FH·W·S
Hochschule
für angewandte Wissenschaften
Würzburg-Schweinfurt

FHWS SCIENCE JOURNAL

Jahrgang 1 (2013) - Ausgabe 1

V. i. S. d. P. Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt
Der Präsident
Münzstraße 12
D-97070 Würzburg
Telefon +49 (0)931/3511-6002
Fax +49 (0)931/3511-6044

Ignaz-Schön-Straße 11
D-97421 Schweinfurt
Telefon +49 (0)9721/940-602
Fax +49 (0)9721/940-620

ISSN 2196-6095

Bibliographische Information der Deutschen Bibliothek:
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliographie; detaillierte bibliographische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Neue wissenschaftliche Schriftenreihe der FHWS

Robert Grebner,
Präsident der FHWS

Die Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt (FHWS) führt in fünf „In-Instituten“, über 90 Laboren und in verschiedenen Projekten anwendungsorientierte Forschung und Entwicklung durch. Die Intensivierung der angewandten Forschung ist ein wesentliches Ziel der FHWS. Um die Ergebnisse der Forschungsaktivitäten an der Hochschule zu veröffentlichen, etabliert die FHWS eine neue wissenschaftliche Schriftenreihe: das FHWS Science Journal.

Diese vorliegende erste Ausgabe des FHWS Science Journals erscheint als Sonderausgabe im Nachgang zum Kongress „Mobility moves Business“ 2013, bei dem Anwender, Anbieter, Wissenschaftler und Studierende in Würzburg zusammentrafen, um sich über die Verbesserung von mobilen Softwaresystemen für Unternehmen auszutauschen. Daher liegt der Fokus der vorliegenden Sonderausgabe auf Beiträgen, die sich aus dem zurückliegenden Kongress ergeben haben – ergänzt um weitere Themen aus dem Bereich der Informatik und Wirtschaftsinformatik.

Die noch folgenden Ausgaben des FHWS Science Journals sollen das komplette Spektrum der Forschungsaktivitäten der FHWS anspruchsvoll ab-bilden. Publiziert werden disziplinübergreifende Highlights aus der angewandten Forschung, zum Beispiel wissenschaftliche Veröffentlichungen und Artikel, Forschungsberichte, Ergebnisse aus Masterarbeiten sowie Beiträge zu Kongressen oder Tagungen. Die wissenschaftliche Schriftenreihe soll zu eigenen und gemeinschaftlichen Veröffentlichungen mit Partnern anregen und wissenschaftliche Ergebnisse außerhalb und innerhalb der FHWS kommunizieren.

Seien Sie nun gespannt auf diese und weitere Ausgaben dieses neuen hochschuleigenen Wissenschaftsjournals.

Prof. Dr. Robert Grebner
Präsident der Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt

Artificial Communication - Can Computer Generated Speech Improve Communication of Autistic Children?

EBERHARD GRÖTSCH, ALFREDO PINA, MICHAEL SCHNEIDER, BENNO WILLOWEIT

Autistic children are often motivated in their communication behavior by pets or toys. Our aim is to investigate, how communication with “intelligent“ systems affects the interaction of children with untypical development. Natural language processing is intended to be used in toys to talk to children. This challenging Háblame-project (as part of the EU-funded Gaviota project) is just starting. We will discuss verification of its premises and its potentials, and outline the technical solution.

Categories and Subject Descriptors: Artificial Intelligence

Additional Key Words and Phrases: Natural Language Processing, Autistic, Autonomous Toy, Robot, Children.

1 INTRODUCTION

It is a well-established fact that autistic children often are motivated in their communication behavior by pets or toys, e.g. in the IROMEC project (Ferari, Robins, Dautenhahn, 2009), (IROMEC, 2013). We found analogous results in a group of disabled persons who were motivated by technical systems to move or dance. (Gaviota, 2012).

