearbeiten oder zu löschen.

Um einen Mehrbenutzerbetrieb zu ermöglichen wird im high-fidelity Prototypen eine rudimentäre Nutzerverwaltung implementiert. Auf der ersten Bildschirmmaske des Entwicklungswerkzeuges wird eine Anmeldung über einen Nutzernamen und ein Passwort verlangt. Bei der ersten Nutzung des Entwicklungswerkzeuges kann ein eigener Nutzername und ein zugehöriges Passwort erstellt werden. Beim Anlegen eines neuen Nutzernamens wird geprüft, ob der Nutzername bereits existiert. Zudem wird für jeden Nutzernamen ein gleichnamiger Ordner im Dateisystem des Middleware-Servers angelegt. In diesem werden die erstellten Applikationsspezifikationen (Dateiendung *.xml*) und die ausführbaren Applikationen (Dateiendung *.html*) abgelegt. Jeder angemeldete Nutzer hat nur Zugriff auf seinen eigenen Applikationsordner. Beim Start des Entwicklungswerkzeuges werden die Dateinamen der erstellten ausführbaren Applikationen ausgelesen und in einer Liste angezeigt.

10.5.2 Verteilungsfunktion

Die Umsetzung einer Verteilungsfunktion leitet sich aus Anforderung 13 ab. Dabei wird gefordert, dass ein Endbenutzer eine entwickelte Applikation anderen Endbenutzern zur Nutzung zur Verfügung stellen kann.

Um eine Verteilungsfunktion auf Basis der im vorherigen Kapitel erläuterten Nutzer- und Applikationsverwaltung zu ermöglichen, wäre eine Möglichkeit einen eingeschränkten Zugriff auf die ausführbaren mobilen ERP-Applikationen im Applikationsordner eines anderen Endbenutzers zu ermöglichen. Eine alternative Umsetzungsvariante ist das Kopieren der ausführbaren mobilen ERP-Applikationen in den Applikationsordner anderer Endbenutzer. Letztere stellt aus Implementierungsperspektive eine vergleichsweise einfach umzusetzende Variante dar. Nachteilig bei dieser Umsetzungsvariant ist, dass eine nachträgliche Änderung der Applikation auf die verteilten Instanzen keine Auswirkung hat. Um diese zu ermöglichen, müsste jedoch eine aufwändigere Applikationsverwaltung konzipiert und implementiert werden. Der in dieser Arbeit erstellte low-fidelity Prototyp beschränkt sich auf eine rudimentäre Verteilungsfunktion und setzt daher die beschriebenen Kopiermechanismus um. Abbildung 10-9 illustriert die beschriebene Verteilungsfunktion.

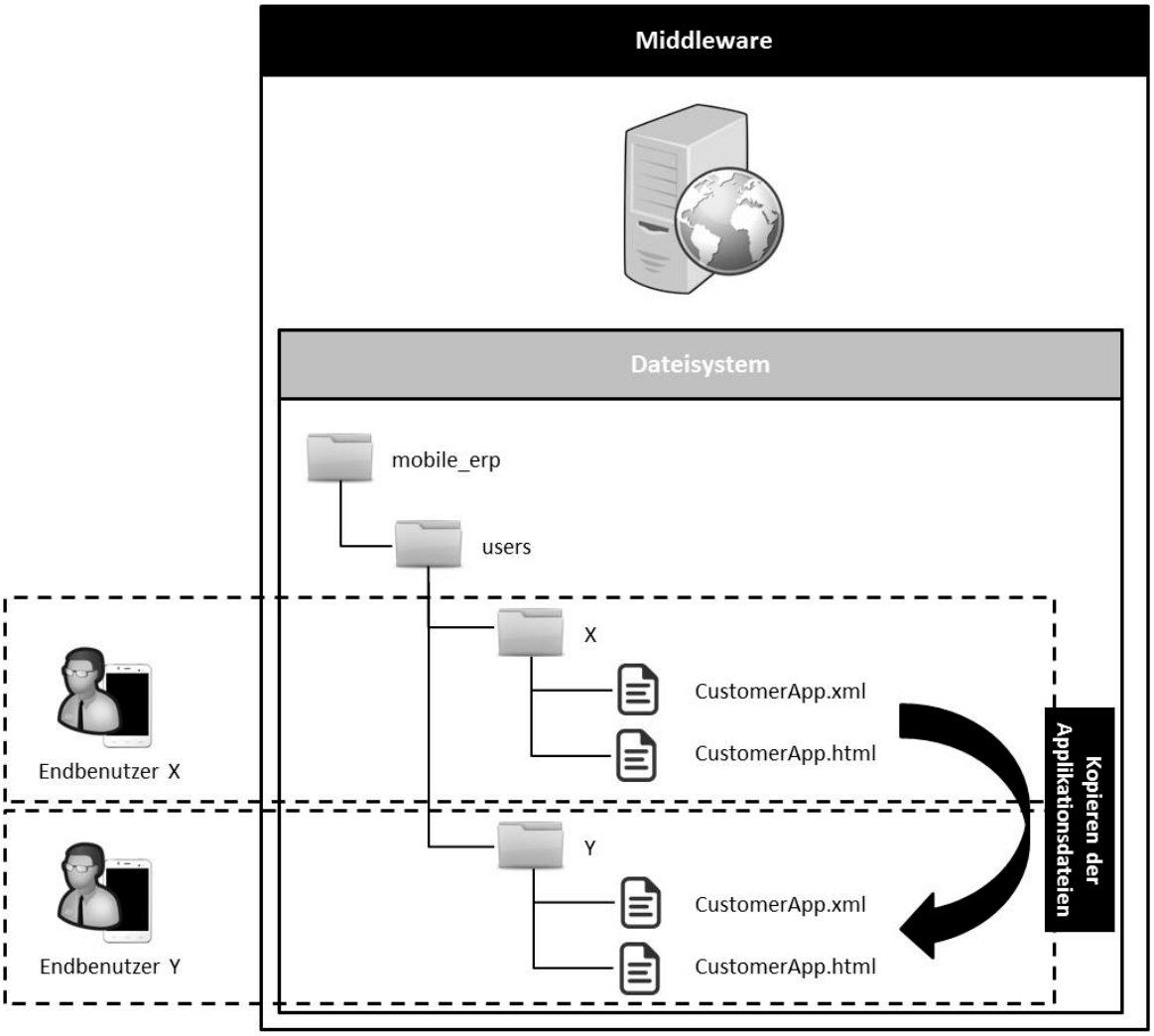

Abbildung 10-9: Illustration der Applikations-Verteilungsfunktion

Quelle: Eigene Darstellung

10.5.3 Layoutschablonen

Die Umsetzung von Layoutschablonen leitet sich aus Anforderung 6 ab. Hierbei wird gefordert, dass wiederverwendbare Layoutschablonen für die Entwicklung von mobilen ERP-Applikationen angeboten werden, um die Corporate Identity eines Unternehmens umsetzen zu können.

Eine Layoutschablone umfasst im Verständnis dieser Arbeit folgende Aspekte:

- ein Farbschema, welches die Farben der Anzeige- und Bedienelemente festlegt
- ein optionales Unternehmenslogo im Fußbereich der Bildschirmmasken der mobilen Applikationen
- ein optionales Hintergrundbild auf den Bildschirmmasken der mobilen Applikationen
- ein optionaler Copyright-Vermerk

In der vorgestellten prototypischen Implementierung wird jQuery Mobile als zugrundeliegendes HTML5 Framework zur Umsetzung der mobilen ERP-Applikationen eingesetzt (siehe Kapitel 9.1.1). Um Farbschemas für die Anzeige- und Bedienelemente einer mobilen Applikation zu definieren, stellt jQuery Mobile sogenannte *Themes* zur Verfügung (Bai 2011, 62 ff.). Diese basieren auf einer Menge von CSS-Elementen (Bai 2011, 63). jQuery Mobile stellt bereits eine Reihe von vordefinierten Themes zur Verfügung, von welchen diese Arbeit Gebrauch macht. Zusätzlich können über das jQuery Mobile Werkzeug „Theme Roller[62]“ eigene Farbschemas kreiert werden. Diese können bei Bedarf ebenfalls in das Entwicklungswerkzeug integriert werden.

Das Einbinden eines Unternehmenslogos erfolgt über den Aufruf einer Grafikdatei im HTML-Code der mobilen ERP-Applikation. Im Fußbereich der Bildschirmmaske wird über ein zugehöriges **`<img>`** Tag die gewünschte Grafik eingebunden. Hierzu muss der Speicherort der Grafikdatei über eine URL zugreifbar sein.

Ähnlich zur Einbindung eines Unternehmenslogos kann ein Hintergrundbild für die Bildschirmmasken der mobilen ERP-Applikation eingebunden werden. Auch in diesem Fall muss der Speicherort der gewünschten Grafikdatei über eine URL zugreifbar sein.

In der prototypischen Implementierung wird der Ansatz verfolgt, dass der Endbenutzer eine Layoutschablone nicht selbst entwickeln kann. Stattdessen wird eine Menge verfügbarer Layoutschablonen zur Auswahl angeboten. In einer zukünftigen Implementierung könnte ein zusätzliches Werkzeug für fortgeschrittene Endbenutzer bereitgestellt werden, welches die Entwicklung eigener Layoutschablonen ermöglicht. In der prototypischen Implementierung werden die Layoutschablonen direkt im XSLT-Stylesheet (siehe Kapitel 9.2.2) umgesetzt. Abbildung 10-10 illustriert die Auswahl einer Layoutschablone im Entwicklungswerkzeug.

[62] http://themeroller.jquerymobile.com/, zugegriffen am 14.03.2014

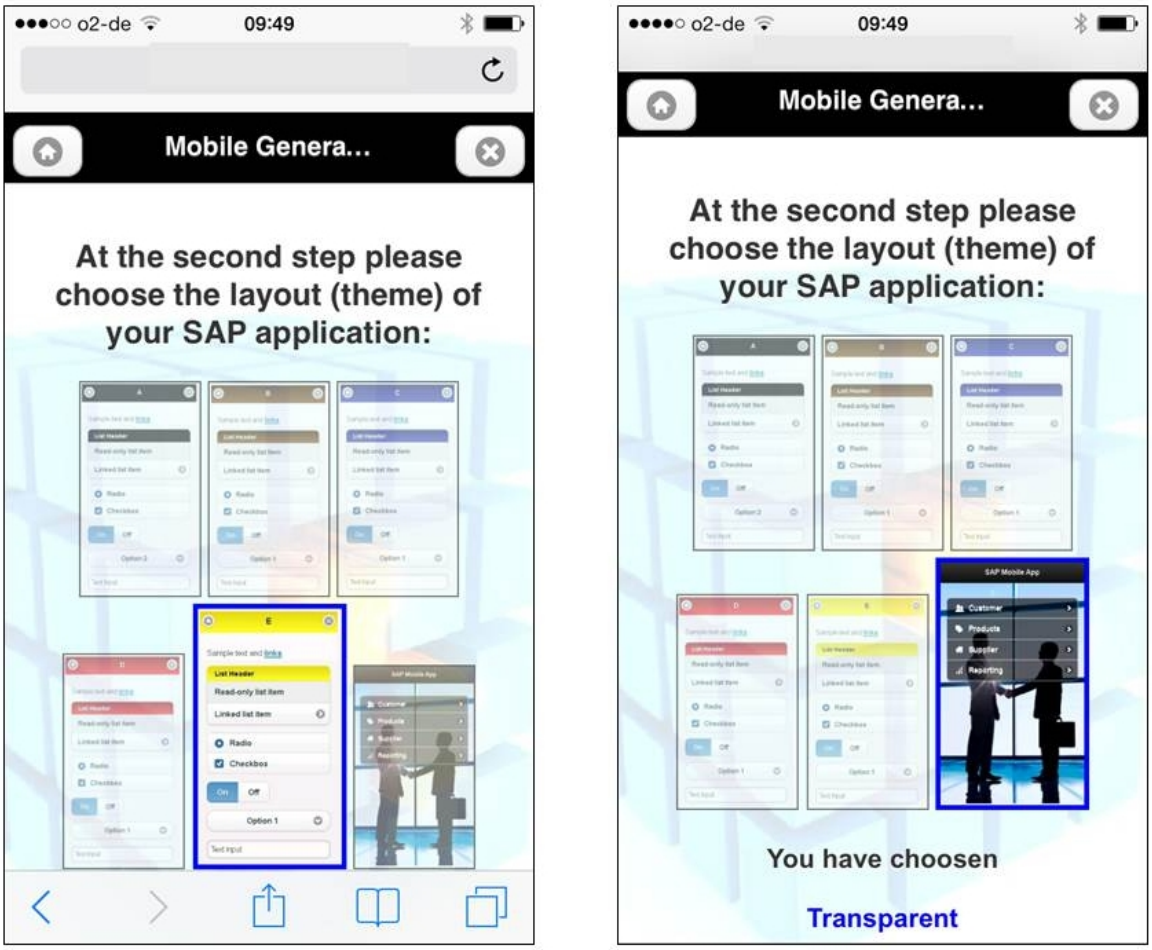

Abbildung 10-10: Auswahl einer Layoutschablone im Entwicklungswerkzeug
Quelle: Eigene Darstellung

10.6 Erweiterter und angepasster Architekturentwurf

Um die im vorherigen Kapitel beschriebene Erweiterung des Funktionsumfanges des Entwicklungswerkzeuges sowie die Verbesserungsvorschläge aus den bisherigen Evaluationsergebnissen zu berücksichtigen wird der in Kapitel 9.1.5 vorgestellte Architekturentwurf angepasst.

Um die gewünschte Verbesserung des Antwortverhaltens der entwickelten mobilen ERP-Applikationen zu erzielen wird die Kommunikation mit dem SAP-ERP-System vom offenen SOAP-Protokoll auf das proprietäre SAP-Protokoll RFC umgestellt. Um die Nutzung dieses Protokolls zu ermöglichen wird der SAP JCo SAP eingesetzt. Um Anforderung 11 zu erfüllen, wurde die Benutzungsoberfläche des Entwicklungswerkzeuges auf ein Smartphone

portiert. Zur Umsetzung der Benutzungsschnittstelle des Entwicklungswerkzeuges werden analog zu den mobilen ERP-Applikationen selbst HTML5-Technologien eingesetzt. Hierdurch ist das Entwicklungswerkzeug auf den beiden mobilen Betriebssystemen, iOS und Android, lauffähig. Abbildung 10-11 illustriert den angepassten Architekturentwurf.

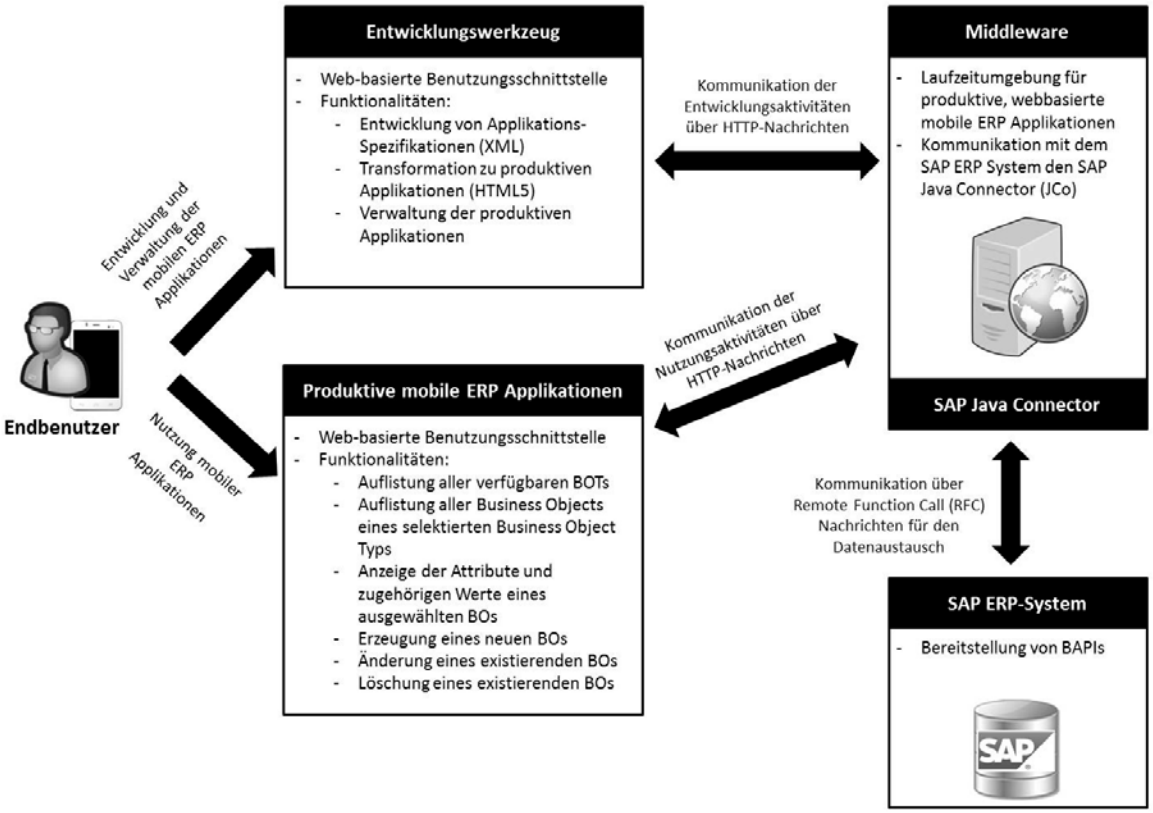

Abbildung 10-11: Angepasster Architekturentwurf des Entwicklungswerkzeuges
Quelle: Eigene Darstellung

10.7 Prototypische Umsetzung

Nachdem im vorherigen Kapitel der erweiterte und angepasste Architekturentwurf für das Entwicklungswerkzeug skizziert wurde, wird in diesem Kapitel dessen prototypische Umsetzung beschrieben. Bestandteile der Beschreibung sind die eingesetzten Technologien, der Aufbau der Testumgebung und die implementierte Benutzungsschnittstelle.

10.7.1 Technologische Basis und Aufbau der Testumgebung

Um den skizzierten Architekturentwurf aus Kapitel 10.6 umzusetzen, ist eine geeignete technologische Basis für die prototypische Umsetzung der Middleware notwendig. In der prototypischen Implementierung des Codegenerators aus Kapitel 9.2 wurde ein Apache HTTP Server verwendet. Zur Umsetzung der Applikationslogik wurde die Programmiersprache PHP eingesetzt. Um die gewünschte Verbesserung des Antwortverhaltens bei der Kommunikation mit dem SAP-ERP-System erreichen zu können, soll das proprietäre SAP Netzwerkprotokoll

RFC in Kombination mit dem SAP JCo verwendet werden (siehe Kapitel 10.4). Für den Einsatz des SAP JCo ist die Nutzung der Programmiersprache Java notwendig. Demzufolge wird die Applikationslogik nicht mehr in der Programmiersprache PHP, sondern in der Programmiersprache Java implementiert. Da die vorliegende Implementierung eine serverseitige Programmlogik repräsentiert, wird die zugehörige Java-basierte Technologie, die sogenannten *Servlets* verwendet. Dabei handelt es sich um serverseitige Komponenten der Java Enterprise Edition (EE), welche eine dynamische Reaktion auf Anfragen der Clients (i.d.R. Webbrowser) ermöglichen (Perry 2004, 1 ff.).

Für den Einsatz von Servlets ist eine zugehörige Laufzeitumgebung erforderlich, ein sogenannter *Servlet-Container*. Für die Testumgebung dieser Arbeit wird der verbreitete Servlet-Container *Tomcat* [63] der Apache Software Foundation verwendet. Dieser stellt eine Kombination aus einem Webserver und einem Servlet-Container dar und ermöglicht die Ausführung von Webapplikation auf Basis einer Servlet-basierten Programmlogik (Apache 2014). Zur Transformation der XML-basierten Applikationsspezifikationen in lauffähige HTML5-basierte mobile ERP-Applikationen, wird der Java-kompatible XSLT-Prozessor Xalan-J[64] verwendet. Die genannten Softwarekomponenten wurden auf einem Microsoft Windows Server 2008 R2 Betriebssystem installiert. Insgesamt besteht die Testumgebung des Middleware-Servers aus folgenden Komponenten:

- Tomcat Webserver und Servlet-Container in Version 7.0
- Xalan-J XSLT-Prozessor in Version 2.7.0
- SAP Java Connection in Version 3.0
- Windows Server 2008 R2 Betriebssystem

Als ERP-System wurde ein SAP-ERP-System mit nachfolgenden Ausprägungen verwendet:

- Komponentenversion: SAP Enterprise Core Component (ECC) in Version 6.0
- Datenbank: IBM DB2 Enterprise Server Edition (ESE) Version in 9.7
- Betriebssystem: AIX Version 6.1 Technology Level (TL) 7

Analog zur Testumgebung zur Evaluation des Codegenerators (siehe Kapitel 9.2.1) wurde aufgrund der großen Anzahl an existierenden BOTs und der aufwändigen Parametrisierung zugehöriger BAPIs, eigene, weniger komplexe Programmierschnittstellen erstellt. Hierzu wurden eigene BOTs mit zugehörigen BAPIs im verwendeten SAP-ERP-System entwickelt. Hierzu wurden BOTs zur Repräsentation von Kunden, Lieferanten und Produkten und BAPIs

[63] http://tomcat.apache.org, zugegriffen am 15.03.2014
[64] http://xalan.apache.org/xalan-j/index.html, zugegriffen am 15.03.2014

zum Lesen, Anlegen, Ändern und Löschen zugehöriger BOs implementiert. Zudem wurde für die Evaluation eine Reihe von Testdatensätzen bereitgestellt.

Um die Benutzungsschnittstelle des prototypisch implementierten Entwicklungswerkzeuges sowie der entwickelten mobilen ERP-Applikationen zu testen, wurden folgenden Testgeräte verwendet:

- Apple iPhone 4s mit installiertem iOS in Version 5.1
- Samsung Galaxy S3 mini mit installiertem Android in Version 4.1

Abbildung 10-12 illustriert den beschriebenen Aufbau der Testumgebung.

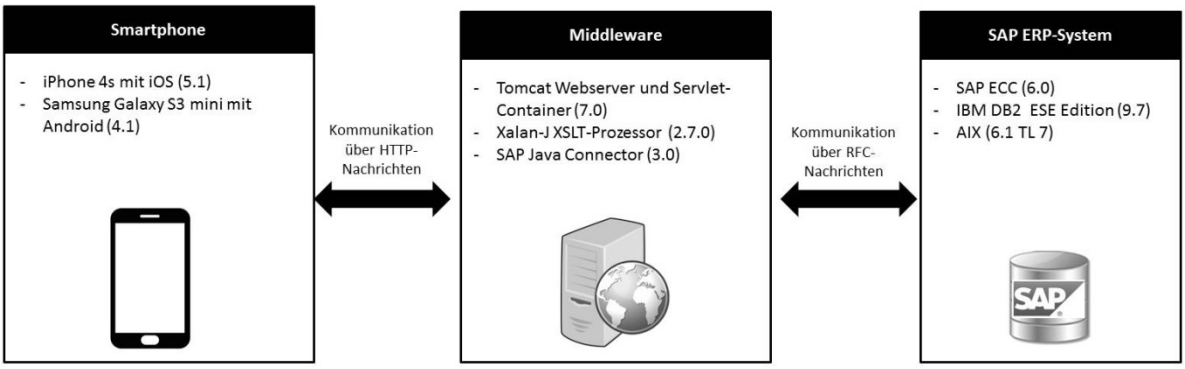

Abbildung 10-12: Testumgebung zur Evaluation des Entwicklungswerkzeuges
Quelle: Eigene Darstellung

10.7.2 Benutzungsschnittstelle

In Kapitel 10.3 wurde eine erweitere und optimierte Benutzungsschnittstelle des Entwicklungswerkzeuges auf konzeptueller Ebene vorgestellt, welche die bisherigen Evaluationsergebnisse berücksichtigt. Dabei wurde die Navigationsstruktur sowie die Gestaltung einzelner Bildschirmmasken vorgestellt. In diesem Kapitel wird die prototypische Umsetzung der vorgestellten Konzepte behandelt.

Zur prototypischen Implementierung der Benutzungsschnittstelle wurde analog zur Implementierung der mobilen ERP-Applikationen das HTML5-basierte *jQuery Mobile* Framework eingesetzt. Der Vorteil von jQuery Mobile gegenüber anderen HTML5-basierten Frameworks für die vorliegende Arbeit wurde bereits in Kapitel 8.4.3.1 diskutiert. Zudem reduziert die Verwendung eines einheitlichen Frameworks für das Entwicklungswerkzeug und die mobilen ERP-Applikationen die Komplexität der prototypischen Implementierung.

In Abbildung 10-13 sind die implementierten Bildschirmmasken zur Anmeldung über ein existierendes Benutzerkonto (in der Abbildung links) und zur Applikationsverwaltung (in der

Abbildung rechts) illustriert. Die Anmeldung erfolgt über einen existierenden Benutzernamen und ein zugehöriges Passwort. Sollte kein Benutzerkonto existieren, kann über die Schaltfläche „Create“ ein neues Benutzerkonto angelegt werden.

In der illustrierten Bildschirmmaske zur Applikationsverwaltung ist erkennbar, dass bereits zwei mobile ERP-Applikationen entwickelt wurden. Durch die Selektion einer der gelisteten Applikationen ist deren Nutzung möglich. Die Funktionalitäten zum Löschen oder Bearbeiten einer existierenden Applikation sind über die Schaltflächen im unteren Bereich der Bildschirmmaske erreichbar. Zudem kann über die Schaltfläche „Create“ der Entwicklungsassistent zur Entwicklung einer neuen mobilen ERP-Applikation aufgerufen werden.

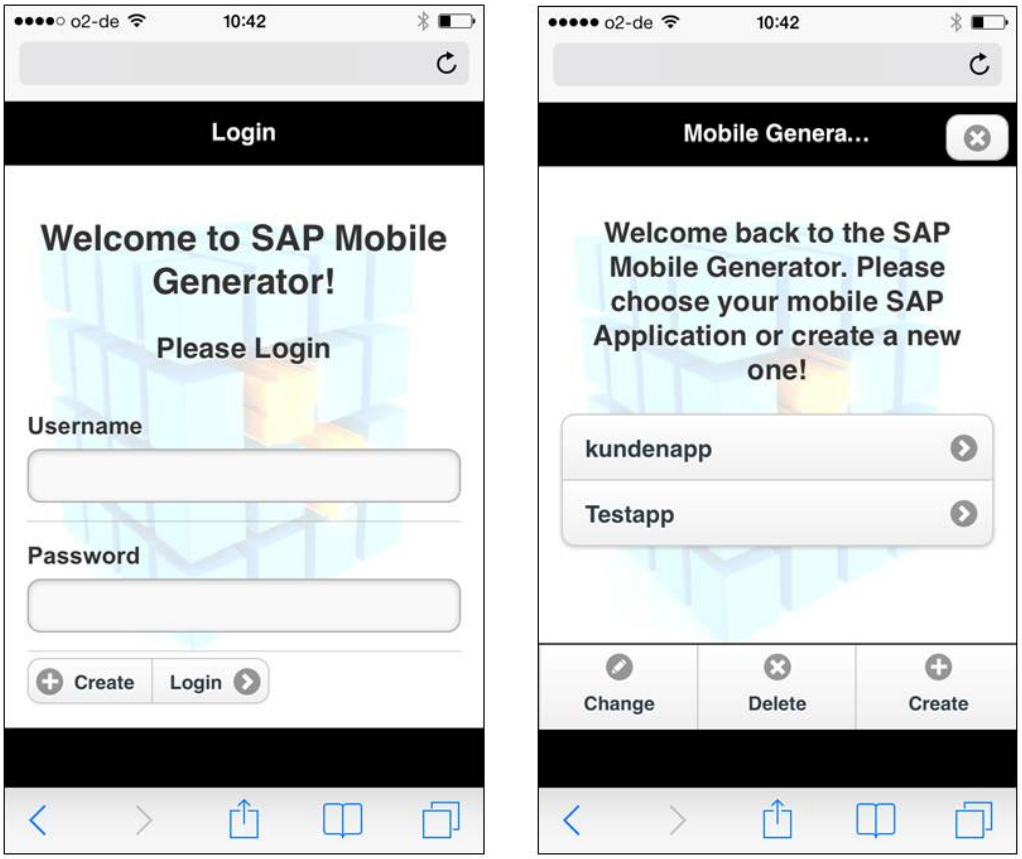

Abbildung 10-13: Bildschirmmasken zur Benutzer- und Applikationsverwaltung
Quelle: Eigene Darstellung

In Abbildung 10-14 ist ein Ausschnitt der prototypisch implementierten Bildschirmmasken des Entwicklungsassistenten illustriert. In der Einstiegsbildschirmmaske des Entwicklungsassistenten (in der Abbildung links) muss der Name der neuen Applikation eingegeben werden. Sollte der eingegebene Applikationsname bereits existieren oder Sonderzeichen enthalten, so ist eine Navigation zur nächsten Bildschirmmaske nicht möglich. Stattdessen wird eine Fehlermeldung unterhalb des eingegebenen Applikationsnamens angezeigt. Über die Schaltfläche „Next“ kann zur nächsten Bildschirmmaske navigiert werden. In dieser Bildschirmmaske kann das Layout der Applikation gewählt werden (in der Abbildung in der Mitte), um entweder einer persönlichen Präferenz oder der Corporate Identity des Unternehmens zu entsprechen. In der prototypischen Implementierung wurden sechs beispielhafte Layouts angelegt, welche zur Auswahl stehen. In der folgenden Bildschirmmaske (in der Abbildung rechts) müssen die gewünschten MBOTs gewählt werden. In der beispielhaften Abbildung wurde das MBOT „Suppliers“ gewählt, um eine Applikation zur Verwaltung von Lieferantenstammdaten zu entwickeln.

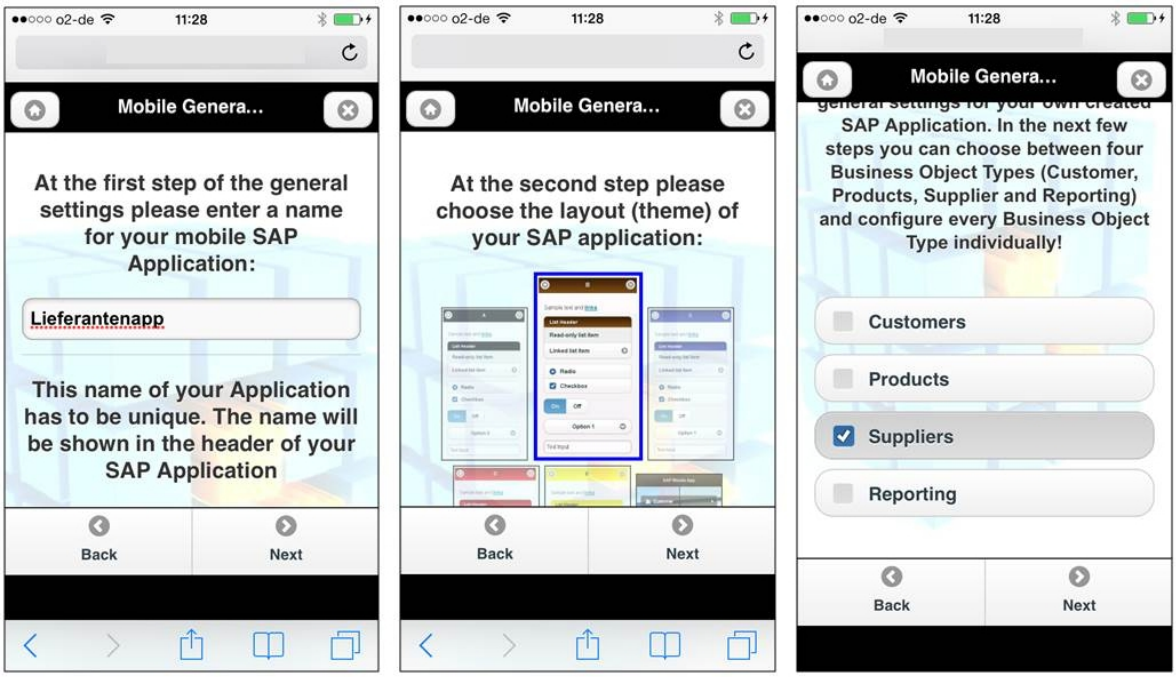

Abbildung 10-14: Prototypisch implementierte Bildschirmmasken des Entwicklungsassistenten (Teil 1)
Quelle: Eigene Darstellung

In Abbildung 10-15 ist ein Teil der prototypisch implementierten Bildschirmmasken zur Spezifikation der MBOT-Attribute und MBOT-Methoden illustriert. Nach Auswahl eines MBOT in einem vorherigen Entwicklungsschritt muss nun das in der sortierten Listenanzeige verwendete Attribut des MBOT ausgewählt werden (in der Abbildung links). In der folgenden Bildschirmmaske können die in der formularbasierten MBO-Anzeige verwendeten Attribute ausgewählt werden (in der Abbildung in der Mitte). Hierbei werden alle Attribute des MBOT aufgelistet und können einzelnen selektiert werden. In der darauffolgenden Bildschirmmaske werden die verfügbaren MBOT-Methoden inkl. einer knappen Beschreibung angezeigt (in der

Abbildung rechts). Der Endbenutzer hat hierbei die Möglichkeit die angezeigten Methoden zu selektieren bzw. deselektieren.

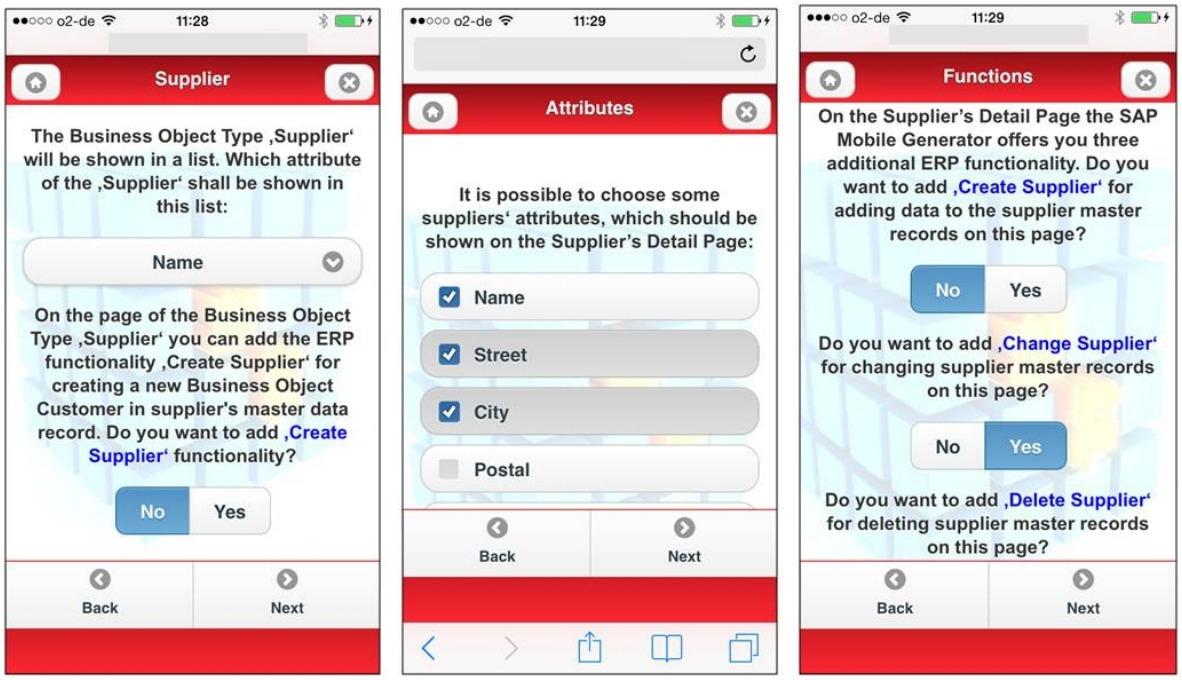

Abbildung 10-15: Prototypisch implementierte Bildschirmmasken des Entwicklungsassistenten (Teil 2)
Quelle: Eigene Darstellung

10.8 Evaluation

Ein Teil der zu prüfenden Evaluationsaspekte (siehe Kapitel 7.3) wurde bereits in vorherigen Kapitelen geprüft. Der restliche Teil der Evaluationsaspekte (EA5, EA6, EA8, EA11, E13) konnte bisher noch nicht geprüft werden, da die zu prüfenden Aspekte in den vorherigen Implementierungsstufen noch nicht umgesetzt waren. Deren Prüfung ist Bestandteil der folgenden Ausführungen. Zusätzlich werden die Evaluationsaspekte EA4 und EA7 untersucht. Diese beschäftigen sich mit der Notwendigkeit eines Endbenutzer-Trainings für die Nutzung des Entwicklungswerkzeuges sowie der Dauer des Entwicklungsvorganges. Um diese beiden Aspekte zu prüfen wurde in kontrolliertes Experiment mit potentiellen Endbenutzern als Probanden vorgeschlagen. Da zur Durchführung des Experimentes ein funktionstüchtiger Prototyp (high-fidelity Prototyp) notwendig ist, können diese Aspekte erst mit Hilfe der in Kapitel 10.7 beschriebenen prototypischen Implementierung durchgeführt werden. Um die beiden Evaluationsaspekte zu prüfen wird ein kontrolliertes Experiment zur Prüfung der Benutzungsfreundlichkeit durchgeführt. Nach Vorstellung von Verfahren zur Evaluation der Benutzungsfreundlichkeit wird der Aufbau des Experimentes beschrieben und die Evaluationsergebnisse dargestellt.

10.8.1 Statische Analyse zur Prüfung von EA5

Evaluationsaspekt EA5 erfordert die Prüfung eines schrittweisen Entwicklungsprozesses. Bei der Beschreibung der Benutzungsschnittstelle in Kapitel 10.3 wurde die Navigationsstruktur des Entwicklungsassistenten vorgestellt (siehe Abbildung 10-5). Zusätzlich wurden in Kapitel 10.7.2 die einzelnen Bildschirmmasken der prototypischen Implementierung des Entwicklungsassistenten illustriert und beschrieben. Dabei startet der Entwicklungsprozess mit der Eingabe und Selektion von allgemeinen Applikationseinstellungen. Anschließend werden die gewünschten MBOT-Attribute und Methoden selektiert. Wie in Abbildung 10-5 illustriert erfolgen diese Aktivitäten Schritt-für-Schritt. Dabei hat der Endbenutzer neben der Möglichkeit zur Vorwärtsnavigation auch die Möglichkeit zur Rückwärtsnavigation, um vorherige Eingaben zu ändern. Zusammenfassend kann Evaluationsaspekt EA5 somit positiv bestätigt werden.

10.8.2 Funktionaler Test zur Prüfung von EA6

Evaluationsaspekt EA6 erfordert die Prüfung der Existenz und Nutzbarkeit von Layoutschablonen. Hiermit sollen ein Farbschema eines Unternehmens sowie ein Unternehmenslogo in die Applikation integriert werden können, um die Corporate Identity des Unternehmens zu unterstützen.

In der prototypischen Implementierung des Entwicklungswerkzeuges wurden sechs verschiedene Layoutschablonen implementiert. Diese unterscheiden sich im Wesentlichen im verwendeten Farbschema. Neben fünf einfacheren Layoutschablonen wurde eine komplexere Layoutschablone mit Hintergrundgrafik implementiert. Im Rahmen eines funktionalen Tests wurde geprüft, ob die bereitgestellten Layoutgrafiken selektierbar sind und ob deren Wahl in der entwickelten Applikation angewendet wird. Dieser funktionale Test wurde für alle sechs Layoutschablonen durchgeführt. Das Ergebnis hat gezeigt, dass alle Layoutschablonen selektierbar und nutzbar sind. Zusammenfassend kann der Evaluationsaspekt EA6 positiv bestätigt werden. Abbildung 10-16 illustriert die Bildschirmmaske zur Auswahl der verfügbaren Layoutschablonen (in der Abbildung links) sowie zwei beispielhafte Bildschirmmasken von Applikationen mit unterschiedlichen Layoutschablonen.

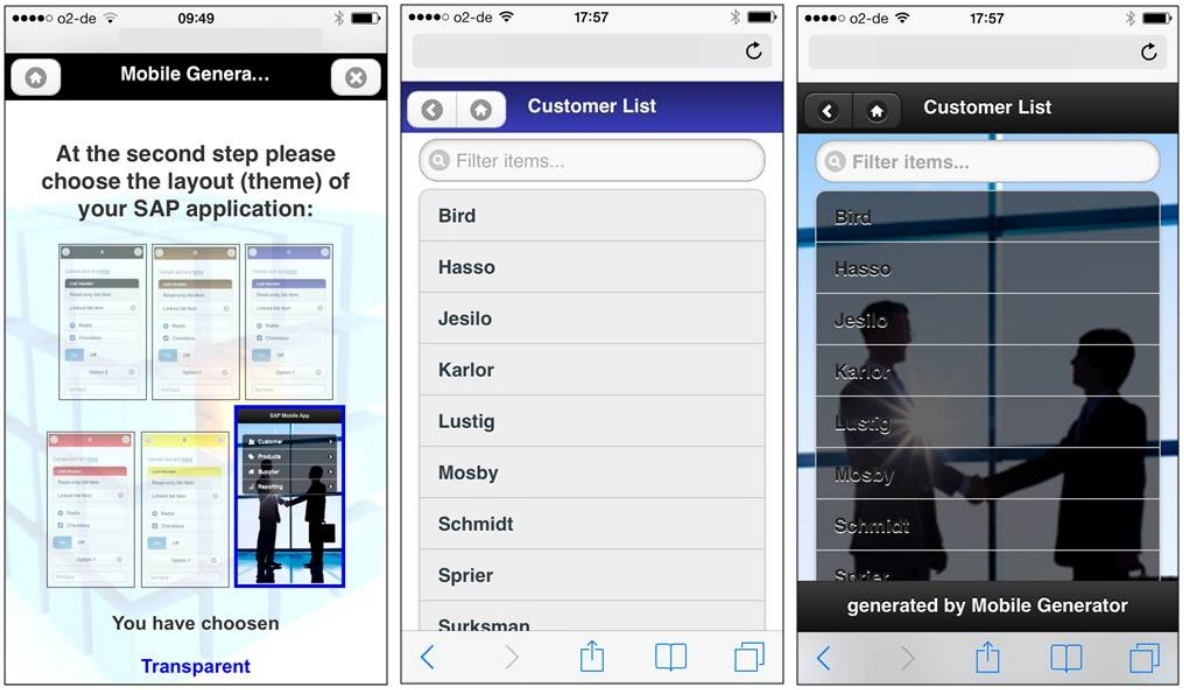

Abbildung 10-16: Funktionaler Test bzgl. EA6 auf einem iPhone
Quelle: Eigene Darstellung

10.8.3 Statische Analyse zur Prüfung von EA8

Evaluationsaspekt E8 erfordert die Prüfung der Konfigurationsmöglichkeiten zur Einstellung der Verbindung zum SAP-ERP-System. Es wird gefordert, dass Endbenutzer keine Möglichkeit besitzen, Verbindungseinstellungen durchzuführen. Grund für diese Anforderung ist die Vermeidung von Frustration bei Endbenutzern, indem Werte für Konfigurationsparameter voreingestellt werden.

In der prototypischen Implementierung erfolgt die Konfiguration der Verbindungsparameter zum SAP-ERP-System über eine zentrale Textdatei. In dieser Textdatei sind die Werte für die notwendigen Verbindungsparameter zum SAP-ERP-System der Testumgebung hinterlegt. Dies sind die SAP-ID (SID), der Message-Server, die Logon-Gruppe, die Instanznummer, der SAP-Router-String sowie ein Benutzerkonto und das zugehörige Passwort. Diese Verbindungswerte werden von allen mobilen ERP-Applikationen gemeinsam verwendet. Eine individuelle Eingabe der Werte wird durch das Entwicklungswerkzeug oder durch die entwickelten Applikationen nicht unterstützt. Demzufolge kann Evaluationsaspekt EA8 positiv bestätigt werden.

Nachteilig an der aktuellen Umsetzung ist, dass alle mobilen ERP-Applikationen mit dem gleichen Benutzerkonto auf das SAP-ERP-System zugreifen. Dies ist insbesondere bei Datenänderungen problematisch, da die Änderung anschließend keinem Benutzerkonto zugeordnet werden kann. In einer zukünftigen Ausbaustufe des Entwicklungswerkzeuges sollte ein

individuelles Benutzerkonto für den Zugriff auf das SAP-ERP-System konfiguriert werden können.

10.8.4 Funktionaler Test zur Prüfung von EA11

Evaluationsaspekt E8 erfordert die Prüfung der Lauffähigkeit des Entwicklungswerkzeuges auf den beiden mobilen Betriebssystemen iOS und Android. Um diesen Aspekt zu prüfen, wurde das Entwicklungswerkzeug auf den beiden Testgeräten, einem iPhone 4s mit iOS Version 5.1 und eine Samsung Galaxy S3 mini mit Android Version 4.1, getestet. Hierbei wurden alle Funktionen des Werkzeugs getestet und jeweils mehrere mobile ERP-Applikationen entwickelt. Zusammenfassend konnten auf beiden Testgeräten keine Einschränkungen festgestellt werden. Daher kann EA11 positiv bestätigt werden. Abbildung 10-17 illustriert die Bildschirmmaske zur Applikationsverwaltung auf dem Samsung Galaxy S3 (links) und auf dem iPhone 4s (rechts).

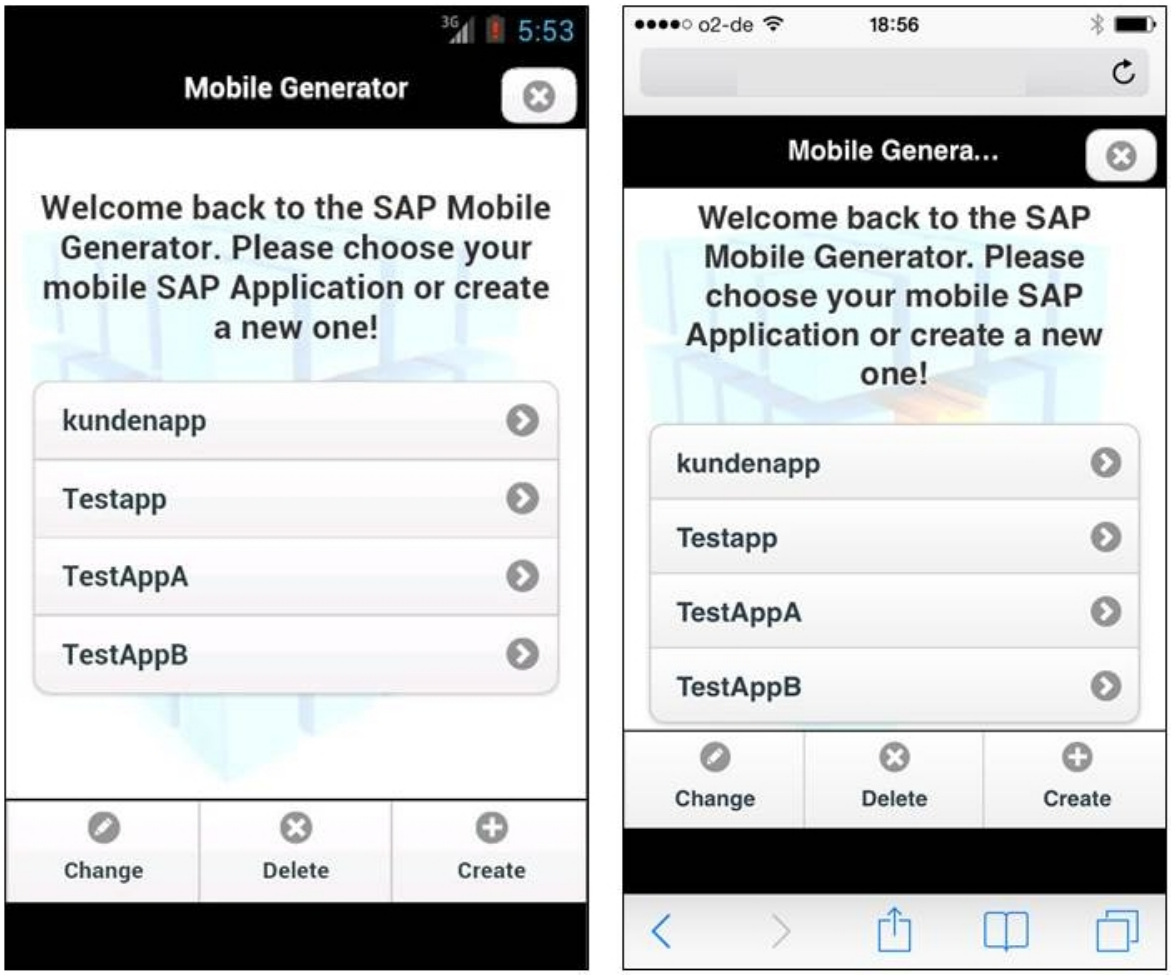

Abbildung 10-17: Funktionaler Test bzgl. EA11

Quelle: Eigene Darstellung

10.8.5 Funktionaler Test zur Prüfung von EA13

Evaluationsaspekt E13 erfordert den Test der Verteilungsfunktion der entwickelten Applikationen an andere Endbenutzer. Um dies zu evaluieren wurden fünf Benutzerkonten angelegt, mehrere Applikationen entwickelt und anschließend an andere Benutzerkonten verteilt. Anschließend erfolgte eine Anmeldung an diesen Benutzerkonten, um die Nutzung der verteilten Applikation zu prüfen. Zusammenfassend konnten alle Testaktivitäten erfolgreich geprüft werden und EA13 positiv bestätigt werden. Abbildung 10-18 illustriert die Bildschirmmaske der Verteilungsfunktion im beschriebenen Testszenario auf einem iPhone.