Within the Gaviota Project (Gaviota, 2012), we want to investigate, how communication with “intelligent“ systems affects the interaction of children with untypical development.

2 PREVIOUS WORK

2.1 The Beginning: Eliza

As early as 1966 Weizenbaum (Weizenbaum, 1966) implemented an interaction technique which was introduced by Carl Rogers (client centered psycho-therapy, (Rogers, 1951)). This therapy mainly paraphrases the statement of the client. The Eliza implementation used to react to a limited number of key words (family, mother, ...) to continue a dialog. Eliza had no (deep) knowledge about domains - not even shallow reasoning, rather a tricky substitution of strings. Modern versions of Eliza can be tested on several websites, e.g. (ELIZA, 2013).

Author's address: Eberhard Grötsch[1], Alfredo Pina[2], Michael Schneider[1], Benno Willoweit[1]; [1] Fakultät für Informatik und Wirtschaftsinformatik, Hochschule für Angewandte Wissenschaften, Sanderheinrichsleitenweg 20, Würzburg, Deutschland [2] Departamento de Informática, Universidad Pública de Navarra, Campus Arrosadia, Pamplona, España. eberhard.groetsch@fhws.de, _pina@unavarra.es, {michael.schneider, benno.willoweit}@student.fh-wuerzburg.de.

2.2 Robots in Autism Therapy

So far robots in autism therapy have been used to enhance the abilities of children to play, using robots as a toy, which means they playfully interact with robots. The robot's simple face can be changed to show feelings of sadness or happiness by different shapes of the mouth (IROMEC, 2013). These robots (which are just special computer screens in a first step) execute pre-defined scenarios of interaction, and are controlled by humans.

So far results have shown that more children are responding to those robots compared to the children that do not respond.

2.3 State-of-the-Art Dialog Systems

State of the art dialog systems (e.g. the original Deutsche Bahn system giving information about train time tables, or the extended system by Philips) are able to guide people who call a hotline and execute standardized business processes (delivering account data, changing address data, etc.). Those systems work well, but within an extremely limited domain.

2.4 Natural Language Processing (NLP)

A spectacular demonstration of natural language processing was given by IBM's artificial intelligence computer system Watson in 2011, when it competed on the quiz show Jeopardy! against former human winners of that popular US television show (JEOPARDY, 2011).

IBM used the Apache UIMA framework, a standard widely used in artificial intelligence (UIMA, 2013). UIMA means "Unstructured Information Management Architecture".

UIMA can be viewed from different points of view:

1) architectural: UIMA represents a pipeline of subsequent components which follow each other in an analytical process, to build up structured knowledge out of unstructured data. UIMA primarily does not standardize the components, but the interfaces between components.

"... for example "language identification" => "language specific segmentation" => "sentence boundary detection" => "entity detection (person/place names etc.)". Each component implements interfaces defined by the framework and provides self-describing metadata via XML descriptor files. The framework manages these components and the data flow between them. Components are written in Java or C++; the data that flows between components is designed for efficient mapping between these languages". (UIMA, 2013).

2) UIMA supports the software architect by a set of design patterns.

3) UIMA contains two different ways of representing data: a fast in-memory representation of annotations (high-performance analytics) and an XML representation (integration with remote web services).

The source code for a reference implementation of this framework is available on the website of the Apache Software Foundation.

Systems that are used in medical environments to analyze clinical notes serve as examples.

2.5 Natural Language Processing in Pedagogics

So far there are no reasoning systems with knowledge about the domain of how to behave properly in a pedagogical way.

3 HYPOTHESIS: NATURAL LANGUAGE SPEAKING MIGHT BE HELPFUL

The IROMEC project demonstrated that weekly sessions with a robot with rather simple abilities to move and show emotions by standardized facial expressions are helpful to enable/empower children to play more naturally than without those sessions (Ferari, Robins, Dautenhahn, 2009). So we concluded that it is worth trying to build a robot, which is talking autonomously with a child in rather simple and standardized words and sentences. We decided to start a subproject Háblame („talk to me“) to investigate the chances and problems of building such a robot as part of the EU-funded Gaviota project.