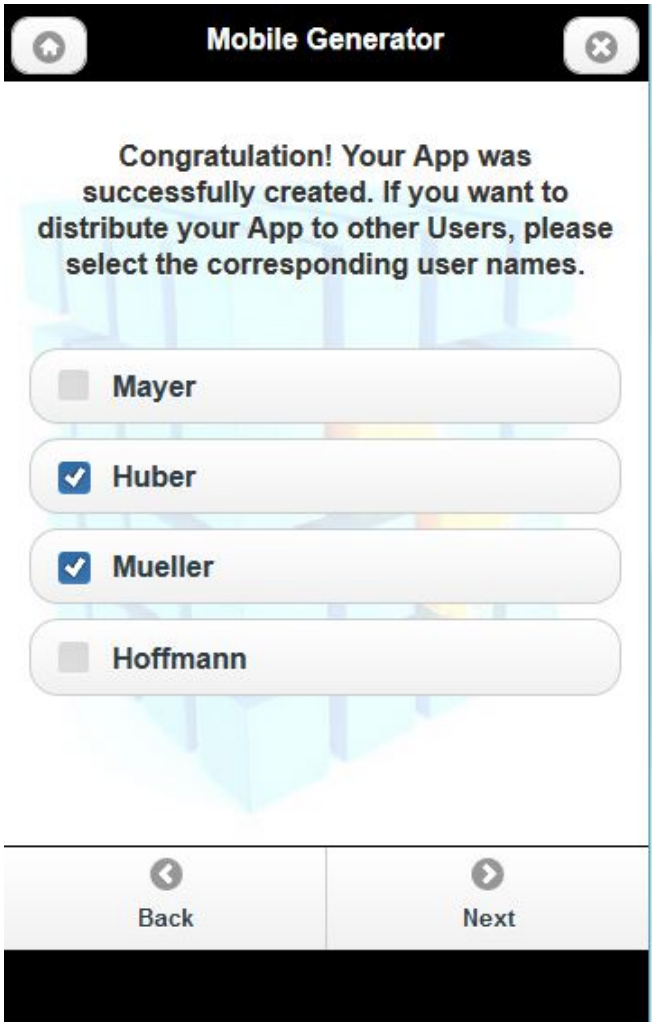

Abbildung 10-18: Prototypisch implementierte Bildschirmmaske der Verteilungsfunktion
Quelle: Eigene Darstellung

10.8.6 Kontrolliertes Experiment zur Untersuchung von EA4 und EA7

Der Evaluationsaspekt EA4 erfordert die Prüfung des notwendigen Trainingsaufwandes zur Nutzung des Entwicklungswerkzeuges. Es besteht die Anforderung das Entwicklungswerkzeug ohne vorherigen Trainingsaufwand nutzen zu können. Evaluationsaspekt EA7 erfordert die Prüfung der Dauer des Entwicklungsvorganges einer mobilen ERP-Applikation mit dem Entwicklungswerkzeug. Dieser sollte laut Anforderung 7 maximal 15 Minuten betragen.

Zur Prüfung der genannten Evaluationsaspekte, erscheint ein kontrolliertes Experiment mit potentiellen Endbenutzern als Probanden als zweckmäßig. Hierdurch soll ein möglichst realistisches Testergebnis erzielt werden. Im Rahmen eines solchen Experimentes könnten zudem auch weitere Verbesserungspotentiale der implementierten Benutzungsschnittstelle identifiziert werden.

Im Folgenden wird zunächst ein kurzer Überblick über Verfahren zur Untersuchung der Benutzungsfreundlichkeit gegeben. Anschließend wird die gewählte Methode in der vorliegenden Arbeit begründet und der Aufbau des Experimentes beschrieben. Schließlich werden die Evaluationsergebnisse vorgestellt.

10.8.6.1 Verfahren zur Untersuchung der Benutzungsfreundlichkeit

Unter dem Begriff *Benutzungsfreundlichkeit* bzw. *Gebrauchstauglichkeit* (engl. usability) wird das Ausmaß verstanden, „in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ (DIN 1998, 4). Das Kriterium *Effektivität* bezieht sich auf die Genauigkeit und Vollständigkeit der Erreichung des Aufgabenzieles (DIN 1998, 4). Das Kriterium *Effizienz* betrachtet hingegen den für die Aufgabenbewältigung benötigten Aufwand (DIN 1998, 4). Das Kriterium *Zufriedenheit* bezieht sich auf die Freiheit von Beeinträchtigungen und das positive Empfinden des Benutzers bei der Nutzung des Produktes (DIN 1998, 4). Aufgrund seiner subjektiven Ausprägung (DIN 1998, 7) ist das Kriterium Zufriedenheit am schwersten messbar. Zudem ist zu berücksichtigen, dass sich die drei Kriterien gegenseitig beeinflussen können. Beispielsweise führt eine hohe Effektivität und Effizient bei der Aufgabenbewältigung i.d.R. auch zu einer hohen Zufriedenheit.

Schwachstellen in der Benutzungsfreundlichkeit können personen-immanente Ursachen besitzen, beispielsweise aufgrund mangelnder Domänenkenntnisse oder mangelnder Kenntnisse der Benutzer im Umgang mit Computersystemen (Sarodnick/Brau, 25). Die zweite Gruppe von Schwachstellen der Benutzungsfreundlichkeit ist direkt auf die gewählte Gestaltungsvariante der Benutzungsschnittstelle zurückzuführen. Im Folgenden sollen in Anlehnung an Sarodnick und Brau (2011, 26) nur solche Benutzungsprobleme berücksichtigt werden, welche Benutzer mit hinreichenden Domänen- und Computerkenntnissen daran hindern ein fokussiertes Anwendungsszenario durchzuführen oder dessen Durchführung erschweren.

Verfahren zum Evaluieren der Benutzungsfreundlichkeit können in Bezug auf die evaluierende Personengruppe in *Expertentests* und *Nutzertests* unterteilt werden (Dix et al. 2004, 320

ff.). Expertentests basieren auf der Beurteilung einer Benutzungsschnittstelle durch Experten und haben folglich eine *analytische Ausprägung* (Sarodnick/Brau, 119). Unter einem Experten wird in diesem Zusammenhang ein *Usability-Experte* verstanden, d.h. eine Person mit einer speziellen Ausbildung oder einem ausgeprägten Erfahrungshintergrund hinsichtlich der Benutzbarkeit und Benutzungsproblemen von Softwareapplikationen (Sarodnick/Brau 2011, 144 ff.). Nutzertests gewinnen ihre Erkenntnisse hingegen durch Befragungen oder Beobachtungen tatsächlicher Benutzer und haben folglich eine *empirische Ausprägung* (Sarodnick/Brau, 119).

Verfahren zur Prüfung der Benutzungstauglichkeit können in unterschiedlichen Phasen des Gestaltungsprozesses eingesetzt werden. Entwicklungsbegleitende Evaluationen dienen der frühzeitigen Erkennung von Benutzungsschwachstellen sowie deren anschließender Beseitigung und haben folglich eine *formative Bewertungsfunktion* (Herczeg 2009, 154; Sarodnick/Brau 2011, 24). Da Expertentests im Vergleich zu Nutzertests mit weniger Aufwand in der Testvorbereitung und –durchführung verbunden sind, werden sie häufig entwicklungsbegleitend eingesetzt, um Designvorschläge zu evaluieren (Dix et al. 2004, 320). Evaluationen in der Endphase des Gestaltungsprozesses dienen hingen i.d.R. der Qualitätskontrolle und haben folglich eine *summative Bewertungsfunktion* (Herczeg 2009, 154; Sarodnick/Brau 2011, 24). Aufgrund der realistischeren Nutzungssituation durch repräsentative Benutzer werden in dieser Phase oftmals Nutzertests durchgeführt. Der Einsatz von Nutzertests in späteren Phasen des Entwicklungsprozesses ist auch darin begründet, dass hierfür i.d.R. funktionstüchtige Prototypen notwendig sind (Dix et al. 2004, 327).

Ein Beispiel für einen Expertentest ist das Verfahren *Cognitive Walkthrough*. (Wharton et al. 1994, 105 ff.; Dix et al. 2004, 321 f.). Voraussetzungen für dieses Testverfahren sind: eine Design Spezifikation der Benutzungsschnittstelle (z.B. in Form von Mock-ups), Annahmen über die Charakteristiken der potentiellen Benutzer, konkrete Aufgabenstellungen und eine Liste von Aktionen zur Bewältigung der genannten Aufgabenstellungen (Wharton et al. 1994, 106). Auf Basis dieser Informationsmaterialien versetzen sich Experten in die Rolle hypothetischer Benutzer und analysieren die notwendigen Aktionen zur Aufgabenbewältigung und identifizieren Schwachstellen sowie Verbesserungsvorschläge in der Benutzungsfreundlichkeit. Hierbei steht insbesondere die einfache Erlernbarkeit einer Softwareapplikation im Fokus der Tests (Wharton et al. 1994, 107; Dix et al. 2004, 321).

Ein weiteres Beispiel für einen Expertentest ist die *heuristische Evaluation* (Nielsen 1994, 25 ff.; Dix et al. 2004, 324 ff.). Bei diesem Testverfahren werden Gestaltungsvorschläge für eine Benutzungsschnittstelle von mehreren Experten auf Basis etablierter Heuristiken für eine gute Benutzungsfreundlichkeit von Softwareapplikationen untersucht (Dix et al. 2004, 324). Beispiele für etablierte Heuristiken sind die Usability Heuristiken von Jacob Nielsen (1993, 115 ff.) oder die „eight golden rules of interface design“ von Ben Shneiderman (Shneiderman/Plaisant 2010, 88 f.). Der Vorteil einer heuristischen Evaluation ist, dass diese mit einem vergleichsweise geringen Aufwand verbunden ist (Nielsen 1994, 25). Eine heuristische Evaluation ist prinzipiell in jeder Phase des Gestaltungsprozesses durchführbar; eignet sich aufgrund ihrer vergleichsweise geringen Voraussetzungen jedoch insbesondere in früheren Phasen (Dix et al. 2004, 324).

Beispiele für Nutzertest sind *fragebasierte Verfahren* (Dix et al. 2004, 348 ff.). Mögliche Varianten sind Interviews (Dix et al. 2004, 348) oder Fragebögen (Dix et al. 2004, 348 ff.; Sarodnick/Brau 2011, 181 ff.). *Interviews* stellen eine Möglichkeit dar, um die Meinung bzgl. bestimmter Designvorschläge direkt von potentiellen Nutzern abzufragen. Dabei wird i.d.R. vorab ein bestimmter Fragenkatalog erstellt, welcher zur groben Strukturierung des Interviews dient. Der Vorteil von Interviews ist, dass der Interviewer seine Fragen flexibel gemäß der Interviewsituation anpassen kann (Dix et al. 2004, 348). In Kapitel 10.2 der vorliegenden Arbeit wurde ein Interview mit potenziellen Nutzern des Entwicklungswerkzeuges durchgeführt, um einen konkreten Designvorschlag für die Benutzungsschnittstelle des Entwicklungswerkzeuges zu evaluieren. Die zweite Variante sind *Fragebögen*. Hierbei werden die Anzahl sowie die Formulierung der Fragen hingegen vorab festgelegt und können während der Befragung nicht verändert werden. Dadurch besitzen Fragebögen eine geringer Flexibilität als Interviews (Dix et al. 2004, 348). Aufgrund ihres vergleichsweise geringeren Aufwandes für die einzelne Befragung kann jedoch eine größere Nutzergruppe befragt werden (Dix et al. 2004, 349; Sarodnick/Brau 2011, 184). Zudem ist durch die einheitliche Befragung eine bessere Vergleichbarkeit der Antworten möglich (Dix et al. 2004, 349; Sarodnick/Brau 2011, 184).

Eine weitere Möglichkeit der Nutzertests sind *Usability-Tests* (Shneiderman/Plaisant 2010, 156 ff.; Sarodnick/Brau 2011, 162 ff.). Hierbei werden die Probanden bei der Lösung definierter Aufgabenstellungen beobachtet. Ziel ist es hierbei, bestimmte Hypothesen bzgl. des Umgangs mit der Softwareapplikation zu prüfen (Dix et al. 2004, 329 ff.; Sarodnick/Brau 2011, 164). In Bezug auf den Ausführungsort kann zwischen einer Feldstudie und einem Laborexperiment unterschieden werden (Dix et al. 2004, 327 ff.). In der *Feldstudie* erfolgt die Evaluation in der tatsächlichen Arbeitsumgebung des Probanden. Da hierbei explizit der Kontext berücksichtigt wird, können vergleichsweise realistische Ergebnisse erzielt werden. Zudem eigenen sich Feldstudien für Usability-Tests, welche über einen längeren Zeitraum durchgeführt werden müssen. Nachteilig sind die äußeren Störeinflüsse, welche die Testergebnisse beeinflussen können (Dix et al. 2004, 328 f.). In einem *Laborexperiment* erfolgt die Testdurchführung hingegen in einer kontrollierten Umgebung (Dix et al. 2004, 327 f.). Hierdurch können äußere Störeinflüsse vermieden werden. Zusätzlich ist die Nutzung von technischen Hilfsmitteln, beispielsweise zur Ton- oder Videoaufzeichnung, in einer Laborumgebung einfacher handhabbar (Dix et al. 2004, 327 f.). Abbildung 10-19 illustriert die beschriebenen Verfahren zur Evaluation der Benutzungsfreundlichkeit einer Softwareapplikation.

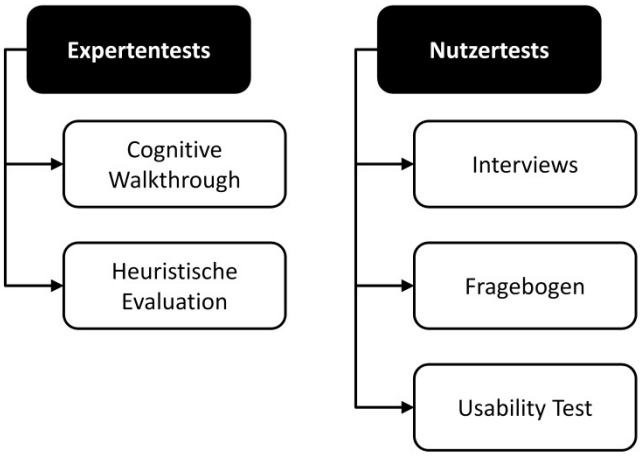

Abbildung 10-19: Evaluationsverfahren für die Benutzungsfreundlichkeit
Quelle: Eigene Darstellung

Für die Datenerhebung bei Nutzertests existieren unterschiedliche Protokollierungstechniken. Beispiele sind 1) manuelle Mitschriften mit „Papier und Bleistift“, 2) Tonaufnahmen, 3) Videoaufnahmen und 4) Logging, d.h. die automatisierte Aufnahme der Computerinteraktionen des Benutzers (Dix et al. 2004, 344 f.). *Manuelle Mitschriften* sind verhältnismäßig einfach umzusetzen; sind jedoch u.a. durch die Schreibgeschwindigkeit des Protokollanden limitiert und wird zusätzlich von der Interpretation des Protokollanden beeinflusst (Dix et al. 2004, 344). *Tonaufnahmen* können hingegen im Zuge der Auswertung mehrfach wiedergegeben werden; eignen sich jedoch nur für sprachbasierte Tests (Dix et al. 2004, 344). *Videoaufnahmen* haben den zusätzlichen Vorteil, dass auch non-verbale Aktionen der Probanden aufgezeichnet und anschließend analysiert werden können (Dix et al. 2004, 344). Durch entsprechende Softwarewerkzeuge ist Logging einfach umsetzbar. Nachteilig ist, dass zwar die durchgeführten Aktivitäten des Benutzers protokolliert werden, nicht jedoch die Gründe für die ausgeführten Aktivitäten (Dix et al. 2004, 344). Zusätzlich kann bei einem Logging über einen längeren Zeitraum ein großes Datenvolumen zum Problem werden (Dix et al. 2004, 344 f.). Aufgrund der spezifischen Vorzüge und Einschränkungen der genannten Protokollierungstechniken wird oftmals eine Kombination der genannten Techniken eingesetzt (Dix et al. 2004, 345). Neben den genannten Protokollierungstechniken stellt die *Think aloud Methode* (Dix et al. 2004, 343 f.; Shneiderman/Plaisant 2010, 161 f.) eine weitere Möglichkeit der Datenerhebung dar. Hierbei werden die Probanden gebeten, ihre Gedankengänge bei der Nutzung der zu testenden Softwareapplikation zu artikulieren (Dix et al. 2004, 343; Sarodnick/Brau 2011, 170). Die Beobachtung mittels der Think aloud Methode kann wertvolle Erkenntnisse zur aktuellen Nutzung der Softwareapplikation liefern und ist vergleichsweise einfach in der Durchführung (Dix et al. 2004, 343). Nachteilig ist, dass die Qualität und Menge der ausgesprochenen Gedanken verhältnismäßig stark vom Probanden abhängig ist (Dix et al. 2004, 343). Zudem senkt die Doppelbelastung aus Aufgabenbearbeitung und lautem Denken die Bearbeitungsgeschwindigkeit (Sarodnick/Brau 2011, 171).

10.8.6.2 Aufbau des Experimentes

Im vorherigen Kapitel wurden verschiedene Verfahren zur Untersuchung der Benutzungsfreundlichkeit von Softwareapplikationen vorgestellt. Da der vorliegende Prototyp des Entwicklungswerkzeuges bereits einen angemessen Reifegrad besitzt, d.h. insbesondere die gestellten funktionalen Anforderungen implementiert hat, eignet sich ein Nutzertest zur Prüfung der Evaluationsaspekte EA4 und EA7 sowie zur Prüfung der Benutzungsfreundlichkeit. Dies ermöglicht es, ausgewählte Anwendungsszenarien mit tatsächlichen Endbenutzern zu testen.

Zur Evaluation des Prototypen erscheint ein Usability Test mit definierten Aufgabenstellungen für die Probanden zweckmäßig. Hierdurch kann geprüft werden, ob die Probanden die gestellten Aufgabenstellungen ohne vorherige Einarbeitung in das Entwicklungswerkzeug lösen können (siehe Evaluationsaspekt EA4). Zudem kann der von den Probanden benötigte Zeitaufwand gemessen werden (siehe Evaluationsaspekt EA7). Um im Rahmen der Testausführung Schwachstellen der aktuellen Implementierung der Benutzungsschnittstelle feststellen zu können, sollte die Interaktion der Probanden mit dem Touch-Display des Smartphones protokolliert werden (Protokollierungstechnik: Logging). Um weitere Schwachstellen und Details zu den identifizierten Schwachstellen zu eruieren erscheint die Methode Think Aloud vorteilhaft. Um die anschließende Analyse der von den Probanden artikulierten Gedanken zu vereinfachen wird neben dem Logging auch eine Videoaufnahme mit Tonaufzeichnung des Experimentes durchgeführt.

10.8.6.2.1 Testszenarien

Zur Durchführung des Experimentes wurden zwei Aufgabenstellungen mit unterschiedlichem Schwierigkeitsgrad festgelegt. Unter dem Schwierigkeitsgrad wird in diesem Zusammenhang der Funktionsumfang der zu entwickelnden mobilen ERP-Applikation verstanden. Ein Teil der Probanden sollte die einfachere Aufgabenstellung durchführen; der andere Teil die schwierigere Aufgabenstellung. Ziel war es herauszufinden, ob der Funktionsumfang der zu entwickelnden mobilen ERP-Applikation eine überproportionale Auswirkung auf die beiden zu prüfenden Evaluationsaspekte EA4 und EA7 hat.

Testszenario A untersucht die einfachere Aufgabenstellung. Diese umfasst die Nutzung von zwei MBOTs, Kunde und Berichtswesen, mit zugehörigen MBOT-Methoden zum Anzeigen der jeweiligen MBO-Details. Testszenario B untersucht die komplexere Aufgabenstellung. Diese umfasst die Nutzung von drei MBOTs, Kunde, Berichtswesen und Produkte. Neben MBOT-Methoden zur Anzeige von MBO-Details sollte die entwickelte Applikation auch MBOT-Methoden zur Änderung von MBO-Daten enthalten. Um einen praktischen Kontext zu schaffen, wurden die Aufgabenstellungen in Form einer Applikationsanforderung eines fiktiven Unternehmens beschrieben. Die beiden Testszenarien sind im Anhang D bzw. Anhang E zu finden.

10.8.6.2.2 Aufbau des Labors

Um die Aktivitäten der Probanden über Video aufzuzeichnen, wurde eine Spiegelreflexkamera vom Typ EOS700D der Firma Canon verwendet. Über diese Kamera erfolgte die Bild- und Tonaufzeichnung der Experimente. Aufgrund der geforderten Unterstützung der beiden mobilen Betriebssysteme iOS und Android wurden die beiden folgenden Testgeräte eingesetzt:

- Galaxy S3 mini mit Android Version 4.1
- iPhone 4S mit iOS Version 5.1

Für das Logging der Benutzerinteraktionen mit den Smartphone-Displays existieren sowohl für iOS- als auch für Android-Smartphones unterschiedliche Werkzeuge. Im Rahmen der vorliegenden Arbeit wurde für die Bildschirmaufnahme des Galaxy S3 Mini das Werkzeug *MyPhoneExplorer*[65] in der Version 1.8.5 von FJ Software Development verwendet. Dabei handelt es sich um ein Verwaltungswerkzeug für Android-Smartphones, welches u.a. die Datensynchronisation zwischen Smartphone und PC sowie die Verwaltung der auf dem Smartphone gespeicherten Kontaktdaten über den PC ermöglicht. Zudem ist eine Funktionalität zur Fernsteuerung von Android-Smartphones über WLAN oder USB integriert, welche im Rahmen des Experimentes zur Aufnahme der Benutzerinteraktionen wurde. Für die Aufzeichnung der Benutzeraktionen mit dem iPhone Testgerät wurde das Werkzeug iTools[66] eingesetzt. Ähnlich zum MyPhoneExplorer ist der wichtigste Anwendungsfall für iTools die Dateisynchronisation und –verwaltung des iPhones über einen PC. Zudem lässt sich auch mit iTools eine Fernsteuerung eines iPhones über WLAN oder USB durchführen und damit die Bildschirmaufnahmen für das vorliegende Experiment anfertigen. Obgleich beide Werkzeuge eine Übertragung der Bildschirmmasken über WLAN unterstützen, wurde die Verbindungsvariante über USB gewählt. Grund hierfür war die Reduzierung des Risikos von Verbindungsabbrüchen bei der Übertragung.

Mit den genannten Werkzeugen ist eine Übertragung der aktuellen Bildschirmansicht auf dem Smartphone möglich. Da jedoch keine Aufzeichnung dieser Bildschirmansichten möglich ist, wird zusätzlich das Werkzeug BSR Screen Recorder[67] von bsrsoft in der Version 6 verwendet. Dieses erlaubt die Aufzeichnung eins PC-Bildschirmes und die anschließende Speicherung in gängige Videoformate wie beispielsweise Audio Video Interleave (AVI) oder Windows Media Video (WMV).

Bei jeder Durchführung des Experimentes waren drei Personen anwesend. Neben dem Probanden war ein Versuchsleiter anwesend, welcher eine kurze Einführung in den Ablauf des Experimentes gegeben hat. Zusätzlich war der Versuchsleiter Ansprechpartner bei Rückfragen. Neben dem Versuchsleiter war ein zusätzlicher Beobachter anwesend, welcher den Betrieb der technischen Hilfsmittel überwachte. Dazu zählten die Videoaufzeichnung über die

[65] http://www.fjsoft.at/de, zugegriffen am 10.1.2014
[66] http://itools-for-windows.en.softonic.com, zugegriffen am 10.1.2014
[67] http://www.bsrsoft.com, zugegriffen am 10.1.2014

Kamera sowie das Logging der Bildschirmmasken des Smartphones über die Werkzeuge MyPhoneExplorer bzw. iTools. Abbildung 10-20 illustriert den erläuterten Laboraufbau.

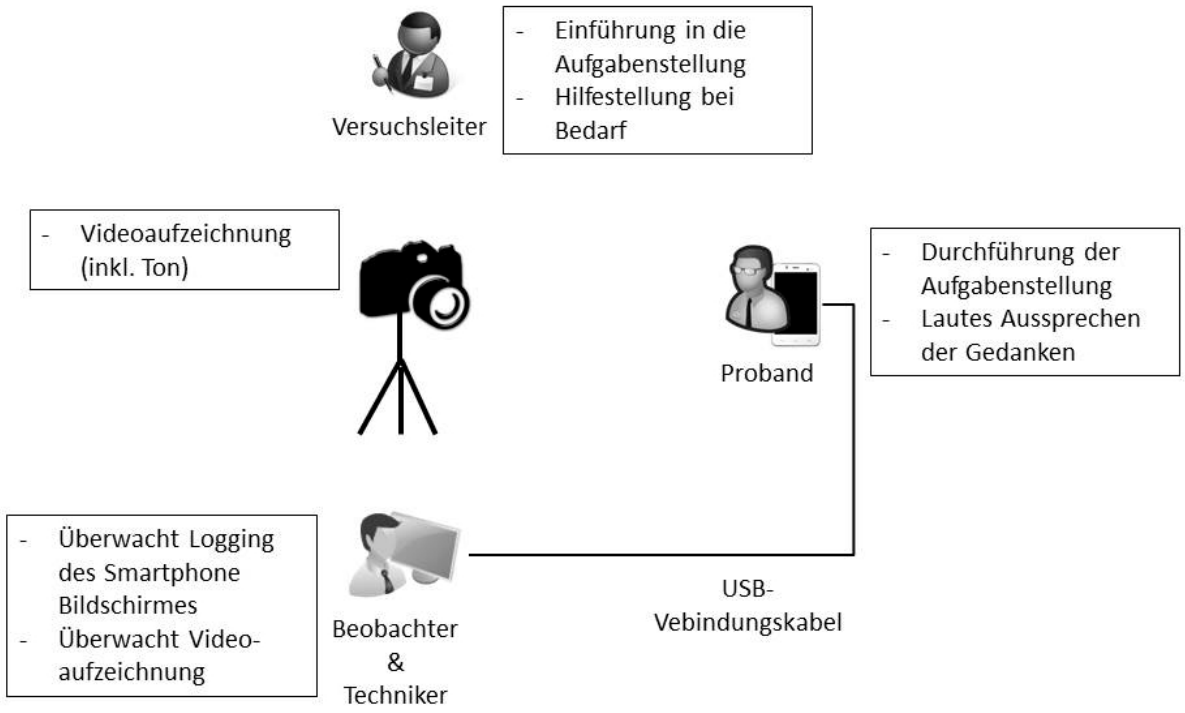

Abbildung 10-20: Aufbau des Labors
Quelle: Eigene Darstellung

10.8.6.2.3 Auswahl der Probanden

Bei der Auswahl der Probanden wurde darauf geachtet, dass es sich um Endbenutzer im Verständnis dieser Arbeit handelt, d.h. um Personen ohne Programmiererfahrung im Bereich mobiler Applikationen. Weiterhin wurde darauf geachtet, dass alle Probanden Erfahrung im Umgang mit Smartphones besaßen. Um das Testergebnis nicht zu beeinflussen, wurde zudem darauf geachtet, dass es sich bei den Probanden nicht um die gleichen Personen handelt wie bei der Zwischenevaluation (siehe Kapitel 10.2). Insbesondere aufgrund der letzten Einschränkung, konnte leider keine ausreichende Anzahl von Personen mit ERP-Hintergrundkenntnissen für die Teilnahme am Experiment gewonnen werden. Dies wurde dadurch gelöst, dass vor dem Test eine kurze Einführung in die im Rahmen des jeweiligen Testszenarios verwendete ERP-Terminologie durchgeführt wurde. Aufgrund des begrenzten Funktionsumfangs der fokussierten mobilen ERP-Applikationen, hatten alle Probanden anschließend ein hinreichendes Domänenverständnis für die Testdurchführung.

Insgesamt konnten 13 Probanden für die Teilnahme am Experiment gewonnen werden. Die Suche nach Probanden erfolgte ausschließlich über persönliche Kontakte. Unter den Probanden waren sieben weibliche und sechs männliche Teilnehmer. Der Mittelwert des Alters der Probanden liegt bei 30,3 Jahren. Insgesamt wurde die einfachere Aufgabenstellung A von acht Probanden bearbeitet; die komplexere Aufgabenstellung B wurde von den restlichen fünf Probanden bearbeitet. Die Zuteilung der Aufgabenstellung erfolgte hierbei durch den Wunsch der Probanden.

10.8.6.3 Ablauf des Experimentes

Jede Versuchsreihe startete mit einer kurzen Einführung des Probanden in das Ziel des Experimentes. Anschließend wurden die Aufgabenbeschreibungen in ausgedruckter Form überreicht. Nachdem sich die Probanden in die Aufgabenstellung eingelesen hatten, konnten offene Fragen zur Aufgabenstellung oder zum weiteren Ablauf des Experimentes gestellt werden.

Anschließend wurde die Videoaufzeichnung über die Kamera sowie die Bildschirmaufzeichnung des Smartphones über das Logging-Werkzeug gestartet. Während der Testdurchführung hatte der Versuchsleiter die Aufgabe, den Probanden in Problemfällen zu unterstützen und diesen zu motivieren seine Gedankengänge zu äußern. Dabei wurden die Probanden motiviert ihre Gedanken sowohl unmittelbar während der Aufgabendurchführung zu artikulieren, als auch in einer zusammenfassenden Form nach der Testdurchführung.

10.8.6.4 Auswertung der Experimente

Nach der Testdurchführung wurden die beiden Videoaufzeichnungen (Smartphone-Display und Videokamera) in einer Videoaufnahme zusammengeführt, synchronisiert und nebeneinander dargestellt. Dies war notwendig, um im Rahmen der Auswertung die Äußerungen der Probanden ihren Interaktionen mit dem Entwicklungswerkzeug zuordnen zu können. Abbildung 10-21 illustriert einen beispielhaften zusammengeführten Videoausschnitt.

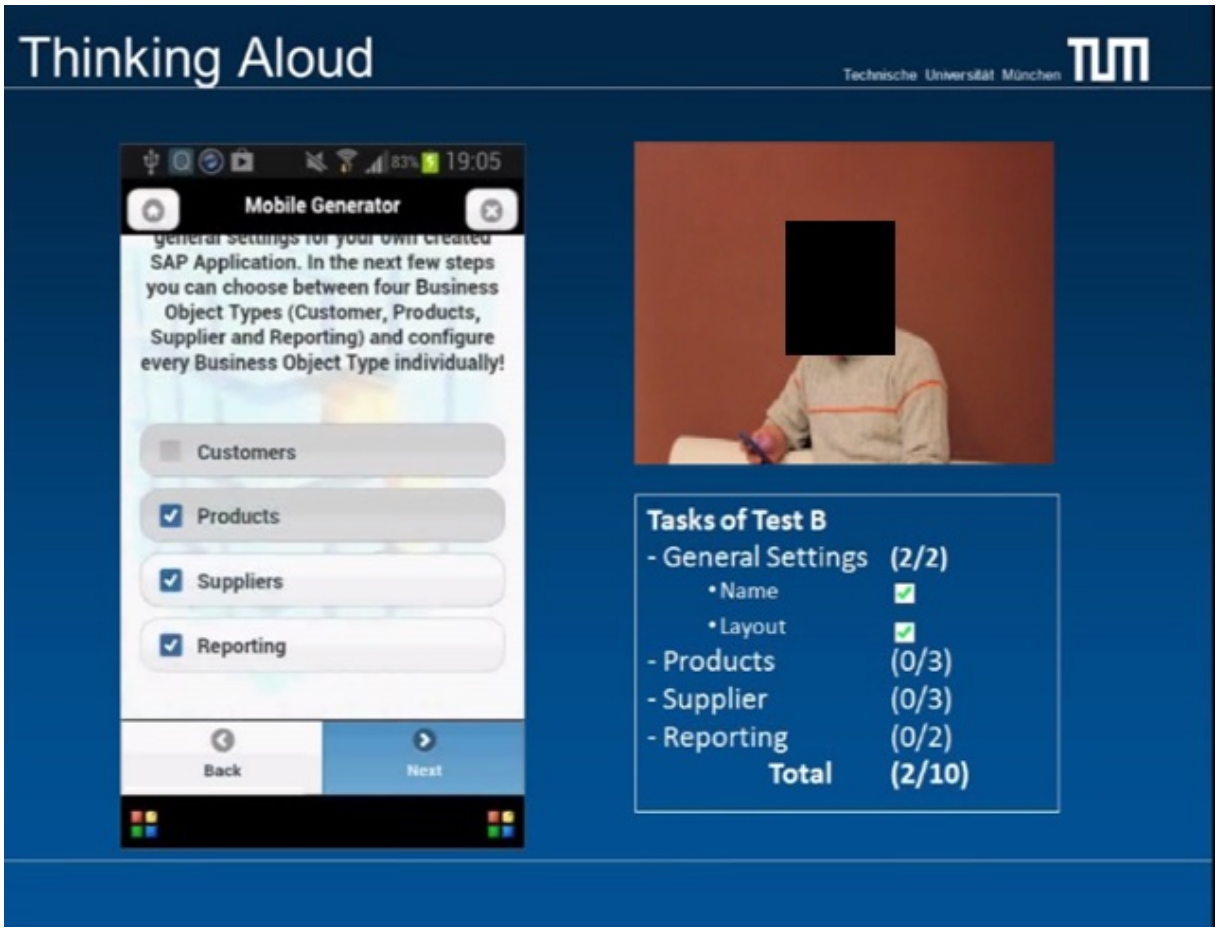

Abbildung 10-21: Ausschnitt einer synchronisierten Videoaufnahme
Quelle: Eigene Darstellung

Anschließend wurden die synchronisierten Videoaufnahmen von zwei Personen unabhängig voneinander ausgewertet. Hierbei wurden zum einen die benötigten Zeiten für die Aufgabenbewältigung gemessen. Zudem wurde der Erfüllungsgrad der Aufgabe über ein Punktesystem ermittelt. Im Testszenario A wurde eine vollstände Aufgabenerfüllung mit sieben Punkten bewertet. Hierbei wurden zwei Punkte für die erfolgreiche Eingabe eines Applikationsnamens und die Auswahl eines Applikationslayouts vergeben; drei Punkte für die korrekte Nutzung der MBOT-Methoden des MBOT „Kunde“ (jeweils ein Punkt pro geforderter MBOT-Methode) sowie zwei Punkte für die korrekte Nutzung des MBOT Berichtswesen. In Testszenario B wurde eine vollständige Aufgabenerfüllung mit 10 Punkten gewichtet. Davon wurden wiederrum zwei Punkte für die erfolgreiche Eingabe eines Applikationsnamens und die Auswahl eines Applikationslayouts vergeben; drei Punkte für die korrekte Auswahl der MBOT-Methoden des MBOT „Produkt“ (jeweils ein Punkt pro geforderter MBOT-Methode); drei Punkte für die korrekte Auswahl der MBOT-Methoden des MBOT „Lieferant“ (jeweils ein Punkt pro geforderter MBOT-Methode) und zwei Punkte für die korrekte Nutzung des MBOT „Berichtwesen“.

Zusätzlich wurden die im Rahmen der Interaktion der Probanden mit dem Entwicklungswerkzeug identifizierten Bedienprobleme von den auswertenden Personen niedergeschrieben, abstrahiert und in Kategorien zusammengefasst.

10.8.6.5 Evaluationsergebnisse

Im Folgenden werden die Ergebnisse der beiden Testszenarien beschrieben. Tabelle 10-1 enthält die Ergebnisse für das Testszenario A; Tabelle 10-2 enthält die Ergebnisse für das Testszenario B. Die Ergebnisdarstellung enthält eine fortlaufende *Nummerierung der Probanden*, die benötigte *Bearbeitungszeit* vom Startzeitpunkt der Aufzeichnung bis zur Generierung der Applikation, die erreichte *Punktezahl* bei der Aufgabenbewältigung sowie den aus der Punktezahl errechneten *Erfüllungsgrad* der Aufgabenstellung. Zusätzlich wird das jeweils verwendete Testgerät angeführt. Hierbei wird die Abkürzung „I“ für das iPhone und „A“ für das Android Testgerät verwendet.

Tabelle 10-1: Evaluationsergebnisse bzgl. Testszenario A
Quelle: Eigene Darstellung

	P1	P2	P3	P4	P5	P6	P7	P8	Ø
Bearbeitungszeit (in Minuten:Sekunden)	7:03	14:22	10:43	10:27	8:55	7:30	14:49	12:31	10:47
Punktezahl	7	7	7	7	7	6	7	6	6,75
Erfüllungsgrad (in %)	100	100	100	100	100	85	100	85	96,25
Testgerät	A	A	A	I	I	A	A	A	

Tabelle 10-2: Evaluationsergebnisse bzgl. Testszenario B
Quelle: Eigene Darstellung

	P9	P10	P11	P12	P13	Ø
Bearbeitungszeit (in Minuten:Sekunden)	19:58	6:18	11:58	11:29	15:03	12:57
Punktezahl	9	10	10	10	10	9,80
Erfüllungsgrad (in %)	90	100	100	100	100	98,00
Testgerät	I	A	A	A	A	

Die Ergebnisse des Testszenarios A haben gezeigt, dass die Bewältigung der gestellten Aufgabe im Durchschnitt 10 Minuten und 47 Sekunden in Anspruch nimmt. Die Aufgabenbewältigung des Testszenarios B nimmt hingegen im Durchschnitt 12 Minuten und 57 Sekunden in Anspruch. Auf Basis dieser Ergebnisse kann der Evaluationsaspekt EA4 positiv bestätigt werden. Hierbei ist zusätzlich anzumerken, dass die gemessene Bearbeitungszeit von zwei Faktoren negativ beeinflusst wird. Zum einen beansprucht die gewählte Think Aloud Metho-

de einen nicht unerheblichen Teil der Bearbeitungszeit, da die Probanden neben der eigentlichen Aufgabenbewältigung mit der Artikulation ihrer Gedankengänge beschäftigt sind. Zum anderen handelte es sich bei den durchgeführten Tests um den ersten Kontakt der Probanden mit der Anwendung. Dies war notwendig, um Evaluationsaspekt 7 im Rahmen des Experimentes zu prüfen. Die informellen Besprechungen und Versuche der Probanden nach dem offiziellen Test lassen jedoch vermuten, dass die Aufgabenbewältigung bei wiederholter Nutzung des Entwicklungswerkzeuges deutlich weniger Zeit in Anspruch nehmen wird. Insgesamt lagen lediglich die Bearbeitungszeiten der Probanden P9 und P15 bei der Bearbeitung der komplexeren Aufgabenstellung über den in Anforderung 7 geforderten 15 Minuten. Auf der anderen Seite hat Proband P1 nur 7 Minuten und 3 Sekunden für die Aufgabenbewältigung von Testszenario A und Proband P10 nur 6 Minuten und 18 Sekunden für die Bewältigung von Testszenario B benötigt. Die Ursache hierfür liegt evtl. im geübteren Umgang dieser Probanden mit Smartphone-Applikationen.

In Bezug auf den benötigen Trainingsaufwand hat das Testergebnis gezeigt, dass ein Großteil der Probanden die geforderten Applikationen mit vollem Funktionsumfang erstellen konnten. Insgesamt lag der durchschnittliche Erfüllungsgrad der Aufgabenstellung in Testszenario A bei 96,25%; in Testszenario B bei 98%. Diejenigen Probanden, welche keine volle Punktzahl erreichten, haben in allen drei Fällen lediglich eine gewünschte MBOT-Methode vergessen oder stattdessen eine nicht geforderte MBOT-Methode gewählt. Insgesamt kann damit gezeigt werden, dass ein Trainingsaufwand vor Nutzung des Entwicklungswerkzeuges nicht notwendig ist. Damit lässt sich Evaluationsaspekt 7 positiv bestätigen.

Im Folgenden werden die im Rahmen der Auswertung des Videomaterials identifizierten Schwachstellen in der Benutzungsfreundlichkeit des prototypisch implementierten Entwicklungswerkzeuges angeführt. Diese wurden in die folgenden Kategorien eingeteilt: Navigation, Hilfetexte, Farbgebung, Wertselektion, Auswahl des Applikationslayouts, Eingabefeld für den Applikationsnamen, Markierung von Signalbegriffen und Sprachauswahl.

Navigation

Die Schritt-für-Schritt Navigation wurde von einem Großteil der Probanden als intuitiv wahrgenommen. Sieben Probanden betonten den Vorteil der einfach gestalteten und selbsterklärenden Vorwärts- und Rückwärtsnavigation. Als besonders intuitiv wurde die grafische Visualisierung bei der Auswahl des Applikationslayouts empfunden. Einen Proband erwähnte nach kurzer Zeit, dass er das Muster bei der Selektion der MBOTs und zugehörigen MBOT-Methoden erkannt hat und als einfach erlernbar empfindet.

Ein Proband empfand die Positionierung der Navigationsschaltflächen am unteren Ende der Bildschirmfläche als suboptimal, da diese teilweise erst durch ein Verschieben des sichtbaren Bildschirmbereiches erreicht werden können.

Über einen „Homebutton“ ist es im Entwicklungsassistenten möglich, zur Startseite zurückzukehren. In diesem Fall werden jedoch alle bisher getätigten Entwicklungsaktivitäten gelöscht. Ein Proband hat den „Homebutton“ versehentlich getätigt. Aus diesem Grund hat der

Proband eine Sicherheitsabfrage mit einem Hinweis zur Löschung der bisher getätigten Einstellungen empfohlen.

Hilfetexte

In der aktuellen Umsetzung wurden Hilfetexte zur Erklärung der vom Endbenutzer durchzuführenden Aktivität bzw. zur Erläuterung der aktuellen Bildschirmmaske eingesetzt. Nahezu alle Probanden (12) empfanden die Umsetzung insgesamt zu textlastig. Bei der Analyse ist zudem aufgefallen, dass ein Großteil der Probanden die Textpassagen nicht vollständig gelesen hat. Ein genannter Verbesserungsvorschlag war, dass die Textpassagen gekürzt werden und ein ausführlicherer Hilfetext erst bei Bedarf eingeblendet werden kann.

Ein Proband empfand die textuelle Beschreibung hingegen als hilfreich für das Verständnis. Dieser Proband empfand die Erfolgsmeldungen im Text nach einer abgeschlossenen Aktivität hingegen als unnötig.

Farbgebung

In der Umsetzung des Prototypen wurden unterschiedliche Hintergrundfarben für die Konfiguration unterschiedlicher MBOTs verwendet. Dies empfanden fünf Probanden als störend. Insbesondere die Verwendung der Signalfarben „rot“ und „gelb“ hat einige Probanden irritiert. Die Probanden hatten beim Auftreten dieser Farben einen Bedienfehler angenommen.

Wertselektion

Zur Selektion von MBOTs und zugehöriger MBOT-Methoden werden im prototypisch implementierten Entwicklungswerkzeug Checkboxen und Buttongroups verwendet. Dabei wurde eine Vorauswahl vorgenommen. Diese Vorauswahl empfanden zwei Probanden als unerwünschte Bevormundung.

Bei einer möglichen Mehrfachauswahl, bspw. bei der Selektion von MBOTs, war für zwei Probanden die Mehrfachauswahl nicht ersichtlich. Sie hätten sich in diesem Fall einen kurzen erläuternden Text gewünscht.

Bei einer größeren Menge an Wertselektionsmöglichkeiten, bspw. bei der Selektion der anzuzeigenden Attribute eines verwendeten MBOT, befinden sich die Selektionsmöglichkeiten teilweise im nicht sichtbaren Bereich. Um diesen zu erreichen muss zunächst der sichtbare Bildschirmbereich durch „scrollen“ verschoben werden. Dies empfanden drei Probanden als störend bzw. nicht sofort erkennbar. Als Verbesserungsvorschlag wurde ein Hinweis auf die weiteren Selektionsmöglichkeiten, z.B. in textbasierter Form oder durch ein Symbol genannt.

Auswahl des Applikationslayouts

In der Aufgabenbeschreibung war kein Layout vorgegeben. Die Probanden konnten sich selbst für eines der angebotenen Applikationslayouts entscheiden. Drei Probanden fanden die angebotenen Layouts nicht ansprechend. Die Probanden würden eine größere Gestaltungs-

freiheit bei der Layoutauswahl bevorzugen. Genannte Beispiele waren die Einstellung der Schriftart und Schriftgröße.

Eingabefeld für den Applikationsnamen

In der aktuell implementierten Benutzungsschnittstelle des Entwicklungswerkzeuges wurde bewusst auf Textfelder verzichtet, da diese als fehleranfälliger eingestuft wurden als bspw. Auswahlfelder. Einzige Ausnahme ist das Textfeld zur Eingabe des Applikationsnamens. Hierbei wird geprüft, ob der Applikationsname den definierten Vorgaben entspricht. Eine dieser Vorgaben ist, dass der Applikationsname keine nummerischen Werte enthält. Diese Einschränkung wurde von einem Probanden bemängelt.

Zwei Probanden haben unter Nutzung des iPhone festgestellt, dass der voreingestellte Applikationsname „Name of Application“ im Eingabefeld bei der Selektion nicht markiert wird, was dessen anschließende Löschung erschwert.

Markierung von Signalbegriffen

In der aktuell implementierten Benutzungsschnittstelle des Entwicklungswerkzeuges wurden die Hauptfunktionalitäten der aktuellen Bildschirmmaske farblich hervorgehoben. Dies empfanden sechs Probanden als hilfreich. Der Vorteil liegt aus ihrer Sicht darin, dass der Hauptzweck der aktuellen Bildschirmmaske schnell erfasst werden kann. Hingegen fanden zwei Probanden die Markierung der Signalbegriffe als Schwachstelle, da dies nach ihrer Ansicht das aufmerksame Lesen des restlichen Textes erschwert.

Sprachauswahl

Die verwendeten Texte der Benutzungsschnittstelle wurden in englischer Sprache verfasst. Ein Wechseln der Sprache wurde nicht unterstützt. Dies empfanden sechs Probanden als verbesserungswürdig. Sie möchten die Sprache gerne flexibel umstellen können.