4 THE PROJECT „HÁBLAME“

4.1 Verification of the Hypothesis

Before we start the core project, we have to verify our hypothesis: we have to show that autistic children positively react to toys which talk to them. We will build a simple prototype without NLP-functions. Speech will be produced by a hidden person via microphone and suitably placed speakers.

4.2 Concept of a Dialog System

Within the project, we first had to / have to get experience with natural language processing. When we studied basic concepts of NLP (Figure 1), we decided to put stress on syntax parsing and semantic parsing.

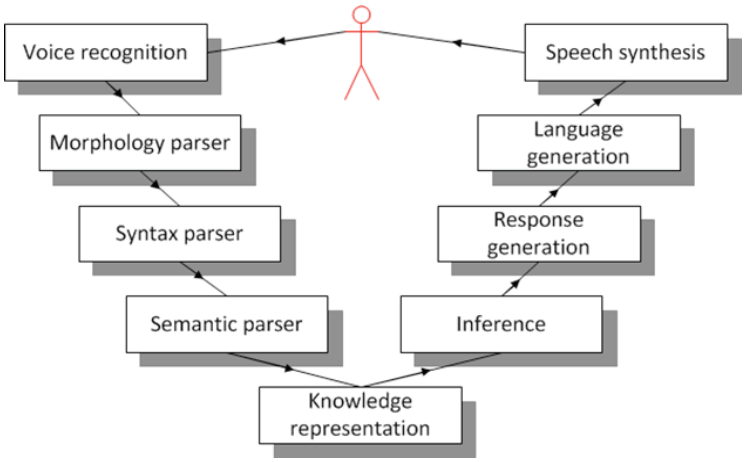

Figure 1: Concept of a dialog system (Schneider, 2012)

4.3 Parsing Syntax of Natural Languages

First a prototype parser – based on a grammar developed by Roland Hausser (Hausser, 2000) – was implemented, which can analyze simple sentences entered in English. The parser processes the sentence entered, splits it into words and compares them to a lexicon, specified in an external text file.

It tries to recombine the sentence word by word, taking account of the *valences*, also specified in the lexicon. If the sentence can be re-combined correctly and all free valences are filled, the parsing process was successful. Otherwise the sentence is grammatically incorrect (or the parser could not deal with it).

4.3.1 Parser Prototype and Valences

The parser works with valences of words, e.g.:

- to sleep has 1 nominative valence → Peter sleeps.
- to give has 1 nominative valence (abbreviated Nx), 1 dative valence (Dx) and 1 accusative valence (Ax) → Peter gives Mary books.
- All valences (mostly opened by verbs) have to be filled (mostly by nouns). Otherwise the sentence is not correct, e.g.: Peter gives Mary. → accusative noun is missing

One can think of valences as slots, which have to be filled with proper words.

4.3.2 Processing Valences

Valid words, their valences and their function (V = verb, PN = plural noun, etc.) have to be specified in an external lexicon, e.g.:

- *sleeps NS3x V*
  (S3: use only with 3rd person singular)
- *give N-S3x Dx Ax V*
  (-S3: use NOT with 3rd person singular)
- *books PN*

Words currently have to be entered in the lexicon with all flection forms used, e.g.:

- *give N-S3x Dx Ax V*
- *gives NS3x Dx Ax V*
- *gave Nx Dx Ax V*

The parser takes the first word of the sentence and combines it with the following word to a more complex starting sequence using predefined rules, e.g.:

- Noun phrase followed by a verb with corresponding valence → erase the valence satisfied:
  Peter (SNP) sleeps (NS3x V). →
  Peter sleeps (V).
- Article followed by adjective → do not change any valences:
  The (SNx SNP) beautiful (ADJ) … →
  The beautiful (SNx SNP) …

This combining procedure is repeated bottom-up until the end of the sentence is reached (Figure 2).