Neben diesen Schwachstellen der Benutzungsschnittstellen traten in den Experimenten weitere Schwachstellen auf, welche jedoch der Kompatibilität des verwendeten jQuery Mobile Frameworks mit den verwendeten Testgeräten zugeordnet werden können bzw. aus den Spezifika der Testgeräte resultieren. Beispielsweise kam es beim iPhone teilweise zu einem „Flackern“ des Bildschirmes. Dieser wurden von den Probanden als doppeltes Laden der Bildschirmmaske empfunden. In der iPhone-Tastatur existiert eine Schaltfläche „Öffnen“ als auch eine Schaltfläche „Enter“. In der aktuellen Implementierung des Entwicklungsassistenten führte nur die Schaltfläche „Öffnen“ zum Schließen der Tastatur und zur anschließenden Navigation zum nächsten Entwicklungsschritt. Die Taste „Enter“ besitzt hingegen keine Funktionalität. Dies hat bei den Experimenten teilweise zur Verwirrung geführt. Die Taste „Enter“ sollte in einer künftigen Implementierung ebenfalls mit der gleichen Funktionalität wie die Taste „Öffnen“ belegt werden.

10.8.7 Funktionaler Test zur Prüfung der Verbesserung des Antwortverhaltens

In Kapitel 9.3 wurde im Rahmen der Evaluation des Codegenerators ein unzureichendes Antwortverhalten der entwickelten mobilen ERP-Applikationen festgestellt. Diese machte sich dadurch bemerkbar, dass Ladevorgänge bei Datenabfragen vom SAP-ERP-System teilweise mehrere Sekunden beanspruchten. In Kapitel 10.4 wurde daher die Nutzung des proprietären, leistungsfähigeren RFC-Netzwerkprotokolls als Ersatz für das SOAP-Netzwerkprotokoll vorgeschlagen. In diesem Kapitel soll das Antwortverhalten bei Nutzung des RFC-Netzwerkprotokolls geprüft werden.

Zur Prüfung des Antwortverhaltens wurde eine beispielhafte Applikation mit folgenden Funktionalitäten mithilfe des Entwicklungswerkzeuges erstellt:

- Auflistung der beiden verfügbaren MBOTs „Kunde“ und „Lieferant“
- Auflistung aller MBOs des selektierten MBOT „Kunde“
- Auflistung aller MBOs des selektierten MBOT „Lieferant“
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten „Kunden“ MBO
- Anzeige der Attribute und zugehörigen Werte eines ausgewählten „Lieferant“ MBO

Im Rahmen der Messung des Antwortverhaltens wurde das Werkzeug Weinre[68] verwendet. Weinre stellt einen Debugger für HTML-basierte Applikationen auf mobilen Geräten dar. Zur eigentlichen Messung wurde der im Google Webbrowser Chrome integrierte Web Inspector[69] genutzt. Abbildung 10-22 illustriert eine beispielhafte Bildschirmansicht des Web Inspectors.

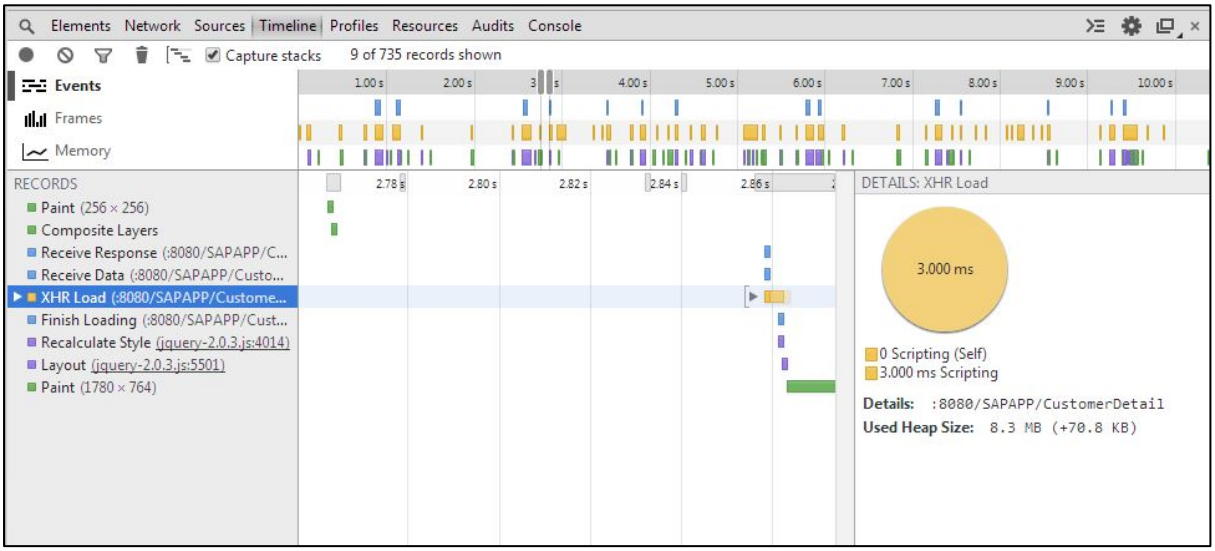

Abbildung 10-22: Beispielhafte Bildschirmansicht des Web Inspektors

Quelle: Eigene Darstellung

Zur Evaluation wird die Ladezeit der Daten vom SAP-ERP-System für die SAP-ERP-Abfragen (CustomerList, CustomerDetail, SupplierList und SupplierDetail) gemessen. Insgesamt wurde jede Abfrage zehnmal wiederholt. Da keine Zwischenspeicherung der abgefragten Daten im mobilen Gerät oder in der Tomcat Servlet Engine stattfindet, beeinflussen sich die einzelnen Abfragen nicht gegenseitig.

Die für den Test verwendete Datenbasis verfügte über zwölf gespeicherte Kundendatensätze und vier gespeicherte Lieferantendatensätze. Jedes Kunden-MBO verfügte über sieben Attribute und jedes Lieferanten-MBO über sechs Attribute. Tabelle 10-3 enthält die erzielten Messergebnisse, für die einzelnen Tests (T1 – T10) in Millisekunden.

Tabelle 10-3: Messergebnisse der Antwortzeiten
Quelle: Eigene Darstellung

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Ø
CustomerList	12	11	15	15	12	15	13	11	14	13	13,1
CustomerDetail	3	4	4	4	4	3	5	4	4	4	3,9
SupplierList	7	7	7	6	8	8	6	6	6	6	6,7
SupplierDetail	2	3	3	3	2	4	3	3	3	3	2,9

Insgesamt lässt sich feststellen, dass die durchschnittliche Ladezeit für die Abfrage der Kundeliste 13,1 ms und die Abfrage der Lieferantenliste 6,7 ms benötigte. Die Abfrage eines Kundendetails benötigte im Durchschnitt 3,9 ms und die eines Lieferantendetails 2,9 ms. Gegenüber der teilweise über 10 Sekunden benötigen Abfragen über die SOAP-Implementierung kann daher eine deutliche Verbesserung des Antwortverhaltens festgestellt werden.

10.9 Schlussfolgerungen der Evaluation

Insgesamt konnte durch die Evaluation gezeigt werden, dass das prototypisch implementierte Entwicklungswerkzeug die gestellten Anforderungen erfüllt. Im Rahmen des Nutzertestes konnte zudem gezeigt werden, dass auch potenzielle Endbenutzer die Bedienung des Werkzeuges ohne vorherigen Trainingsaufwand beherrschen und mobile ERP-Applikationen erstellen können. Auch die durchschnittliche Erstellungszeit einer mobilen ERP-Applikation lag in den Nutzertests unter den geforderten 15 Minuten. Zudem besteht die Vermutung, dass der benötige Zeitaufwand nach mehrmaliger Nutzung des Werkzeuges und ohne die Aufforderung die Gedankengänge laut zu äußern deutlich geringer ausfällt.

Im Nutzertest konnten einige Schwachstellen in der Benutzungsfreundlichkeit des Werkzeuges festgestellt werden. Vergleichsweise häufig wurde die als zu textlastig empfundenen Hilfestellungen genannt sowie die teilweise irreführende Farbgebung. Weiterhin müsste bei einer überarbeiteten Fassung des Prototypen eine bessere Bildschirmaufteilung gefunden werden, so dass keine Wertselektionen unterhalb des sichtbaren Bildschirmbereiches platziert sind oder die Platzierung für die Endbenutzer zumindest unmittelbar erkennbar ist.

11 Fazit und Ausblick

Das letzte Kapitel dieser Arbeit resümiert die erzielten Ergebnisse anhand der drei forschungsleitenden Fragestellungen (vgl. Kapitel 1.3) und fasst den wissenschaftlichen Beitrag der Arbeit zusammen. Darüber hinaus werden die mit den erzielten Ergebnissen einhergehenden Limitationen aufgezeigt und ein Ausblick auf weitere Forschungsmöglichkeiten gegeben.

11.1 Zusammenfassung der Ergebnisse und wissenschaftlicher Beitrag

Ziel der vorliegenden Forschungsarbeit war die Konzeption und prototypische Implementierung eines domänenspezifischen Entwicklungswerkzeuges zur Entwicklung mobiler ERP-Applikationen. Zielgruppe des Werkzeuges sind sogenannte Endbenutzer, d.h. Personen ohne professionelle Softwareentwicklungsexpertise im Bereich mobiler Applikationen. Aus dieser Zielsetzung wurden drei forschungsleitende Fragestellungen abgeleitet. Die nachfolgenden Ausführungen fassen die zentralen Ergebnisse zur Beantwortung dieser Fragestellungen zusammen. Anschließend wird der wissenschaftliche Beitrag der Arbeit zusammenfassend dargestellt.

Forschungsfrage 1: Welche gemeinsamen Charakteristiken besitzen mobile ERP-Applikationen und welche Herausforderungen existieren aktuell bei ihrer Entwicklung?

Zur Beantwortung der ersten Forschungsfrage wurden in Kapitel 2 zunächst zentrale Begriffe aus dem Umfeld mobiler ERP-Applikationen und deren Entwicklung definiert und abgegrenzt. Um domänenspezifische Charakteristiken zu identifizieren, beschäftigte sich Kapitel 3 mit der Untersuchung gemeinsamer Charakteristiken mobile ERP-Applikationen. Hierzu wurden zunächst gemeinsame Charakteristiken mobiler ERP-Applikationen durch eine Studie wissenschaftlicher und praxisnaher Literaturquellen ermittelt. Diese wurden anschließend durch weitere Charakteristiken auf Basis einer Untersuchung von existierenden mobilen ERP-Applikationen ergänzt. Eine wesentliche Erkenntnis dieser Untersuchungen war die Unterscheidung unterschiedlicher Applikationsschwerpunkte mobiler ERP-Applikationen. Dabei wurde zwischen Produktivitäts-, Prozesszentrierten-, Kunden- und Analytische Applikationen unterschieden. Zudem haben die Untersuchungen gezeigt, dass sich die Funktionalitäten mobiler ERP-Applikationen größtenteils auf die Anzeige und Bearbeitung sogenannter Business Objekte des ERP-Systems konzentrieren und die zugehörigen Benutzungsschnittstellen, insbesondere die der Produktivitätsapplikationen, ähnlich realisiert sind. Durch diese Erkenntnis konnte die Idee eines Bausteinkonzeptes zur Entwicklung mobiler ERP-Applikationen bekräftigt werden.

Um aktuelle Herausforderungen bei der Entwicklung mobiler ERP-Applikationen zu ermitteln wurde in Kapitel 4 eine empirische Untersuchung durchgeführt. Hierzu wurden zwei Befragungen mit unterschiedlichen Schwerpunkten durchgeführt. Schwerpunkt der ersten Befragung waren die Potentiale und Herausforderungen der modellgetriebenen Entwicklung, als wichtigen Lösungsansatz der Endbenutzer-Entwicklung, zur Umsetzung mobiler Unterneh-

mensapplikationen. Als zentrale Herausforderungen konnten die aktuell hohe Heterogenität und die limitierten Ressourcen bei der Entwicklung mobiler Unternehmensapplikationen identifiziert werden. Die hohe Heterogenität wird bedingt durch die unterschiedlichen, am Markt befindlichen mobilen Geräte sowie der zugehörigen mobilen Betriebssysteme. Bei der Entwicklung von mobilen Applikationen für die unterschiedlichen Geräte müssen unterschiedliche Programmiersprachen, Entwicklungsumgebungen und Entwicklungsbibliotheken eingesetzt werden. Es hat sich gezeigt, dass die verfügbare Zeit in Entwicklungsprojekten für mobile Unternehmensapplikationen oftmals knapp bemessen ist und die Entwickler daher den Einarbeitungsaufwand in neue Technologien möglichst vermeiden möchten. Zudem wünschen sich die Entwickler zügig erste vorzeigbare Resultate ihre Entwicklungsanstrengungen, welche sie idealerweise mit ihren Auftraggebern diskutieren können.

Schwerpunkt der zweiten Befragung waren die Potentiale und Herausforderungen der Endbenutzer-Entwicklung zur Umsetzung mobiler Unternehmensapplikationen. Das Ergebnis hat verdeutlicht, dass insgesamt ein Potential für die Endbenutzer-Entwicklung mobiler ERP-Applikationen besteht. Als zentrale Herausforderungen konnte ein zügiger Entwicklungsvorgang und ein geringer Einarbeitungsaufwand identifiziert werden. Dies wurde dadurch begründet, dass die Softwareentwicklung nicht das Kernaufgabengebiet der Endbenutzer darstellt. Daher sollte der Einarbeitungs- und Entwicklungsaufwand in ein zugehöriges Werkzeug so weit wie möglich reduziert werden. Als zentrale Lösungsideen zur Bewältigung dieser Herausforderung wurden u.a. eine bewusste Begrenzung der Funktionalitäten des Entwicklungswerkzeuges und ein schrittweiser, geführter Entwicklungsprozess vorgeschlagen.

Forschungsfrage 2: Welche Anforderungen hinsichtlich der Gestaltung eines Endbenutzer-Entwicklungswerkzeuges für mobile ERP-Applikationen ergeben sich aus den identifizierten Charakteristiken und Herausforderungen und welche konzeptuelle Softwarearchitektur und technische Basis ist geeignet um diese Anforderungen umzusetzen?

Zur Beantwortung von Forschungsfrage 2 wurden in Kapitel 5 aus den Ergebnissen der Forschungsfrage 1 sowohl Anforderungen an die zu entwickelnden mobilen ERP-Applikationen, als auch Anforderungen an das zugehörige Endbenutzer-Entwicklungswerkzeug abgeleitet. Als übergeordnete Anforderungen dienten ein möglichst geringer Einarbeitungsaufwand, ein zügiger Entwicklungsvorgang und die Unterstützung der beiden aktuell dominierenden mobilen Betriebssysteme iOS und Android.

Um die aktuelle Umsetzung der identifizierten Anforderungen in existierenden Entwicklungswerkzeugen für mobile Applikationen zu untersuchen, wurden drei Entwicklungswerkzeuge für mobile ERP-Applikationen und drei Endbenutzer-Entwicklungswerkzeuge analysiert. Die Untersuchung der ersten Gruppe hat gezeigt, dass vor der Nutzung dieser Werkzeuge zunächst ein größerer Installations- und Konfigurationsaufwand notwendig ist. Aufgrund des hohen Funktionsumfangs ist zudem mit einem hohen Einarbeitungsaufwand zu rechnen. Zudem bieten die untersuchten Werkzeuge einen vergleichsweise hohen Freiheitsgrad bei der Entwicklung, was das Fehlerpotential bei der Entwicklung erhöht. Einen anderen Ansatz

verfolgen die untersuchten Endbenutzer-Entwicklungswerkzeuge. Diese leiten den Endbenutzer schrittweise durch den Entwicklungsprozess und bieten lediglich vordefinierte Entwicklungsoptionen auf Basis einer formularbasierten Interaktionstechnik an. Hierdurch können mobile Applikationen ohne Einarbeitungsaufwand in wenigen Minuten erstellt werden. Aktuell unterstützen die vorgestellten Werkzeuge lediglich gängige Internetdienste wie Twitter oder Facebook als Datenquellen. Zudem sind die Entwicklungswerkzeuge selbst nicht auf einem Smartphone lauffähig und benötigen stattdessen einen Desktop-PC.

Um das gewünschte Endbenutzer-Entwicklungswerkzeug für mobile ERP-Applikationen zu konzipierten wurden in Kapitel 7 das grobe Architekturkonzept des Werkzeuges sowie das Vorgehen zur Umsetzung des Architekturkonzeptes beschrieben. Kernidee der Beschreibung war ein modellbasierter Entwicklungsansatz, bei welchem ein Endbenutzer eine mobile ERP-Applikation auf Basis domänenspezifischer Bausteine spezifiziert. Die Applikationsspezifikationen werden anschließend durch einen geeigneten Codegenerator in lauffähige mobile ERP-Applikationen überführt. Die genannten Bausteine sind Teil einer domänenspezifischen Sprache für mobile ERP-Applikationen, welche neben den Bausteinen auch deren Verknüpfungsregeln festlegt. In Bezug auf das Vorgehen wurden drei Design-Zyklen beschreiben. In einem ersten Designzyklus wurde die domänenspezifische Sprache sowie der Codegenerator konzipiert und prototypisch umgesetzt. In einem zweiten Designzyklus wurde ein erster Entwurf der Benutzungsschnittstelle des Entwicklungswerkzeuges konzipiert und evaluiert. In einem dritten Designzyklus wurde ein funktionsfähiger Prototyp des Entwicklungswerkzeuges mit Integration zum Codegenerator konzipiert und umgesetzt. Als abschließenden Evaluationsschritt wurde ein Laborexperiment mit repräsentativen Endbenutzern durchgeführt.

Kapitel 8 beschreibt die Gestaltung der domänenspezifischen Sprache. Die hierbei angewendete Vorgehensweise orientierte sich an den vorgeschlagenen Phasen von Mernik et al. (2005, 320 ff.) zur Entwicklung domänenspezifischer Sprachen. Dieses umfasst die fünf Phasen: Entscheidung, Analyse, Design, Implementierung und Einsatz. Als Resultat wurde eine domänenspezifische Sprache für mobile ERP-Applikationen basierenden auf den *funktionalen Konzepten* „Mobile Business Object Type“, „Mobile Business Object“, Attribut“ und „Methode“ und den *Bedienkonzepten* „Sortierte Listen-Anzeige“, „Formularbasierte MBO-Anzeige“ und „Formularbasierte MBO-Bearbeitung“ sowie zugehöriger Domänenregeln vorgestellt. Als prototypische Umsetzung der domänenspezifischen Sprache wurde anschließend eine XML-basierte Notationsform entwickelt.

In Kapitel 9 wurde der Architekturentwurf des Codegenerators beschreiben. Hierzu wurde zunächst ein HTML5-basiertes Zielformat der zu entwickelnden mobilen ERP-Applikationen begründet. Anschließend wurde ein Transformationswerkzeug auf Basis von XSLT vorgestellt, welches aus den XML-basierten Applikationsspezifikationen durch geeignete Transformationsregeln entsprechende HTML5-basierte, lauffähige Applikationen generiert. Die Transformationsregeln wurden hierbei auf Basis wiederverwendbarer Codeschablonen umgesetzt.

Kapitel 10 fokussiert die Konzeption der Benutzungsschnittstelle des Entwicklungswerkzeuges. Hierzu wurde zunächst ein erster Entwurf der Benutzungsschnittstelle konzipiert und von

potentiellen Endbenutzern evaluiert. Die Verbesserungsvorschläge wurden anschließend verwendet, um eine optimierte Benutzungsschnittstelle für das Entwicklungswerkezug zu gestalten. Ergebnis ist eine webbasierte Benutzungsschnittstelle auf Basis des HTML5-Frameworks jQuery Mobile. Das Werkzeug setzt einen schrittweisen, formularbasierten Entwicklungsprozess um. Die Integration mit dem SAP-ERP-System erfolgt über den Aufruf entsprechender BAPIs durch den SAP Java Connector. Die Funktionalität des Entwicklungswerkzeuges ist durch Java Servlets umgesetzt, welche als Laufzeitumgebung eine Apache Tomcat Servlet Engine nutzen.

Forschungsfrage 3: Welche Implikationen hinsichtlich der Weiterentwicklung und Nutzung des vorgestellten Entwicklungswerkzeuges ergeben sich aus dessen praktischer Nutzung durch Endbenutzer?

Neben der Evaluation einzelner Anforderungen, wurde in Kapitel 10 ein umfangreicheres, kontrolliertes Experiment mit potentiellen Endbenutzern durchgeführt. Hierbei mussten die Probanden vorgegebene Funktionalitäten einer mobilen ERP-Applikationen mit dem prototypisch implementierten Entwicklungswerkzeug umsetzen. Um die Ursachen für Bedienprobleme besser verstehen zu können, wurden die Probanden durch Anwendung der Think aloud Methode angehalten ihre Gedanken zu artikulieren. Die einzelnen Experimente wurden über eine Videokamera aufgezeichnet und gemeinsam mit den aufgezeichneten Bildschirmaufnahmen des Entwicklungswerkzeuges ausgewertet.

Insgesamt hat sich gezeigt, dass die Probanden zügig mit dem Werkzeug umgehen konnten. Alle Probanden waren in der Lage die gewünschten Applikationsfunktionalitäten ohne vorherige Einarbeitung in das Werkzeug umzusetzen. Zudem dauerte der gesamte Entwicklungsvorgang trotz erstem Kontakt mit dem Werkzeug nur wenige Minuten. Auf Basis dieser Ergebnisse kann die konzipierte Softwarearchitektur als geeignet eingestuft werden, um die gestellten Anforderungen umzusetzen. Neben diesen positiven Ergebnissen konnten eine Reihe von Verbesserungs- und Erweiterungsvorschläge identifiziert werden. Diese beziehen sich größtenteils auf das Layout der Benutzungsschnittstelle des Entwicklungswerkzeugs und die Fehlerbehandlung.

Wissenschaftlicher Beitrag

Der wissenschaftliche Beitrag dieser Arbeit besteht zum einen in der systematischen Untersuchung mobiler ERP-Applikationen. Die Literaturstudie hat gezeigt, dass bisherige Forschungsarbeiten keine Charakterisierung von mobilen ERP-Applikationen vornehmen. Als Ergebnis der Untersuchung in dieser Arbeit ist eine Liste von 12 Charakteristiken mobiler ERP-Applikationen entstanden.

Zum anderen besteht der wissenschaftliche Beitrag in der Konzeptionierung, prototypischen Implementierung und Evaluierung eines EUD-Ansatzes. In diesem Zusammenhang sind folgende Wissensbausteine entstanden:

- Eine Liste von 13 Anforderungen an ein EUD-Werkzeug für mobile ERP-Applikationen.

- Eine domänenspezifische Sprache für mobile ERP-Applikationen und zugehöriger Umsetzung in Form einer XML-basierten Notation.
- Ein Konzept für einen Codegenerator zur Transformation von mit Hilfe der domänenspezifischen Sprache spezifizierten Applikationen in lauffähige mobile ERP-Applikationen.
- Ein Konzept für ein EUD-Werkzeug mit einer Benutzungsschnittstelle auf einem Smartphone.
- Eine lauffähige prototypische Umsetzung des konzipierten EUD-Werkzeuges mit Integration zum Codegenerator.
- Eine umfangreiche Evaluation in Form von funktionalen Tests, statischen Analysen sowie einem Nutzertest, welche die Tauglichkeit der erarbeiteten Konzepte belegt.

Tabelle 11-1 stellt die im Rahmen der Arbeit vorgestellten Charakteristiken mobiler ERP-Applikationen, die Anforderungen an das Endbenutzer-Entwicklungswerkzeug, die wiederkehrenden Funktionalitäten mobiler ERP-Applikationen, die funktionalen Konzepte und Bedienkonzepte der entwickelten domänenspezifischen Sprache sowie die verwendeten Evaluationsaspekte zusammenfassend dar.

Tabelle 11-1: Zusammenfassende Darstellung wesentlicher Aspekte der Forschungsarbeit
Quelle: Eigene Darstellung

Charakteristik	**Beschreibung**
C1	Daten und Funktionalitäten stammen aus dem ERP-System
C2	Stellen die Präsentationsschicht in einem mobilen ERP-Applikationsszenario dar
C3	Besitzen unterschiedliche Applikationsschwerpunkte
C4	Nutzen unterschiedliche Implementierungsvarianten
C5	Unterschiedliche Verfahren zur Datenspeicherung werden unterstützt
C6	ERP-Daten und -Funktionalitäten werden reduziert
C7	Nutzung einer mobilen Unternehmensplattform
C8	Fokussierung auf ausgewählte Business Objekt Typen
C9	Bereitstellung ähnliche Funktionalitäten
C10	Nutzung ähnliche Interaktionsmuster
C11	Nutzung ähnlicher Benutzungsschnittstellen
C12	Eingeschränkte Parametrisierungsmöglichkeiten
Anforderung	**Beschreibung**
A1	Die entwickelten mobilen ERP-Applikationen müssen auf den beiden mobilen Betriebssystemen Android und iOS lauffähig sein.
A2	Die entwickelten mobilen ERP-Applikationen müssen auf die Daten eines SAP-ERP-Systems über BAPI-Schnittstellen zugreifen.
A3	Das Entwicklungswerkzeug muss die Erstellung von Produktivitätsapplikationen unterstützen mit den folgenden Funktionalitäten für die bereitgestellten Business Objekt Typen bzw. Business Objekte unterstützen:

	• Auflistung aller verfügbaren Business Objekt Typen • Auflistung aller Business Objekte eines selektierten Business Objekt Typs • Anzeige der Attribute und zugehörigen Werte eines ausgewählten Business Objekts • Erzeugung eines neuen Business Objekts • Änderung eines existierenden Business Objekts • Löschung eines existierenden Business Objekts
A4	Die Erstellung mobile ERP-Applikationen mit Hilfe des Entwicklungswerkzeuges muss ohne Trainingsaufwand möglich sein.
A5	Das Entwicklungswerkzeug muss den Endbenutzer schrittweise durch den Entwicklungsvorgang führen.
A6	Das Entwicklungswerkzeug muss die Auswahl unterschiedlicher Layoutschablonen zur Umsetzung der Corporate Identity anbieten.
A7	Der Entwicklungsvorgang einer mobilen ERP-Applikation darf maximal 15 Minuten betragen.
A8	Das Entwicklungswerkzeug bietet keine Konfigurationsmöglichkeit für die Verbindung zu einem SAP-ERP-System für den Endbenutzer.
A9	Das Entwicklungswerkzeug bietet nur eingeschränkte Möglichkeiten zur Gestaltung der Benutzungsschnittstelle einzelner Bildschirmmasken einer mobilen ERP-Applikation. Beispielsweise wird das Hinzufügen oder Positionieren einzelner Anzeige- und Bedienelemente nicht unterstützt.
A10	Das Entwicklungswerkzeug stellt wiederverwendbare Schablonen für die Umsetzung der Benutzungsschnittstellen für die unter Anforderung 3 beschriebenen Funktionalitäten bereit.
A11	Das Entwicklungswerkzeug sollte auf einem Smartphone genutzt werden und auf den beiden mobilen Betriebssystemen Android und iOS lauffähig sein.
A12	Das Entwicklungswerkzeug sollte erweiterbar sein, so dass bei Bedarf mobile ERP-Applikationen neben iOS und Android auch für weitere mobile Betriebssysteme entwickelt werden können.
A13	Das Entwicklungswerkzeug muss die Verteilung entwickelter mobiler ERP-Applikationen an weitere Mitarbeiter unterstützen.
Funktionalität	**Beschreibung / Beispiele**
Auswahl eine Business Objekt Typs	Beispielsweise wird in der Applikation einer der folgenden Menüpunkte gewählt: Kunden, Produkte oder Lieferanten.
Auflistung aller Business Objekte	Beispielsweise werden alle Lieferanten oder alle Mitarbeiter eines Unternehmens aufgelistet.
Sortierung einer Liste von Business Objekten nach einem bestimmten Attribut	Beispielsweise werden alle Mitarbeiter nach ihrem Nachnamen oder alle Produkte nach ihrem Preis sortiert aufgelistet.
Gruppierung einer Liste von Business Objekten nach einem bestimmten Attribut	Beispielsweise werden alle Produkte nach ihrer Produktgruppe oder ihrer Gewichtsklasse sortiert aufgelistet.
Suche nach einem	Beispielsweise wird ein Mitarbeiter nach seinem Nachnamen oder seiner

Business Objekt über ein bestimmtes Attribut	Personalnummer gesucht.
Filterung einer Liste von Business Objekten nach einem Attribut	Beispielsweise werden nur Mitarbeiter einer ausgewählten Abteilung aufgelistet oder nur Kunden aus einer ausgewählten Region.
Anzeige der Attribute und zugehörigen Werte eines ausgewählten Business Objekts	Beispielsweise wird eine bestimmte Produktinstanz ausgewählt. Daraufhin werden z.B. u.a. der Produktname, die Produktkategorie, seine Farbe, sein Gewicht und sein Preis angezeigt.
Navigation zu einem verknüpften Business Objekt eines anderen Business Objekt Typen	Beispielsweise kann von einer ausgewählten Kunden-Instanz zur Anzeige der Aufträge des Kunden navigiert werden.
Erzeugung eines neuen Business Objekts	Beispielsweise wird ein neuer Kundenauftrag angelegt oder eine neue Schadensmeldung erzeugt.
Änderung eines existierenden Business Objekts	Beispielsweise wird die Telefonnummer oder die Büronummer eines Mitarbeiters geändert.
Löschung eines existierenden Business Objekts	Beispielsweise verlässt ein Mitarbeiter das Unternehmens und wird demzufolge aus dem Mitarbeiterverzeichnis gelöscht.
Hinzufügen eines ausgewählten Business Objekts zu einer Favoritenliste	Beispielsweise wird ein häufig aufgerufenes Mitarbeiterprofil in die Favoritenliste aufgenommen, damit es zukünftig schneller zugreifbar ist.
Funktionale Konzepte	**Beschreibung**
Mobile Business Object Type	Ein Mobile Business Object Type (MBOT) repräsentiert einen in Bezug auf ein mobiles Szenario reduzierten Objekttyp aus dem ERP-System.
Mobile Business Object	Ein MBO repräsentiert eine Instanz eines MBOT.
Attribut	Ein Attribut repräsentiert ein Datenfeld in einem MBOT. Ein Attribut besitzt einen Datentyp. Bei der Instanziierung eines MBO kann zudem ein Wert zugeordnet werden.
Methode	Eine Methode repräsentiert die Funktionalität eines MBOT. Durch Methoden können beispielsweise Attributwerte von MBOs gelesen und manipuliert werden.
Bedienkonzepte	**Beschreibung**
Sortierte Listen-Anzeige	Auflistung von MBOTs oder MBOs in einer bestimmten Reihenfolge.
Formularbasierte MBO-Anzeige	Anzeige der Attribute und zugehörigen Werte eines selektierten MBO.
Formularbasierte MBO-Bearbeitung	Bearbeitung ausgewählter Attributwerte eines selektierten MBO.
Evaluationsaspekt	**Beschreibung (+ Evaluationsmethode)**
EA1	Unterstützte mobile Betriebssysteme der entwickelten mobile ERP-Applikationen (Funktionaler Test)
EA2	Datenkommunikation der entwickelten mobile ERP-Applikationen mit einem SAP-ERP-System (Funktionaler Test)

EA3	Funktionsumfang der mobilen ERP-Applikationen (Funktionaler Test)
EA4	Trainingsaufwand zur Nutzung des Entwicklungswerkzeuges (Kontrolliertes Experiment)
EA5	Schrittweiser Entwicklungsprozess (Statische Analyse)
EA6	Nutzung von Layoutschablonen für mobile ERP-Applikationen (Funktionaler Test)
EA7	Dauer des Entwicklungsvorganges (Kontrolliertes Experiment)
EA8	Konfigurationsmöglichkeit für die SAP-ERP Verbindungseinstellung (Statische Analyse)
EA9	Gestaltungsspielraum bei der Entwicklung der Benutzungsschnittstelle der mobilen ERP-Applikationen (Statische Analyse)
EA10	Nutzung wiederverwendbarer Codeschablonen (Statische Analyse)
EA11	Unterstützte mobile Betriebssysteme des Entwicklungswerkzeuges (Funktionaler Test)
EA12	Erweiterbarkeit des Entwicklungswerkzeuges hinsichtlich der Unterstützung weiterer mobiler Betriebssysteme (Statische Analyse)
EA13	Verteilungsfunktion für die entwickelten mobilen ERP-Applikationen (Funktionaler Test)

11.2 Limitationen und Ausblick

In der vorliegenden Arbeit wurde ein Entwicklungswerkzeug zur Umsetzung mobiler ERP-Applikationen durch Endbenutzer konzipiert, dessen Tauglichkeit durch eine ausführliche Evaluation demonstriert werden konnte. Jedoch wurde hierbei von einer Reihe von Limitationen ausgegangen, die Raum für weiteren Forschungsbedarf bieten.

Das konzipierte Entwicklungswerkzeug fokussiert sich auf die Umsetzung von mobilen ERP-Applikationen vom Typ Produktivitätsapplikationen (siehe Kapitel 3.2), welche im ERP-Umfeld häufig eingesetzt werden. Produktivitätsapplikationen sind auf die Durchführung von kurzweiligen Aktivitäten, wie bspw. Datenabfragen oder Genehmigungen, fokussiert. Sie nutzen gängige, formularbasierte Anzeige- und Bedienelemente und einfache Navigationsstrukturen. Durch diese Fokussierung war es möglich, den Funktionsumfang des Entwicklungswerkzeuges zu begrenzen und einen intuitiven Entwicklungsvorgang umzusetzen. Mobile ERP-Applikationen mit abweichenden Funktionalitäten oder Layoutanforderungen können mit dem vorgestellten Werkzeug nicht umgesetzt werden. Daher wäre eine interessante, anknüpfende Forschungsfrage, inwieweit andere Applikationstypen mit dem vorgestellten Entwicklungswerkzeug umgesetzt werden können bzw. welche Anpassungen hierfür notwendig wären.

Der durchgeführte Nutzertest hat gezeigt, dass potentielle Endbenutzer mit dem Entwicklungswerkzeug ohne Einarbeitungsaufwand mobile ERP-Applikationen erstellen können. Für alle Probanden war der Nutzertest der erste Kontakt mit dem Entwicklungswerkzeug. Es besteht jedoch die Annahme, dass sich die Anforderungen an das Werkzeug bei steigender

Nutzungshäufigkeit ändern. Daher besteht weiterer Forschungsbedarf hinsichtlich der optimalen Unterstützung fortgeschrittener Endbenutzer. Ein möglicher Anwendungsfall für fortgeschrittenere Endbenutzer ist die Entwicklung von Datenservices. Das vorgestellte Entwicklungswerkzeug fokussiert sich auf die Entwicklung der Benutzungsschnittstelle der mobilen ERP-Applikationen. Der Datenaustausch mit dem SAP-ERP-System erfolgt in der prototypischen Implementierung über Java Servlets, welche ihrerseits den SAP JCo zum Aufruf der BAPIs im SAP-ERP-System nutzen. Die Kommunikation mit BAPIs erfolgt über existierende Servlets. Endbenutzer können mit dem vorgestellten Werkzeug selbst keine neuen Servlets entwickeln; diese müssen von professionellen Softwareentwicklern bereitgestellt werden. Weitere Forschung sollte daher Möglichkeiten finden, wie auch Datenservices für ERP-Systeme von Endbenutzer entwickelt werden können.

Schwerpunkt der vorgestellten Softwarearchitektur ist die Entwicklung von mobilen ERP-Applikationen. Für den Einsatz im Unternehmensumfeld sind jedoch auch Betriebsaspekte wie die Verwaltbarkeit der entwickelten mobilen ERP-Applikationen oder Sicherheitsaspekte entscheidend. Daher besteht weiterer Forschungsbedarf hinsichtlich der Betriebs- und Sicherheitsaspekte der vorgestellten Lösung. Es wäre u.a. zu klären inwieweit das vorgestellte Entwicklungswerkzeug in eine existierende mobile Unternehmensplattform integriert werden kann.

Sowohl die Literaturstudie als auch die empirischen Untersuchungen dieser Arbeit haben gezeigt, dass die Motivation der Endbenutzer das entscheidende Erfolgskriterium der Endbenutzerentwicklung ist. Die vorliegende Arbeit versucht die Motivation für die Endbenutzer durch einen geringen Einarbeitungsaufwand und das Ermöglichen zügiger Resultate anzusprechen. Der Fokus liegt jedoch auf der technischen Umsetzung des Werkzeuges. Weitere Anreizfaktoren werden nicht untersucht. Auch geeignete Organisationsstrukturen oder Rahmenbedingungen zur erfolgreichen Einführung und Nutzung des Werkzeuges in Unternehmen werden nicht eruiert. Weitere Forschungsmöglichkeiten bestehen daher in der Untersuchung der notwendigen Rahmenbedingungen und zusätzlicher Anreizfaktoren für den erfolgreichen Einsatz des Entwicklungswerkzeuges.

Literaturverzeichnis

Alexander, C. (1979): The Timeless Way of Building, Oxford University Press, New York, USA 1979.

Andresen, A. (2003): Komponentenbasierte Softwareentwicklung: mit MDA, UML und XML, Hanser-Verlag, München, Deutschland 2003.

Apache (2014): Apache Tomcat 7 Documation. http://tomcat.apache.org/tomcat-7.0-doc/index.html, zugegriffen am 15.03.2014.

Apple (2013a): iOS Human Interface Guidelines. https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/MobileHIG.pdf, zugegriffen am 15.11.2013.

Apple (2013b): OS X Human Interface Guidelines. https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/AppleHIGuidelines/OSXHIGuidelines.pdf, zugegriffen am 15.11.2013.

Atteslander, P. (2010): Methoden der empirischen Sozialforschung. (13. Aufl.), Erich Schmidt-Verlag, Berlin, Deutschland 2010.

Bach, M. (2000): XSL und XPath - verständlich und praxisnah: Transformation und Ausgabe von XML-Dokumenten mit XSL, Addison-Wesley, München, Deutschland 2000.

Bai, G. (2011): jQuery Mobile: First Look, Packt Publishing, Birmingham, UK 2011.

Balagtas-Fernandez, F.T. (2010): Easing the Creation Process of Mobile Applications for Non-Technical Users: Model-Driven Development of Mobile Applications. Dissertation, Ludwig-Maximilians-Universität München 2010.

Ballard, B. (2007): Designing the mobile user experience, John Wiley, West Sussex, UK 2007.

Balzert, H. (2009): Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. (3. Aufl.), Spektrum Akademischer Verlag, Heidelberg, Deutschland 2009.

Beck, K.; Cunningham, W. (1987): Using Pattern Languages for Object-Oriented Programs. Beitrag vorgestellt auf der Workshop on specification and design for object-oriented programming, Orlando, MI, USA.

Becker, J. (2010): Prozess der gestaltungsorientierten Wirtschaftsinformatik. In: Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz. Hrsg.: Österle, H.; Winter, R.; Brenner, W., Infowerk AG, Nürnberg, Deutschland 2010, 13-17.

Becker, J.; Kugeler, M.; Rosemann, M. (2008): Prozessmanagement: Ein Leitfaden zur prozessorientierten Organisationsgestaltung (6. Aufl.), Springer-Verlag, Berlin, Deutschland 2008.

Beckert, A.; Beckert, S.; Escherich, B. (2012): Mobile Lösungen mit SAP, Galileo Press, Bonn, Deutschland 2012.

Behrens, J.; Giesecke, S.; Jost, H.; Matevska, J.; Schreier, U. (2006): Architekturbeschreibung. In: Handbuch der Software-Architektur. Hrsg.: Reussner, R.; Hasselbring, W., dpunkt-Verlag, Heidelberg, Deutschland 2006, 35-64.

Bensberg, F. (2009): Mobile Business Intelligence. In: Erfolgsfaktoren des Mobile Marketing. Hrsg.: Bauer, H.; Bryant, M.; Dirks, T., Springer-Verlag, Berlin 2009, 71-87.

Beringer, J. (2004): Reducing Expertice Tension. In: Communications of the ACM, Vol. 47 (2004) No. 9, 39-40.

Berti, S.; Paternò, F.; Santoro, C. (2006): Development of Nomadic Interfaces Based on Conceptual Descriptions. In: End User Development. Hrsg.: Lieberman, H.; Paternó, F.; Wulf, V., Springer-Verlag, Dordrecht, Netherlands 2006, 143-160.

Bichler, M. (2006): Design Science in Information Systems Research. In: Wirtschaftsinformatik, Vol. 48 (2006) No. 2006, 133-142.

Böing, T. (2011): Moderne Unternehmen effizent managen - ein Erfahrungsbericht bei zalando.de. In: Wirtschaftliche Geschäftsprozesse durch mobile ERP-Systeme. Hrsg.: Gronau, N.; Fohrholz, C., GITO-Verlag, Berlin, Deutschland 2011, 231-254.

Borchers, J. (2001): A pattern approach to interaction design, John Wiley, Chichester, UK 2001.

Bortz, J.; Döring, N. (2006): Forschungsmethoden und Evaluation. (4. Aufl.), Springer-Verlag, Heidelberg, Deutschland 2006.

Brehm, L.; Heinzl, A.; Markus, M.L. (2001): Tailoring ERP Systems: A Spectrum of Chices and their Implications. Beitrag vorgestellt auf der Proceedings of the 34th Hawaii International Conference on System Sciences, Maui, Hawaii, USA.

Brinzarea-Iamandi, B.; Darie, C.; Hendrix, A. (2009): AJAX and PHP: Building Modern Web Applications. (2. Aufl.), Packt Publishing, Birmingham, UK 2009.

Burnett, M. (2009): What is End-User Software Engineering and Why Does It Matter? In: Lecture Notes in Computer Science, Vol. 5435 (2009), 15-28.

Buxmann, P.; Weitzel, T.; Westarp, F.; König, W. (1999): The Standardization Problem: An Economic Analysis of Standards in Information Systems. Beitrag vorgestellt auf der 1st IEEE Conference on Stanardization and Innovation in Information Technology (SIIT ´99), Aachen, Deutschland, 157-162.

Cappiello, C.; Matera, M.; Picozzi, M. (2013): End-User Development of Mobile Mashups. Beitrag vorgestellt auf der Second International Conference on Design, User Experience, and Usability. Web, Mobile, and Product Design (DUXU 2013), Las Vegas, NV, USA, 641-650

Clark, J. (2010): Tapworthy: Designing Great iPhone Apps, O'Reilly, Sebastopol, CA, USA 2010.

Coar, K.; Bowen, R. (2003): Apache Cookbook, O'Reilly Sebastopol, CA, USA 2003.

Cooper, A.; Reimann, R.; Cronin, D. (2007): About Face: The Essentials of Interactin Design. (3. Aufl.), Wiley Publishing, Indianapolis, IN, USA 2007.

Cuno, A.M. (2013): Konzeption und prototypische Implementierung eines Softwarewerkzeuges zur modellbasierten Generierung mobiler ERP-Applikationen. Masterarbeit, Technische Universität München 2013.

Czarnecki, K.; Eisenecker, U.W. (2000): Generative Programming: Methods, Tools and Applciations, Addison-Wesley, Boston, MA, USA 2000.

Dahm, M. (2006): Grundlagen der Mensch-Computer-Interaktion, Pearson Verlag, München, Deutschland 2006.

Danado, J.; Paternò, F. (2012): Puzzle: A Visual-Based Environment for End User Development in Touch-Based Phones. In Winckler, M.; Forbirg, P. (Eds.), *4th International Conference on Human-Centred Software Engineering* (pp. 199-216). Toulouse, France.

Dearden, A.; Finlay, J. (2006): Pattern Languages in HCI: A critical review. In: Human computer interaction, Vol. 21 (2006) No. 1, 49-102.

DIN (1998): Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 11: Anforderungen an die Gebrauchstauglichkeit (Deutsche Fassung), ISO 9241-11. Deutsches Institut für Normung e.V.

DIN (2006): Ergonomie der Mensch-System-Interaktion - Teil 110: Grundsätze der Dialoggestaltung (Deutsche Fassung), ISO 9241-110. Deutsches Institut für Normung e.V.

Dix, A.; Finlay, J.; Abowd, R.D.; Beale, R. (2004): Human-Computer Interaction. (3. Aufl.), PrenticeHall, Essex, U.K. 2004.

Dospinescu, O.; Fotache, D.; Munteanu, B.A. (2008): Mobile Enterprise Resource Planning: New Technology Horizons. In: Communications of the IBIMA, Vol. 1 (2008), 91-97.

Dudney, B.; Adamson, C. (2010): Entwickeln mit dem iPhone SDK, O'Reilly Köln, Deutschland 2010.

Ebert, C. (2012): Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten. (4. Aufl.), dpunkt Verlag, Heidelberg, Deutschland 2012.

ECMA (2013): The JSON Date Interchange Format. Geneva, Schweiz: ECMA International.

Essigkrug, A.; Mey, T. (2007): Rational Unified Process kompakt. (2. Aufl.), Spektrum Akademischer-Verlag, München, Deutschland 2007.

Ferstl, O.; Sinz, E. (2008): Grundlagen der Wirtschaftsinformatik. (6. Aufl.), Oldenbourg-Verlag, München, Germany 2008.

Fischer, G.; Girgensohn, A.; Nakakoji, K.; Redmiles, D. (1992): Supporting Software Designers with Integrated Domain-Oriented Design Environments. In: IEEE Transactions on Software Engineering, Vol. 18 (1992), 511-522.

Fitzgerald, M. (2004): Learning XSLT: A Hands-On Introduction to XSLT and XPath, O'Reilly Verlag, Sebastopol, CA, USA 2004.

Fling, B. (2009): Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps, O'Reilly, Sebastopol, CA, USA 2009.

Forsthuber, H. (2005): Praxishandbuch SAP-Finanzwesen. (2. Aufl.), Galileo Press, Bonn 2005.

Föse, F.; Hagemann, S.; Will, L. (2008): SAP NetWeaver AS ABAP - Systemadministration. (3. Aufl.), Galileo Press, Bonn, Deutschland 2008.

Fowler, M.; Rice, D.; Foemmel, M.; Hieatt, E.; Mee, R.; Stafford, R. (2002): Patterns of Enterprise Architecture, Addison-Wesley Boston, MA, USA 2002.

Frank, M.R.; Szekely, P. (1998): Adaptive Forms: An Interaction Paradigm for Entering Structured Data. Beitrag vorgestellt auf der Proceedings of the 3rd international conference on Intelligent user interfaces, San Francisco, CA, USA.

Friberg, P. (2013): Web-Apps mit jQuery Mobile: Mobile Multiplattform-Entwicklung mit HTML5 und JavaScript, dpunkt-Verlag, Heidelberg, Deutschland 2013.

Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. (1995): Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, New Jersey, U.S.A. 1995.

Gansemer, S.; Gröner, U.; Maus, M. (2007): Database Classification of Mobile Devices. Beitrag vorgestellt auf der IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Dortmund, Deutschland.

Gartner (2009): Magic Quadrant for Mobile Enterprise Application Platforms. Gartner Research, 2009.

Gartner (2013): Magic Quadrant for Mobile Application Development Platforms (G00248487). Gartner Research, 2013.

Gartner (2014): Annual Smartphone Sales Figures: 2012 - 2013. http://www.gartner.com/newsroom/id/2665715, zugegriffen am 15.04.2014.

Gawelek, M.H. (2014): Prototypische Implementierung einer Anwendung zur Gestaltung mobiler ERP-Anwendungen durch den End-Benutzer am Beispiel iPhone. Masterarbeit, Technische Universität München 2014.

Gläser, J.; Laudel, G. (2009): Experteninterviews und qualitative Inhaltsanalyse: als Instrumente rekonstruierender Untersuchungen. (3. Aufl.), VS Verlag für Sozialwissenschaften, Wiesbaden, Germany 2009.

Glass, R.L. (1998): Enterprise resource planning--breakthrough and/or term problem? In: The DATA BASE for Advances in Information Systems, Vol. 29 (1998) No. 2, 13-16.

Gorlenko, L.; Merrick, R. (2003): No wires attached: Usability challenges in the connected mobile world. In: IBM Systems Journal, Vol. 42 (2003), 639-651.