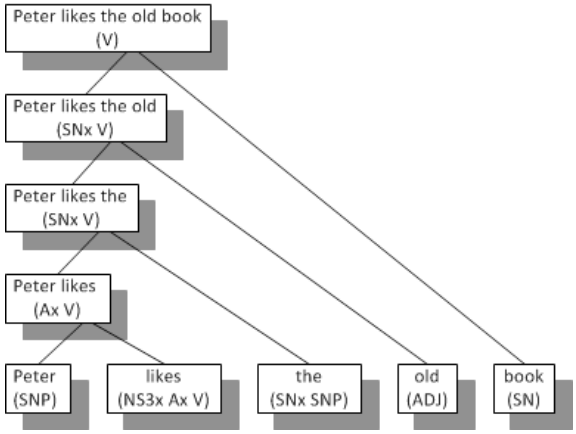

Figure 2: Bottom-up processing of valences (Schneider, 2012), cf. (Hausser, 2000)

Examples of sentences, the prototype of the parser can deal with:
- *The beautiful girl reads an old book.*
- *Does Peter sleep?*
- *Mary has bought a new car.*

Examples of sentences, the prototype currently cannot deal with:
- *Beautiful girls like Peter.*
- *Reading books gives Peter pleasure.*
- *Peter, who is 20 years old, sleeps.*

4.4 Processing of Semantics of Natural Languages – Analyzing Semantics

Analyzing the semantics of natural language, we first define our prerequisites and our goals:

- Prerequisites:

Oral utterances (of children) are transcribed by a supervisor and fed into the system. The sentences are analyzed one by one, and the results of the analysis should be stored in a semantic network

- Goals:
  - Exploring the linguistic techniques for semantic analysis.
  - Determining the technical and linguistic preconditions.
  - Evaluate which software components and libraries may be used to accomplish this task.
  - Evaluate which libraries can be used to access a semantic network, and how to create the necessary ontologies.
  - Building a software prototype, which integrates all necessary components.

Basically, there are two approaches towards linguistic analyzing:

The „formal“ approach:

Every sentence represents a logical statement („Proposition“), and we have to translate every sentence into meta-language. Those languages are called „Meaning Representation Languages“ (MRL) and are often based on first order logic or the lambda calculus.

The „cognitive“ approach:

One can‘t determine the exact meaning of a sentence by the sentence itself. A straightforward translation of language into a logical representation is therefore impossible.

In the process of understanding there is a lot of background knowledge involved.

This knowledge may be specific to a single person or a group of persons (e.g. cultural or personal background)

4.4.1 Adoption in Computational Linguistics

The formal approach is well explored and adopted in Computational Linguistics.

Its main advantages are easy integration with code and other logical structures like semantic networks. The disadvantage is that it is not language agnostic and very narrow in scope (one has to define logical expressions for every meaning of a sentence).

The cognitive approach was investigated mainly by adopting Fillmore‘s work on frame semantics, which he developed back in the 1970s (Fillmore, 2006). His idea was that the meaning of a sentence can be described by a so-called frame or a combination of those. A frame is consisting of:

- A description which outlines the meaning of the frame
- A number of frame elements (FE) that describe possible roles or agents
- Relations to other frames, including specialization, part-of or temporal relations
- A number of language specific lexical units, i.e. words or groups of words, which may evoke that frame.

The main advantage of the cognitive, frame-based approach is, that frames are language agnostic, so only the lexical units that may evoke a frame have to be defined per language. Every frame is a formal representation of meaning, so there is no reason to build an own meta-language. The scope is very broad and not limited to a specific application.

4.4.2 Software Tools for FrameNet Based Analysis (Cognitive Approach)

The FrameNet database consists of a large set of XML files (FrameNet, 2012).