Grechenig, T.; Bernhart, M.; Breiteneder, R.; Kappel, K. (2010): Softwaretechnik: Mit Fallbeispielen aus realen Entwicklungsprojekten, Pearson-Verlag, München, Deutschland 2010.

Gregor, S. (2006): The Nature of Theory in Information Systems. In: MIS Quarterly, Vol. 30 (2006) No. 3, 611-642.

Gronau, N. (2010): Enterprise Resource Planning: Architektur, Funktionen und Management von ERP-Systemen (Vol. 2. Auflage). (2. Aufl.), Oldenbourg-Verlag, München, Deutschland 2010.

Gronau, N.; Fohrholz, C.; Plygun, A. (2012): Mobile Prozesse im ERP. In: HMD - Praxis für Wirtschaftsinformatik, Schwerpunktthema: Mobile Computing. Hrsg. dpunkt Verlag, Heidelberg 2012, 23-31.

Gruhn, V.; Köhler, A.; Klawes, R. (2005): Mobile process landscaping by example of residential trade and industry. Beitrag vorgestellt auf der Proceedings of the 13th European Conference on Information Systems, Regensburg, Deutschland.

Gruhn, V.; Pieper, D.; Röttgers, C. (2006): MDA, Springer-Verlag, Berlin, Deutschland 2006.

Hacker, W. (2013): Mobile Prototyping with Axure 7, Packt Publishing, Birmingham, UK 2013.

Hansen, H.R.; Neumann, G. (2005): Wirtschaftsinformatik 2: Informationstechnik. (9. Aufl.), Lucius & Lucius-Verlag, Stuttgart, Deutschland 2005.

Herczeg, M. (2009): Software-Ergonomie: Theorien, Modelle und Kriterien für gebrauchstaugliche interaktive Computersysteme. (2. Aufl.), Oldenbourg-Verlag, München, Deutschland 2009.

Hernandez, J.A.; Martinez, F.; Keogh, J. (2005): SAP R/3 Handbook. (3. Aufl.), McGraw-Hill, New York, USA 2005.

Hessler, M.; Görtz, M. (2008): Basiswissen ERP-Systeme, W3L-Verlag, Herdecke, Deutschland 2008.

Hevner, A.R. (2007): A Three Cycle View of Design Science Research. In: Scandinavian Journal of Information Systems, Vol. 19 (2007) No. 2, 87-92.

Hevner, A.R.; Chatterjee, S. (2010): Design Research in Information Systems: Theory and Practice, Springer-Verlag, New York, NY, USA 2010.

Hevner, A.R.; March, S.T.; Park, J. (2004): Design science in information systems research. In: MIS Quarterly, Vol. 28 (2004) No. 1, 75-105.

Hoffmann, H. (2009): Ein Werkzeug zur Entwicklung nutzerorientierter Software- und Service-Prototypen im Fahrzeug. Dissertation, Technische Universität München 2009.

Homann, M.; Banova, V.; Oelbermann, P.; Wittges, H.; Krcmar, H. (2013a): Towards User Interface Components for Dashboard Applications on Smartphones. Beitrag vorgestellt auf der Fith International Conference on Mobile Computing, Applications and Services, Paris, Frankreich, 19-32.

Homann, M.; Banova, V.; Wittges, H.; Krcmar, H. (2014a): Einsatz einer mobilen Unternehmensplattform. In: ERP Management, Vol. 10 (2014a) No. 1, 32-34.

Homann, M.; Banova, V.; Wittges, H.; Krcmar, H. (2014b): Towards an End-User Development Tool for Mobile ERP Applications. Beitrag vorgestellt auf der ERP Future 2013, Wien, Österreich.

Homann, M.; Wittges, H.; Krcmar, H. (2013b): Entwicklung mobiler Anwendungen mit SAP, Galileo Press, Bonn 2013.

Homann, M.; Wittges, H.; Krcmar, H. (2013c): Towards User Interface Patterns for ERP Applications on Smartphones. Beitrag vorgestellt auf der Business Information Systems, Poznan, Polen, 14-25.

Hug, T. (2001): Erhebung und Auswertung empirischer Daten: Eine Skizze für AnfängerInnen und leicht Fortgeschrittene. In: Einführung in die Forschungsmethodik und Forschungspraxis. Hrsg.: Hug, T., Schneider-Verlag, Hohengehren, Deutschland 2001, 11-29.

IBM (2012): Native, web or hybrid mobile-app development. ftp://public.dhe.ibm.com/software/pdf/mobile-enterprise/WSW14182USEN.pdf, zugegriffen am 4.11.2013.

IBM (2013): IBM Worklight V6.0: Technology Overview. http://public.dhe.ibm.com/common/ssi/ecm/en/wsw14181usen/WSW14181USEN.PDF, zugegriffen am 14.11.2013.

Ichikawa, T.; Jungert, E.; Korfhage, R.R. (1990): Visual Languages and Applications, Springer-Verlag, Heidelberg, Deutschland 1990.

IEEE (1990): IEEE Standard Glossary of Software Engineering Terminology, Std. Nr. 610.121990. New York, NY, USA: The Institute of Electrical and Electronics Engineers.

Jeffries, R.; Rosenberg, J. (1987): Comparing a form-based and a language-based user interface for instructing a mail program. Beitrag vorgestellt auf der Proceedings of the Conference on Human Factors in Computing Systems and Graphics Interface (CHI), Toronto, Canada.

Keller, H.; Krüger, S. (2006): ABAP Objects: ABAP-Programmierung mit SAP NetWeaver. (3. Aufl.), Galileo Press, Bonn, Deutschland 2006.

Kelly, J.F. (2010): Lego Mindstorms NXT-G Programming Guide. (2. Aufl.), Apress, New York, NY, USA 2010.

Kelly, S. (2005): Software-Modellierung ohne Kunstgriffe - Mit domänenspezifischer Modellierung zu hunderprozentiger Code-Generierung. In: Elektronik, (2005) No. 1, 64-69.

Kelly, S.; Tolvanen, J.P. (2008): Domain Specific Modeling, Enabling Full Code Generation, John Wiley Hoboken, NJ, USA 2008.

Khazanchi, D. (2000): Is information systems a science? an inquiry into the nature of the information systems discipline. In: ACM SIGMIS Database, Vol. 31 (2000) No. 2, 24-42.

Kirwan, B.; Ainsworth, L.K. (1992): The task analysis process. In: A guide to task analysis. Hrsg.: Kirwan, B.; Ainsworth, L.K., Taylor and Francis Publishing, London, UK 1992, 15-233.

Klann, M.; Paternó, F.; Wulf, V. (2006): Future Perspectives in End-User Development. In: End User Development. Hrsg.: Lieberman, H.; Paternó, F.; Wulf, V., Springer-Verlag, Dordrecht, Netherlands 2006, 475-486.

Klar, M.; Klar, S. (2006): Einfach generieren: Generative Programmierung verständlich und praxisnah, Hanser-Verlag, München, Deutschland 2006.

Kock, N.; Gray, P.; Hoving, R.; Klein, H.; Myers, M.; Rockart, J. (2002): Is Research Relevance Revisted: Subtle accomplishment, unfulfilled promise, or serial hypocrisy? In: Communications of the Association for Information Systems, Vol. 8 (2002) No. 2002, 330-246.

Koh, S.C.L.; Gunasekaran, A.; Rajkumar, D. (2008): ERP II: The involvement, benefits and impediments of collaborative information sharing. In: International Journal of Production Economics, Vol. 113 (2008), 245-268.

Kolbe, L. (2014): End User Development in Forschung und Praxis: Anwendungsfälle, Werkzeuge und Verbesserungspotentiale für mobile Anwendungen. Masterarbeit, Technische Universität München 2014.

Kolberg, M. (2010): Office 2010: Die perfekte Büroorganisation, Markt+Technik-Verlag, München, Deutschland 2010.

Krannich, D. (2010): Mobile System Design - Herausforderungen, Anforderungen und Lösungsansätze für Design, Implementierung und Usability-Testing Mobiler Systeme, Books on Demand, Norderstedt, Deutschland 2010.

Krasner, G.; Pope, S. (1988): A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. In: Journal of Object Oriented Programming, Vol. 1 (1988) No. 3, 26-49.

Krcmar, H. (2010): Informationsmanagement. (5. Aufl.), Springer-Verlag, Heidelberg, Germany 2010.

Künneth, T. (2012): Android 4: Apps entwickeln mit dem Android SDK. (2. Aufl.), Galileo Press, Bonn, Deutschland 2012.

Küpper, A.; Reiser, H.; Schiffers, M. (2007): Mobilitätsmanagement im Überblick: Von 2G zu 3,5G. In: PIK — Praxis der Informationsverarbeitung und Kommunikation, Vol. 27 (2007) No. 2, 68-73.

Kurbel, K. (2010): Enterprise Resource Planning und Supply Chain Management in der Industrie Oldenbourg-Verlag, München, Deutschland 2010.

Kurbel, K.; Dabkowski, A. (2003): A multi-tier architecture for mobile enterprise resource planning. *Wirtschaftsinformatik* (pp. 75-93).

Kurbel, K.; Jankowska, A.M.; Nowakowski, K. (2006): A mobile user interface for an ERP system. In: Issues in Information Systems, Vol. 7 (2006) No. 2, 146-151.

Kurbel, K.E. (2013): Enterprise Resource Planning and Supply Chain Management Functions, Business Processes and Software for Manufacturing Companies, Springer-Verlag, Heidelberg, Deutschland 2013.

Lahres, B.; Rayman, G. (2009): Praxisbuch Objektorientierung: Professionelle Entwurfsverfahren. (2. Aufl.), Galileo Press, Bonn, Deutschland 2009.

Lamnek, S. (2010): Qualitative Sozialforschung. (5. Aufl.), Beltz-Verlag, Basel, Schweiz 2010.

Laudon, K.C.; Laudon, J.P.; Schoder, D. (2010): Wirtschaftsinformatik: Eine Einführung. (2. Aufl.), Pearson, München, Deutschland 2010.

Lehner, F. (2003): Mobile und drahtlose Informationssysteme: Technologien, Anwendungen, Märkte, Springer-Verlag, Berlin 2003.

Leibs, D.; Goldberg, A. (1993): When Visual is Not Enough. Beitrag vorgestellt auf der Proceedings of the OOPSLA '93 Workshop on Visual Object Oriented Programming, Washington D.C., USA, 43-45.

Léopold, K.; Bischel, M. (2012): Anwendungssicht mobiler Geschäftsanwendungen. In: Smart Mobile Apps. Hrsg.: Verclas, S.; Linnhoff-Popien, C., Springer-Verlag, Berlin Heidelberg 2012, 125-145.

Leyh, C.; Heger, W. (2012): ERP Clients: Browser-Based or Dedicated: Do we need both? - An evaluation based on user perceptions. Beitrag vorgestellt auf der Innovation and Future of Enterprise Information Systems: ERP Future 2012 Conference, Salzburg, Österreich, 71-86.

Lieberman, H. (2000): Programming by Example. In: Communications of the ACM, Vol. 43 (2000) No. 3, 73-74.

Lieberman, H. (2001): Introduction. In: Your Wish is my Command: Programming by Example. Hrsg.: Lieberman, H., Morgan Kaufmann Publishers, San Francisco, CA, USA 2001, 1-6.

Lieberman, H.; Paternò, F.; Klann, M.; Wulf, V. (2006a): End-User Development: An Emerging Paradigm. In: End-User Development. Hrsg.: Lieberman, H.; Paternó, F.; Wulf, V., Springer, Dordrecht, Niederlande 2006a, 1-9.

Lieberman, H.; Paternò, F.; Klann, M.; Wulf, V. (2006b): End-User Development, Springer-Verlag, Dordrecht, Niederlande 2006b.

Limbourg, Q.; Vanderdonckt, J. (2002): comparing Tsak Models for User Interface Design. In: The Handbook of Task Analysis for Human-Computer Interaction. Hrsg.: Diaper, D.; Stanton, N., Lawrence Erlbaum Associates, Mahwah, NJ, USA 2002, 135-154.

Linnhoff-popien, C.; Verclas, S. (2012): Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse. In Verclas, S.; Linnhoff-Popien, C. (Eds.), *Smart Mobile Apps* (pp. 3-16). Berlin, Heidelberg: Springer Berlin Heidelberg.

Lüerßen, H.; Matschiner, M. (2013): Einfach mobiler. In: Software in der Logistik: Prozesse steuern mit Apps. Hrsg.: Seebauer, P., Huss-Verlag, München, Deutschland 2013, 16-20.

MacLean, A.; Carter, K.; Lövstrand, L.; Moran, T. (1990): User-tailorable systems: pressing the issues with buttons. Beitrag vorgestellt auf der CHI '90: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Seattle, Washington, USA, 175-182.

Mahemoff, M.J.; Johnston, L.J. (1998): Principles for a usability-oriented pattern language. Beitrag vorgestellt auf der Proceedings of the Australian Computer Human Interaction Conference (OzCHI), Zürich, Schweitz, 132-139.

Mall, S.; Stefanov, T.; Stadelman, S. (2012): Mobilizing your Enterprise with SAP, SAP Press, Boston, MA, USA 2012.

March, S.T.; Smith, G.F. (1995): Design and natural science research on information technology. In: Decision Support Systems, Vol. 15 (1995) No. 4, 251-266.

Martino, D.; Philipp, T. (2012): Plattformen für mobile Geschäftsanwendungen - am Beispiel der Sybase Unwired Platform. In: HMD - Praxis für Wirtschaftsinformatik, Schwerpunktthema: Mobile Computing, Vol. 286 (2012), 32-42.

Mayring, P. (2010): Qualitative Inhaltsanalyse. (11. Aufl-), Beltz-Verlag, Weinheim, Deutschland 2010.

Meixner, G.; Paternò, F.; Vanderdonckt, J. (2011): Past, Present, and Future of Model-Based User Interface Development. In: i-com, Vol. 10 (2011), 2-11.

Mernik, M.; Heering, J.; Sloane, A.M. (2005): When and How to Develop Domain-Specific Languages. In: ACM Computing Surveys, Vol. 37 (2005) No. 4, 316-344.

Merrnik, M.; Heering, J.; Sloane, A.M. (2005): When and how to develop domain-specific languages. In: ACM Computing Surveys, Vol. 37 (2005) No. 4, 29.

Mertens, P.; Bodendorf, F.; König, W.; Picot, A.; Schumann, M.; Hess, T. (2010): Grundzüge der Wirtschaftsinformatik. (11. Aufl.), Springer Verlag, Berlin, Heidelberg, Deutschland 2010.

Meuser, M.; Nagel, U. (1991): ExpertInneninterviews: vielfach erprobt, wenig bedacht. Ein Beitrag zur qualitativen Methodendiskussion. In: Qualitativ-empirische Sozialforschung: Konzepte, Methoden, Analysen. Hrsg.: Garz, D.; Kraimer, K., Westdeutscher Verlag, Opladen, Deutschland 1991, 441-471.

Microsoft (2013): UX Guide. http://msdn.microsoft.com/en-us/library/Aa511258.aspx, zugegriffen am 31.10.2013.

Mladenova, V.; Homann, M.; Kienegger, H.; Wittges, H.; Krcmar, H. (2011): Towards an Approach to Identify and Assess the Mobile Eligibility of Business Processes. Beitrag vorgestellt auf der American Conference of Information Systems Detroit, USA, 1-11.

Myers, B.A. (1986): Visual programming, programming by example, and program visualization: a taxonomy. Beitrag vorgestellt auf der Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA, 59-66.

Myers, B.A. (1992): Languages for Developing User Interfaces, Jones and Barlett Publishers, Boston, MA, USA 1992.

Myers, B.A.; Rosson, M.B. (1992): Survey on user interface Programming. Beitrag vorgestellt auf der Proceedings of the 10th Annual CHI Conference on Human Factors in Computing Systems, 195-202.

Nardi, B.A. (1993): A Small Matter of Programming: Perspectives on End User Computing, MIT Press, Cambridge, MA, USA 1993.

Naumann, J. (2009): Praxisbuch eCATT, Galileo Press, Bonn, Deutschland 2009.

Neil, T. (2012): Mobile Design Pattern Gallery, O'Reilly, Sebastopol, CA, USA 2012.

Nickerson, J.V. (1994): Visual Programming. Dissertation, New York University 1994.

Nielsen, J. (1993): Usability engineering, Academic Press, Boston, MA, USA 1993.

Nielsen, J. (1994): Heuristic Evaluation. In: Usability Inspection Method. Hrsg.: Nielsen, J.; Mack, R.L., John Wiley & Sons, New York, NY, USA 1994, 105-171.

Nielsen, J.; Budiu, R. (2013): Mobile Usability: Für iPhone, iPad, Android, Kindle, mitp-Verlag, Heidelberg, Deutschland 2013.

Niemann, F. (2011): ERP to Go - Trends bei mobiler Business-Software. Beitrag vorgestellt auf der Wirtschaftliche Geschäftsprozesse durch mobile ERP-Systeme, Potsdam, Deutschland.

Norman, D. (1986): Cognitive engineering. In: User Centered System Design: New Perspectives on Human-Computer Interaction. Hrsg.: Norman, D.; Draper, S., Lawrence Erlbaum Associates, Hillsdale, NJ, USA 1986, 31-61.

Oja, M.-k.; Lucas, W. (2010): Evaluating the Usability of ERP Systems: What can critical incidents tell us? Beitrag vorgestellt auf der Fifth Pre-ICIS workshop on ES Research, St. Louis, 1-6.

Open-Handset-Alliance (2013): Android User Interface Guidelines. http://developer.android.com/guide/topics/ui/index.html?utm_source=twitterfeed&utm_medium=twitter, zugegriffen am 31.10.2013.

Oppermann, R. (1994): Adaptive User Support: Ergonomic Design of Manually and Automatically Adaptable Software Lawrence Erlbaum Publishers, Hillsdale, NJ, USA 1994.

Österle, H.; Becker, J.; Frank, U.; Hess, T.; Karagiannis, D.; Krcmar, H.; Loos, P.; Mertens, P.; Oberweis, A.; Sinz, E.J. (2010): Memorandum zur gestaltungsorientierten Wirtschaftsinformatik. In: Gestaltungsorientierte Wirtschaftsinformatik: Ein Plädoyer für Rigor und Relevanz. Hrsg.: Österle, H.; Winter, R.; Brenner, W., Infowerk AG, Nürnberg, Deutschland 2010, 1-6.

Pang, C.; Dharmasthira, Y.; Eschinger, C.; Motoyoshi, K.; Brant, K.F. (2013): Market Share Analysis: ERP Software, Worldwide, 2012 (G00249091). Gartner Research, 2013.

Paterò, F. (2000): Model-based design and evaluation of interactive applications, Springer-Verlag, London, UK 2000.

Perry, B.W. (2004): Java Servlet & JSP Cookbook, O'Reilly Sebastopol, CA, USA 2004.

Petrasch, R.; Meimberg, O. (2006): Model Driven Architecture: Eine praxisorientierte Einführung, dpunkt-Verlag, Heidelberg 2006.

Pohl, K.; Rupp, C. (2011): Requirements Engineering Fundamentals, Rocky Nook, Santa Barbara, CA, USA 2011.

Poswig, J. (1996): Visuelle Programmierung: Computerprogramme auf graphischem Weg erstellen, Hanser-Verlag, München 1996.

Pousttchi, K.; Becker, F. (2012): Gestaltung mobil-integrierter Geschäftsprozesse. In: HMD - Praxis für Wirtschaftsinformatik, Schwerpunktthema: Mobile Computing, Vol. 286 (2012), 15-22.

Prevezanos, C. (2012): Computerlexikon 2013, Markt + Technik-Verlag, München, Deutschland 2012.

Pühler, M. (2011): Interaktive Gestaltung von Automotive Services durch softwaregestützten Einsatz domänenspezifischer Modellierung. Dissertation, Technische Universität München 2011.

Raeder, G. (1985): A Survey of Current Graphical Programming Techniques. In: Computer, Vol. 18 (1985) No. 8, 11-25.

Reppening, A.; Ioannidou, A. (2006): What Makes End-User Development Tick? 13 Design Guidelines. In: End-User Development. Hrsg. Springer-Verlag, Dordrecht, Niederlande 2006, 51-85.

Reussner, R.; Hasselbring, W. (2009): Handbuch der Software-Architektur, dpunkt-Verlag, Heidelberg, Deutschland 2009.

Richardson, L.; Ruby, S. (2007): RESTful Web Services, O'Reilly Sebastopol, CA, USA 2007.

Rijken, D. (1994): The Timeless Way... the design of meaning. In: SIGCHI Bulleting, Vol. 6 (1994) No. 3.

Robertson, S.; Robertson, J. (2006): Mastering the Requirements Process. (2. Auflage), Addison-Wesley, Upper Saddle River, NJ, USA 2006.

Rockart, J.F.; Flannery, L.S. (1983): The management of end user computing. In: Communications of the ACM, Vol. 26 (1983) No. 10, 776-784.

Rode, J.; Rosson, M.B.; Pérez-Quiñones, M.A. (2006): End User Development of Web Applications. In: End User Development. Hrsg.: Lieberman, H.; Paternó, F.; Klann, M.; Wulf, V., Springer-Verlag, Dordrecht, Netherlands 2006, 161-182.

Rodeweig, K.M.; Wagner, C. (2013): Apps programmieren für iPhone und iPad: Inkl. Xcode, Debugging, Versionierung, zahlreiche Praxisbeispiele. Aktuell zu iOS 7, O'Reilly, Bonn, Deutschland 2013.

Roth, J. (2005): Mobile Computing: Grundlagen, Technik, Konzepte. (2. Aufl.), dpunkt-Verlag, Heidelberg, Deutschland 2005.

Rubin, K.S. (2013): Essential Scrum: A Practical Guide to the most popular Agile Process, Addison-Weslex, Ann Arbor, MI, USA 2013.

Rupp, C. (2007): Requirements-Engineering und -Management: professionelle, iterative Anforderungsanalyse für die Praxis. (4. Aufl.), Hanser-Verlag, München, Deutschland 2007.

Saffer, D. (2008): Designing Gestural Interfaces: Touchscreens and Interactive Devices, Sebastopol, CA, USA 2008.

Salmre, I. (2005): Writing Mobile Code: Essential Software Engineering for Building Mobile Applications, Addison-Wesley, Amsterdam, Niederlande 2005.

Sander, P.; Stucky, W.; Herschel, R. (1995): Automaten, Sprachen, Berechenbarkeit. In: Grundkurs Angewandte Informatik IV. Hrsg.: Stucky, W., 2. Aufl. (Hrsg.), Teubner Verlag, Stuttgart, Deutschland 1995.

SAP (1998): BAPIs - Einführung und Überblick. http://help.sap.com/printdocu/core/print40b/de/pdf/cabfaapiintro.pdf, zugegriffen am 07.01.2013.

SAP (2011a): SAP User Interface Guidelines: for Blackberry Devices. http://www.sapdesignguild.org/resources/uiguidelines.asp, zugegriffen am 10.10.2013.

SAP (2011b): SAP User Interface Guidelines: for iPhone Devices. http://www.sapdesignguild.org/resources/uiguidelines.asp, zugegriffen am 10.10.2013.

SAP (2011c): SAP User Interface Guidelines: for the Android Platform. http://www.sapdesignguild.org/resources/uiguidelines.asp, zugegriffen am 10.10.2013.

SAP (2013): SAP NetWeaver Gateway Plug-In for Eclipse. http://www.sdn.sap.com/irj/scn/downloads?rid=/webcontent/uuid/b09d414f-f227-2f10-bdbf-ba31c844b432, zugegriffen am 09.12.2013.

SAP (o.J.-a): Allgemeine Einführung in die BAPIs (CA-BFA). http://help.sap.com/saphelp_46c/helpdata/de/61/f3f0371bc15d73e10000009b38f8cf/frameset.htm, zugegriffen am 13.12.2013.

SAP (o.J.-b): Flugdatenanwendung - Demo-Beispiel für Integrationstechnologien. http://help.sap.com/erp2005_ehp_04/helpdata/de/08/0a323c3980a57be10000000a11402f/content.htm?frameset=/de/2c/7d623cf568896be10000000a11405a/frameset.htm, zugegriffen am 09.12.2013.

SAP (o.J.-c): SAP Java Connector. https://help.sap.com/saphelp_nw04/helpdata/en/6f/1bd5c6a85b11d6b28500508b5d5211/content.htm, zugegriffen am 2.2.2014.

SAP (o.J.-d): SAPUI5 - UI development toolkit for HTML5. https://sapui5.NetWeaver.ondemand.com/sdk, zugegriffen am 10.12.2013.

Sarodnick, F.; Brau, H. (2011): Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung. (2. Aufl.), Huber-Verlag, Bern, Schweiz 2011.

Schaub, M. (2012): Model Based User Interface Development am Beispiel von Mobilen Enterprise Applikationen. Masterarbeit, Technische Universität München 2012.

Schiffer, S. (1998): Visuelle Programmierung: Grundlagen und Einsatzmöglichkeiten, Addison-Wesley, München, Deutschland 1998.

Scholl, A. (2009): Die Befragung. (2. Aufl.), UVK Verlagsgesellschaft, Konstanz, Deutschland 2009.

Schwarzer, B.; Krcmar, H. (2010): Wirtschaftsinformatik: Grundzüge der betrieblichen Datenverarbeitung. (4. Aufl.), Schäffer Poeschel-Verlag, Stuttgart, Deutschland 2010.

Sebestyen, T.J. (2010): XML - Einstieg für Anspruchsvolle, Addison-Weslex, München 2010.

Shneiderman, B. (1983): Direct Manipulation: A Step Beyond Programming Languages. In: IEEE Computer, Vol. 16 (1983) No. 8, 57-69.

Shneiderman, B.; Plaisant, C. (2010): Designing the user interface : strategies for effective human-computer interaction. (5th), Addison-Wesley, Boston 2010.

Shu, N.C. (1988): Visual Programming, Van Nostrand Reinhold, New York, NY, USA 1988.

Sieb, M. (2012): Mobile Enterprise Resource Planning applications: Model-based User Interface Design for mobile devices with reusable Human-Computer-Interaction Patterns. Bachelorarbeit, Technische Universität München 2012.

Singh, A.; Wesson, J. (2009): Evaluation criteria for assessing the usability of ERP systems. Beitrag vorgestellt auf der Proceedings of the 2009 Annual Research Conference of

the South African Institute of Computer Scientists and Information Technologists on - SAICSIT '09, New York, New York, USA, 87-95.

Smith, D.C.; Cypher, A.; Tesler, L. (2001): Novice programming Comes of Age. In: Your Wish is my Command: Programming by Example. Hrsg.: Lieberman, H., Morgan Kaufmann Publishers, San Francisco, CA, USA 2001, 7-19.

Snell, J.; Tidwell, D.; Kulchenko, P. (2002): Programming Web Services with SOAP, O'Reilly, Sebastopol, CA. USA 2002.

Sommerville, I. (2011): Software engineering. (9. Aufl.), Pearson Education, München, Deutschland 2011.

Sourceforge (o.J.): SAPRFC. http://saprfc.sourceforge.net, zugegriffen am 02.02.2014.

Spahn, M. (2010): Flexibilisierung und Individualisierung des betrieblichen Informationsmanagements durch End-User Development. Dissertation, Universität Siegen 2010.

Spahn, M.; Dörner, C.; Wulf, V. (2008a): End User Development of Information Artefacts: A Design Challenge for Enterprise Systems. Beitrag vorgestellt auf der 16th European Conference on Information Sytems, Galway, Ireland, 482-493.

Spahn, M.; Dörner, C.; Wulf, V. (2008b): End User Development of Information Artefacts: A Design Challenge for Enterprise Systems (Präsentationsfolien). Beitrag vorgestellt auf der European Conference on Information Systems, Galway, Ireland.

Spahn, M.; Wulf, V. (2009): End-User Development of Enterprise Widgets. Beitrag vorgestellt auf der 2nd International Symposium on End-User Development, Siegen, Germany, 106-125.

Spiering, M.; Haiges, S. (2010): HTML 5-Apps: für iPhone und Android entwickeln, Franzis-Verlag, Poing, Deutschland 2010.

Stahl, T.; Völter, M. (2005): Modellgetriebene Softwareentwicklung - Techniken, Engineering, Management, dpunkt-Verlag, Heidelberg, Deutschland 2005.

Stahl, T.; Völter, M. (2006): Model-Driven Software Development: Technology, Engineering, Management, John Wiley, Hoboken, NJ, USA 2006.

Stahlknecht, P.; Hasenkamp, U. (2005): Einführung in die Wirtschaftsinformatik. (11. Aufl.), Springer Verlag, Berlin, Deutschland 2005.

Suleiman, K.A.; Wayne, V.C. (1992): An International Visual Language. Beitrag vorgestellt auf der Proceedings of the 1992 IEEE Workshop on Visual Languages, Seattle, WA, USA, 141-147.

Sutcliffe, A. (2005): Evaluating the Costs and Benefits of End-User Development. *First Workshop on End-User Software Engineering (WEUSE)* (pp. 1-4). Saint Lois, MO, USA.

Szyperski, C. (2002): Component Software: Beyond Object Oriented Programming. (2. Auflage), Addison Wesley, London, UK 2002.

Tanenbaum, A.S. (2002): Moderne Betriebssysteme. (2. Aufl.), Prentice Hall 2002.

Tidwell, J. (2011): Designing interfaces. (2. Aufl.), O'Reilly, Sebastopol, USA 2011.

Tolvanen, J.P. (2006): Domänenspezifische Modellierung für vollständige Code-Generierung. In: Java Spektrum, (2006) No. 1, 9-12.

Tolvanen, J.P.; Kelly, S. (2004): Domänenspezifische Modellierung. In: Objektspektrum, Vol. 4 (2004).

Topi, H.; Lucas, W.; Babaian, T. (2005): Identifying Usability Issues with an ERP Implementation. Beitrag vorgestellt auf der International Conference on Enterprise Information Systems 128-133.

Ullenboom, C. (2012): Java ist auch eine Insel: Das umfassende Handbuch. (10. Aufl.), Galileo Press, Bonn 2012.

van Duyne, D.K.; Landay, J.; Hong, J. (2006): The Design of Sites: Patterns for Creating Winning Websites (Vol. 2. Auflage), Prentice Hall, Upper Saddle River, NJ, USA 2006.

van Welie, M. (2001): Task-Based User Interface Design, Vrije Universiteit Amsterdam 2001.

van Welie, M. (2013): Interaction Pattern Design Library. http:://www.welie.com/patterns, zugegriffen am 30.10.2013.

van Welie, M.; van der Veer, G.C. (2003): Pattern languages in interaction design: Structure and organization. Beitrag vorgestellt auf der Proceedings of Interact, Zuerich, Switserland, 527-534.

Voelter, M. (2013): DSL Engineering: Designing, Implementing and Using Domain-Specific Languages, dslbook.org 2013.

Wang, D.; Lee, J.R. (1993): Visual Reasoning: its Formal Semantics and Applications. In: Journal of Visual Languages and Computing, Vol. 4 (1993) No. 4, 327-356.

Warfel, T.Z. (2009): Prototyping: A Practitioner's Guide, Rosenfeld Media, New York, NY, USA 2009.

Wegelin, M.; Englbrecht, M. (2011): SAP-Schnittstellneprogrammierung. (2. Auflage), Galileo Press, Bonn, Deutschland 2011.

Weßel, C. (2010): Semi-strukturierte Interviews im Software-Engineering: Indikationsstellung, Vorbereitung, Durchführung und Auswertung – Ein Fall-basiertes Tutorium. Beitrag vorgestellt auf der Gesellschaft für Informatik (GI) Jahrestagung, Leipzig, Deutschland, 927-937.

Wharton, C.; Rieman, J.; Lewis, C.; Polson, P. (1994): The Cognitive Walkthrough Method: A Practitioner's Guide. In: Usability Inspection Method. Hrsg.: Nielsen, J.; Mack, R.L., John Wiley & Sons, New York, NY, USA 1994, 105-171.

Wiegers, K.E. (2003): Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. (2. Aufl.), Microsoft Press, Redmond, WA, USA 2003.

Won, M.; Stiemerling, O.; Wulf, V. (2006): Component-Based Apporaches to Tailorable Systems. In: End User Development. Hrsg.: Lieberman, H.; Paternó, F.; Wulf, V., Springer-Verlag, Dordrecht, Netherlands 2006.

Wulf, V.; Jarke, M. (2004): The economics of end-user developmet. In: Communications of the ACM, Vol. 47 (2004) No. 9, 41-42.

Wyllie, D. (2013): Marktüberblick App-Builder: Mobile Apps aus dem Baukasten. http://t3n.de/magazin/marktuberblick-app-builder-mobile-apps-baukasten-233354/, zugegriffen am 21.08.2013.

Yahoo (2011): Yahoo Design Pattern Library. http://developer.yahoo.com/ypatterns/, zugegriffen am 30.09.2011.

Zelewski, S.; Hohmann, S.; Hügens, T. (2008): Produktionsplanungs- und -steuerungssysteme: Konzepte und exemplarische Implementierung mithilfe von SAP R/3, Oldenbourg Verlag, München 2008.

Zheng, P.; Ni, L. (2006): Smart Phone and Next Generation Mobile Computing, Morgan Kaufmann, San Francisco, CA, USA 2006.

Zobel, J. (2001): Mobile Business and M-Commerce: Die Märkte der Zukunft erobern, Hanser-Verlag, München, Deutschland 2001.

Anhang A: Interviewleitfaden für Befragung I

Themengebiet	Fragen
Allgemeines	**Unternehmen** Name: Adresse: Internetadresse: Branche(n): Jahresumsatz: Mitarbeiteranzahl: **Interviewpartner** Name: Titel: Unternehmensbereich: Funktion: Erfahrung in der Entwicklung mobiler Applikationen:
Beschreibung der aktuellen Situation	• Für welche Betriebssysteme werden mobile Applikationen entwickelt? • Warum werden genau / nur diese Betriebssysteme fokussiert? • Welche Entwicklungswerkzeuge werden eingesetzt? • Welches Vorgehensmodell wird für die Entwicklung eingesetzt? • Wie werden die erstellten Applikationen und Anforderungen dokumentiert? • Inwieweit werden bei Ihnen Softwarekomponenten, Modelle etc. wiederverwendet? • Inwieweit werden Design Elemente wie Patterns oder Guidelines eingesetzt (für UI und Funktionalität)? • Welche Probleme entstehen durch die Entwicklung für die verschiedenen Betriebssysteme? • In welcher Form werden aktuell Modelle für die Anforderungserhebung, Dokumentation, o.ä. verwendet? • Wo existieren Potenziale für Optimierungen?
Beurteilung modellgetriebener Entwicklungsansätze im Allgemeinen	• Werden oder wurden in ihrem Unternehmen modellgetriebene Ansätze zur Entwicklung von Applikationen eingesetzt? **Ja: direkt weiter** • Welche Entwicklungswerkzeuge werden bzgl. modellgetriebener Entwicklung eingesetzt?

	• Welche Vor- / Nachteile bringt der Einsatz von modellgetriebener Entwicklung mit sich? • Wo sehen Sie Potentiale für Verbesserungen / Neuerungen im Hinblick auf modellgetriebene Entwicklungsansätze? **Nein: kurze Einführung in die modellgetriebene Entwicklung** • Welche Vor- / Nachteile könnte der Einsatz von modellgetriebener Entwicklung mit sich bringen? • Wo sehen Sie mögliche Schwachstellen des Einsatzes?
Beurteilung modellgetriebener Entwicklungsansätze für Benutzungsschnittstellen	**Kurze Vorstellung der modellgetriebenen Entwicklung von Benutzungsschnittstellen** • Welche Fragen haben Sie in Bezug auf die modellgetriebene Entwicklung von Benutzungsschnittstellen? • Welche Verbesserungen könnten ihrer Meinung nach durch den Einsatz einer modellgetriebenen Entwicklung von Benutzungsschnittstellen entstehen? • Sehen Sie Probleme die durch den Einsatz entstehen könnten? • Welche Funktionen, Prozesse, Tools, o.ä. würden Sie vermissen?

Anhang B: Interviewleitfaden für Befragung II

Themengebiet	Fragen
Allgemeines	**Unternehmen** Name: Adresse: Internetadresse: Branche(n): Jahresumsatz: Mitarbeiteranzahl: **Interviewpartner** Name: Titel: Unternehmensbereich: Funktion:
Beschreibung des persönlichen Erfahrungshintergrundes und der aktuelle Situation	• Was ist ihre Position im Unternehmen? • Für welche Aufgaben sind Sie zuständig? • Welchen themenrelevanten Erfahrungen / Ausbildung hatten Sie vor der aktuellen Position (Studium, Erfahrung im Bereich mobile Unternehmensapplikationen, etc.) • Haben Sie Erfahrung oder Einblicke in der Entwicklung mobiler Unternehmensapplikation. **Wenn nein: Frageblock überspringen** • Wie werden derzeit mobile Applikationen in Ihrem Unternehmen entwickelt? o Welche Arten von mobilen Applikationen werden entwickelt? o Können Sie Beispiele nennen? o Welche Entwicklungswerkzeuge werden verwendet? o Wie verläuft ein typisches Projekt zur Entwicklung einer mobilen Unternehmensapplikation? • Wie werden derzeit mobile Applikation in Ihrem Unternehme genutzt? o Welche Arten von mobilen Applikationen werden genutzt? o Gibt es bestimmte Vorgaben für die Nutzung mobiler Applikationen?
Beurteilung der Endbenutzer-Entwicklung im Allgemeinen	**Kurze Vorstellung der Endbenutzer-Entwicklung** • Sind Sie schon einmal mit einem solchen Entwicklungsan-

	satz in Berührung gekommen? o Wenn ja: Bei welcher Gelegenheit und in welchem Zusammenhang? o Wenn nein: Was ist Ihre spontane Meinung zu dieser Idee? • Wie stellen Sie sich die Umsetzung dieser Idee in Ihrem Unternehmen vor? o Ist dies überhaupt möglich und/oder sinnvoll? o Sehen Sie Potenzial für diesen Ansatz? o Wer sollte die Zielgruppe sein? o Welche Vorteile sehen Sie? o Welche Nachteile sehen Sie? o Welche möglichen Anwendungsfälle können Sie sich in Ihrem Unternehmen vorstellen? o Was sollte dabei beachtet werden bzw. was sind mögliche Risiken? o Gibt es Rahmenbedingungen in Ihrem Unternehme, die bei der Anwendung eines solchen Ansatzes beachtet werden müssen? • Wie würden Sie einen typischen Endbenutzer charakterisieren? o Welche Vor- und Nachteile sehen Sie aus der Perspektive des Endbenutzers bei der Umsetzung des vorgestellten Entwicklungsansatzes? • Welche Änderungen sehen Sie in Bezug auf die klassische Aufgabenverteilung bei der Umsetzung eines solchen Ansatzes? o Was sind die Aufgaben der IT-Abteilung? o Was sind die Aufgaben der Endbenutzer? o Welche Schwierigkeiten können auftreten? o Welche Vorkehrungen sollten getroffen werden, um Ihre genannten Schwierigkeiten zu vermeiden?
Beurteilung der Endbenutzer-Entwicklung speziell für mobile Unternehmensapplikationen	• Welche unterschiede sehen Sie zwischen mobilen Applikationen und traditionellen Desktop Applikationen? o Was verändert sich hierdurch für den Nutzer der Applikation? o Was verändert sich dadurch für den Entwickler der Applikation? • Wie beurteilen Sie die Übertragung des vorgestellten Entwicklungsansatzes der Endbenutzer-Entwicklung auf die Entwicklung mobiler Unternehmensapplikationen? o Finden Sie diesen Entwicklungsansatz für die Entwicklung mobiler Applikationen geeignet? o Was sind Ihre Gründe für Ihre vorherige Aussage

	○ Welche Vor- und Nachteile hat dieser Entwicklungsansatz aus Ihrer Sicht für die Entwicklung mobiler Unternehmensapplikationen? ○ Welche Probleme kann es speziell für die Umsetzung mobiler Unternehmensapplikationen geben? ○ Welche Anwendungsfälle sind aus Ihrer Sicht für einen solchen Ansatz geeignet und welche nicht?
Anforderungsermittlung für ein Endbenutzer-Entwicklungswerkzeug für mobile Unternehmensapplikationen	• Sind Ihnen bereits Entwicklungswerkzeuge bekannt, welche für Endbenutzer geeignet sind? ○ Wenn ja: Beschreiben Sie diese bitte kurz • Welche Anforderungen würden Sie an ein solches Werkzeug stellen? Wie könnte ein solches Werkzeug aussehen? ○ Was sind Muss-Anforderungen an das Werkzeug? ○ Was sind Kann-Anforderungen an das Werkzeug? • Wie würden Sie ein solches Werkzeug aufbauen, welches Ihre genannten Anforderungen umsetzt? ○ Entwicklungsprozess ○ Benutzungsschnittstelle ○ Integration von Backendschnittstellen ○ Unterstützung des Endbenutzers ○ Testen und Debugging ○ Worauf soll das Werkzeug laufen (Desktop-PC, Browser, Smartphone, etc.) ○ Interaktionstechniken des Werkzeuges **Vorstellung des beispielhaften Endbenutzer-Entwicklungswerkzeuges Appery.io** • Was ist Ihr erster Eindruck? ○ Ist das vorgestellte Werkzeug Endbenutzer tauglich? ○ Was hat Ihnen gut gefallen? ○ Was hat Ihnen nicht gefallen? ○ Wo sehen Sie Probleme bei der Nutzung des Werkzeuges? ○ Welche Verbesserungspotenziale sehen Sie? • Wie beurteilen Sie die Verwendung der wiederverwendbaren Komponenten innerhalb des Werkzeuges? ○ Sehen Sie evtl. andere Umsetzungsmöglichkeiten? • Wie beurteilen Sie die Verwendung einer visuellen Sprache? ○ Kann der Endbenutzer damit sinnvoll umgehen?

Anhang C: Interviewleitfaden zur Zwischen-Evaluation

Themengebiet	Fragen
Allgemeines	**Interviewpartner** • Erfahrung mit ERP-Systemen • Erfahrung im Umgang mit Smartphones
Bedienung des Werkzeugs	• Welchen Eindruck haben Sie bei der Erstellung Ihrer Applikationsidee mit dem Werkzeug? • Welchen Eindruck haben Sie von der Bedienung des Werkzeuges? • Sehen Sie sich in der Lage mit dem Werkzeug eigene Applikationen nach Ihren Vorstellungen zu erstellen? • Welche Verbesserungsvorschläge fallen Ihnen hinsichtlich der Bedienung des Werkzeuges ein?
Hilfefunktion des Werkzeugs	• War die angebotene Hilfefunktion für Sie hilfreich? • Auf welchen Formularseiten haben Sie auf die Hilfefunktion zugegriffen? o Formularseite mit allgemeinen Informationen o Formularseite für „Sortierte Listen-Anzeige“ o Formularseite für formularbasierte MBO-Anzeige o Formularseite für formularbasierte MBO-Bearbeitung • Welche Verbesserungsvorschläge fallen Ihnen hinsichtlich der Hilfefunktion des Werkzeuges ein?
Funktionsumfang des Werkzeugs	• Inwieweit können Sie mit dem Werkzeug die benötigen Funktionen ihrer Applikationsidee umsetzen? • Welche benötigten Funktionen waren mit dem Werkzeug nicht umsetzbar und wie wichtig ist Ihnen diese Funktion? • Welche Bestandteile des Werkzeuges empfinden Sie unnötig? • Welche Verbesserungsvorschläge fallen Ihnen hinsichtlich des Funktionsumfanges des Werkzeuges ein?

Anhang D: Aufgabenstellung – Testszenario A

In diesem kurzen Test wird das Testverfahren **Thinking Aloud** durchgeführt. Auf dem vor Ihnen liegenden mobilen Endgerät befindet sich ein Assistent, der Ihnen ermöglicht die in der Aufgabe stehende Applikation zu generieren. **Bitte fühlen Sie sich frei all ihre Schritte, Gedanken und Kritiken zu kommentieren!** Aus ihren im Test getätigten Aussagen während des Tests werden weitere Verbesserungen für den Assistenten abgeleitet. Lesen Sie die Aufgabe komplett durch bevor Sie anfangen.

Sie sind Vertriebsmitarbeiter der Firma Surf'nPro, einem eCommerce Unternehmen, das Surfbretter im Internet vertreibt, und möchten eine mobile SAP-ERP Anwendung für Surf'nPro erstellen. Die Anwendung sollte folgende Funktionalitäten besitzen

- Wählen Sie einen beliebigen Namen und ein Layout für die Applikation
- Die Anwendung soll aus **Kunden** und **Berichtswesen** enthalten (Customers and Reporting)
- Bei den **Kunden** sollen folgende Features verfügbar sein:
    - Anzeigen aller **Kunden mit dem Vornamen in einer Liste mit** der Möglichkeit **Kunden anzulegen** (Customer List, First Name with Create Customer)
    - **Anzeigen der Kundendetails**: Nachname, Vorname, Straße, Stadt, PLZ und Emailadresse (Details : Last Name, First Name, Street, City, Postal, Email)
    - **Pflegen der Kundendaten** auf der Detailseite: Aufnahme neuer Kunden und Ändern von Kundendaten (Customer Details with Create and Change Customer)
- Beim **Berichtswesen** sollen die **besten Kunden nach Umsatz** und die **bestverkauften Produkte** angezeigt werden (Reporting with maximum turnover by customer and best product)

Öffnen Sie ihre erstellte Applikation und schauen Sie sich ihr Berichtswesen an.

Vielen herzlichen Dank für ihre Teilnahme!

Anhang E: Aufgabenstellung – Testszenario B

In diesem kurzen Test wird das Testverfahren **Thinking Aloud** durchgeführt. Auf dem vor Ihnen liegenden mobilen Endgerät befindet sich ein Assistent, der Ihnen ermöglicht die in der Aufgabe stehende Applikation zu generieren. **Bitte fühlen Sie sich frei all ihre Schritte, Gedanken und Kritiken zu kommentieren!** Aus ihren im Test getätigten Aussagen während des Tests werden weitere Verbesserungen für den Assistenten abgeleitet. Lesen Sie die Aufgabe komplett durch bevor Sie anfangen.

Sie sind Angestellter im Bereich Einkauf bei der Firma Surf'nPro, einem eCommerce Unternehmen, das Surfbretter im Internet vertreibt, und möchten eine mobile SAP-ERP Anwendung für Surf'nPro erstellen. Die Anwendung sollte folgende Funktionalitäten besitzen:

- Wählen Sie einen beliebigen Namen und ein Layout für die Applikation
- Die Anwendung soll **Produkte**, **Lieferanten** und **Berichtswesen** enthalten (Products, Suppliers and Reporting)
- Bei den **Produkten** sollen folgende Features verfügbar sein:
    - Anzeige aller **Produkte in einer Liste** mit der Möglichkeit **Produkte anzulegen** (Product List with Create Product)
    - **Anzeige der Produktdetails** Bild, Name, Lieferant und Preis (Product's Details with Picture, Name, Supplier and Price)
    - **Pflege der Produktdaten** auf der Detailseite: Aufnahme neuer Produkte, Ändern von Produktdaten, Löschen von Produktdaten (Create, Change and Delete Product)
- Bei den **Lieferanten** sollen folgende Features verfügbar sein:
    - Anzeige aller **Lieferantenname in einer Liste <u>ohne</u>** die Möglichkeit **neue Lieferanten anzulegen** auf der Listenseite (Supplier List with Name without Create Supplier)
    - **Anzeige der Lieferantendetails** Name, Straße, Stadt, PLZ und Land **<u>ohne</u>** Email (Supplier's Details with Name, Street, City, Postal and Country without Email)
    - **Pflege der Lieferantendaten** auf der Detailseite: Aufnahme neuer Lieferanten, Ändern von Lieferantendaten, Löschen von Lieferantendaten (Create, Change and Delete Supplier)
- Beim **Berichtswesen** sollen die **besten Brands** und **besten Produkte** angezeigt werden (Reporting with best product and best brand)

Öffnen Sie ihre erstellte Applikation und schauen Sie sich ihre Produkte an.

Vielen herzlichen Dank für ihre Teilnahme!