Frame semantic parsers relying on FrameNet already exist, both systems use a probabilistic approach:

- SHALMANESER (English, German) is a project at Saarland University, Saarbrücken, Germany

and

- SEMAFOR (English) is a project at Carnegie Mellon University, Pittsburgh, USA

4.4.3 Preprocessing of Sentences (Cognitive Approach)

In a first step we preprocess the sentences to be analyzed:

- Tokenizing: we split sentences into words (Apache NLP Tools)
- POS-Tagging: we determine the part of speech of each token (Apache NLP Tools)
- Syntactic parsing: Determining the grammatical components of each sentence (Maximum Spanning Tree Parser, Pennsylvania State University)
- Named Entity Recognition: Check if one or more tokens represent a proper noun, a number, a date, etc. (Apache NLP Tools)
- Frame identifications: Find the frames that match the given sentence (Semafor, Carnegie Mellon University, Pittsburgh, USA)

5 RESULTS

So far there are only results, as far as NLP is concerned:

- The pre-trained classifiers for both SHALMANESER and SEMAFOR did not yield good results with our test data.
- SHALMANESER is hard to integrate with other tools.
- There are plenty of java-based tools to preprocess the data and extract features that can be used with probabilistic models. Furthermore, many of these tools can be integrated with the Apache UIMA platform.
- A modular, client/server based approach proved to be necessary for the project.
- A fairly large corpus of transcribed child language is nearly impossible to obtain.
- Although there are FrameNet data sets for a couple of languages (Spanish, German, Chinese, etc.), their number of frames and lexical units is presumably too small to use for semantic parsing.

6 CONCLUSIONS

First we have to verify that autistic children react to the prototype system in the manner expected.

If this is done successfully, there is much work left to be done on the NLP side. We will not do further research on using FrameNet with the Semafor parser however, nor use *database semantics* (another approach, which is not covered in this report).

We will intensify research on custom probabilistic models with the following steps:

1. set up Apache UIMA since the NLP tools are easy to integrate,
2. obtain a domain specific corpus,
3. split that corpus into a training and a test part,
4. annotate the corpus with semantic class labels,
5. select domain specific and situational features
6. incorporate the features generated by the pre-processing tools (i.e. taggers, parsers, etc.)
7. train a probabilistic model, possibly by using the MaxEnt library of the Apache NLP tools,
8. evaluate the performance with different feature sets

6.1 Necessary Data

We need corpora about children's language domains, and we have to decide, which age level, and which speech domains. If no corpus is available, we have to develop one. Those corpora should be in English language to develop and stabilize the system. Later iterations may incorporate German and Spanish language.

6.2 Further Steps

We will set up an experimental environment, based on the work already done, gather experience and knowledge on analyzing/parsing natural language. Then we have to acquire or produce corpora covering our domain of interest (child language). Furthermore we have to work on the problem of creating natural sentences as part of a dialog.

Acknowledgements

We thank Frank Puppe, Universität Würzburg for his helpful discussions.

This work has been partially funded by the EU Project GAVIOTA (DCI-ALA/19.09.01/10/21526/ 245-654/ALFA 111(2010)149.

REFERENCES

ELIZA, 2013.
WWW.MED-AI.COM/MODELS/ELIZA.HTML (MARCH 3,2013)

GAVIOTA, 2012. REPORT ON THE RESULTS OF THE GAVIOTA PROJECT, INTERNATIONAL MEETING, SANTA CRUZ, BOLIVIA (UNPUBLISHED PRESENTATIONS)

FERARI, E., ROBINS, B., DAUTENHAHN, K., 2009. ROBOT AS A SOCIAL MEDIATOR - A PLAY SCENARIO IMPLEMENTATION WITH CHILDREN WITH AUTISM, 8TH INTERNATIONAL CONFERENCE ON INTERACTION DESIGN AND CHILDREN WORKSHOP ON CREATIVE INTERACTIVE PLAY FOR DISABLED CHILDREN, COMO, ITALY

FILLMORE, C. J., 2006. FRAME SEMANTICS, IN GEERAERTS, D. (ED.): COGNITIVE LINGUISTICS - BASIC READINGS, CHAP. 10, MOUTON DE GRUYTER, P. 373–400.

FRAMENET, 2012.
THE FRAMENET PROJECT, UNIVERSITY OF CALIFORNIA, BERKELEY, HTTPS://FRAMENET.ICSI.BERKELEY.EDU (MAR 06, 2013)

HAUSSER, R., 2000. GRUNDLAGEN DER COMPUTERLINGUISTIK – MENSCH-MASCHINE-KOMMUNIKATION IN NATÜRLICHER SPRACHE, SPRINGER VERLAG BERLIN

IROMEC, 2013.
HTTP://WWW.IROMEC.ORG/9.0.HTML (JAN 27, 2013)

JEOPARDY, 2011. HTTP://WWW.NYTIMES.COM/2011/02/17/SCIENCE/17JEOPARDY-WATSON.HTML?_R=0, (JAN 28, 2013)

ROGERS, C. R., 1951. CLIENT-CENTERED THERAPY, OXFORD, HOUGHTON MIFFLIN

SCHNEIDER, M., 2012. PROCESSING OF SEMANTICS OF NATURAL LANGUAGES – PARSING SYNTAX OF NATURAL LANGUAGES, BACHELOR-THESIS, HAW WÜRZBURG-SCHWEINFURT

UIMA, 2013.
HTTP://UIMA.APACHE.ORG/ (JAN 28, 13)

WEIZENBAUM, J., 1966. ELIZA - A COMPUTER PROGRAM FOR THE STUDY OF NATURAL LANGUAGE, COMMUNICATION BETWEEN MAN AND MACHINE. COMMUNICATIONS OF THE ACM. NEW YORK 9.1966,1. ISSN 0001-0782

WILLOWEIT, B., 2012. PROCESSING OF SEMANTICS OF NATURAL LANGUAGES – ANALYZING SEMANTICS, BACHELOR-THESIS, HAW WÜRZBURG-SCHWEINFURT

Präsentationssteuerung mit berührungslosen Gesten

MELANIE SAUL
Hochschule für angewandte Wissenschaften Würzburg-Schweinfurt

Mithilfe der Kinect ist es möglich, neue Interaktionsmöglichkeiten innerhalb von Präsentationen zu realisieren. Eine Abbildung mit der Hand zu greifen, sie umzudrehen, um sie von einer anderen Perspektive zu betrachten oder in die Abbildung zu zoomen, um ein Detail hervorzuheben, sind hierfür exemplarische Szenarien. In dieser Arbeit wird geklärt, inwieweit berührungslose Gesten als Eingabemodalität für Präsentationsanwendungen einen geeigneten Einsatz finden. Als Anwendungsfall wird die Präsentation im Hochschulbereich betrachtet. Die Anforderungen an ein solches Interface und dessen Interaktion mit dem Menschen werden unter Einbeziehung potenzieller Benutzer ermittelt. In diesem Zusammenhang wird die Frage beantwortet, welche Aktivitäten grundsätzlich beim Präsentieren zu unterstützen sind. Daraus ergeben sich im Anschluss Szenarien für den Anwendungsfall.

Kategorie und Themenbeschreibung: Interaction Design

Zusätzliche Schlüsselwörter: Mobile Interaktionsdesign, Präsentationssteuerung, Kinect, berührungslose Gesten, Integration des Publikums, Interaktive Steuerung, Raumgesten.

1 EINLEITUNG

Mit der stetigen Weiterentwicklung der Technik stellt sich zunehmend die Frage, wie Menschen möglichst einfach und effizient mit Computersystemen interagieren können [Zühlke 2012]. Aktuell ist daher der Fokus im Bereich der Mensch-Computer-Interaktion auf die Entwicklung von Schnittstellen gerichtet, welche die menschliche Gestik erkennt und interpretiert. Die Gestik reicht dabei von einfachen Finger- und Handbewegungen bis hin zum Einsatz des ganzen Körpers [Herczeg 2006].

Der aktuelle Trend geht in Richtung Multi-Touch-Anwendungen, wobei dies nur ein erster Schritt in Richtung intuitive Schnittstelle ist. Etwas mit der Hand zu greifen, umzudrehen und zu drücken als Interaktionsmöglichkeiten stellen ein neues Bedienerlebnis dar. Mit der Kinect von Microsoft ist erstmals eine kostengünstige Technik für die berührungslose Interaktion auf dem Markt. Damit ist die Grundlage geschaffen, Gesten zur Steuerung von Computersystemen in einem breiteren Nutzungskontext als bisher einzusetzen [Melgar 2012].

Wie neuartige Werkzeuge sinnvoll in den Lernprozess einzubinden sind, um die Lernergebnisse zu verbessern, gilt es wiederkehrend zu hinterfragen [Tse et al. 2012]. In dieser Arbeit ist zu klären, inwieweit die menschliche Gestik als Eingabeform für Präsentationsanwendungen wie Power-Point im Hochschulbereich geeigneten Einsatz findet.

2 PRÄSENTATIONSSTEUERUNG 2.0 – STATUS QUO

Aktuell existieren mehrere Formen der Präsentationssteuerung wie die Steuerung über Tastatur, Maus oder einer schnurlosen Fernbedienung (Presenter). Als funktionale

Adresse der Autorin: Melanie Saul, Institut Design und Informationssysteme (IDIS), www.fhws.de/idis.

Anforderungen eines handelsüblichen Presenters, welcher häufig das Mittel der Wahl darstellt, ist Starten und Beenden der Präsentation, das Vor- und Zurückblättern und das Pausieren anzusehen [o. V. 2009]. Je nach Auswahl des Presenters kommt die Einbindung zusätzlicher Funktionen hinzu wie z. B. die Integration eines Laser Pointers als Zeigemedium oder eine Zeitanzeige inkl. Vibrationsalarm zum Managen der Präsentationsdauer.

Zu den nichtfunktionalen Anforderungen zählen eine hohe Reichweite zwischen Presenter und USB-Empfänger, eine einfache Softwareinstallation per Plug & Play, intuitive Bedienelemente sowie eine gute Handlichkeit [o. V. 2009].

Im Folgenden werden bereits realisierte Präsentationsteuerungen verwandter Arbeiten betrachtet. Aufgrund der Ähnlichkeiten lassen sich daraus weitere notwendige Anforderungen identifizieren [Nielsen et al. 2004]. Zunächst wird das Anwendungssystem Charade für gestenbasierte Systeme herangezogen. Bei diesem System stehen dem Benutzer mithilfe eines Datenhandschuhs 16 Handgesten zum Navigieren, Markieren und Hervorheben eines Bereiches innerhalb einer Präsentation zur Verfügung. Im Vergleich zum Presenter wurde diese Steuerung bezüglich dessen Zuverlässigkeit kritisch bewertet [Baudel et al. 1993].

Ein weiteres Beispiel ist das System „Maestro“ von Fourney et al. Dieses basiert auf der Verwendung einer Webkamera, über die Handgesten mittels farbunterschiedlichen Handschuhen und mit Unterstützung einer maßgeschneiderten Software erkannt sowie gesteuert werden. Vor der Entwicklung dieses Systems analysierten Fourney et al. Videopräsentationen hinsichtlich der verwendeter Gesten. Dabei zeigt sich, dass Präsentierende häufig Zeigegesten gebrauchen, um einzelne Punkte hervorzuheben oder sie von anderen Inhalten abzugrenzen. Auch ist als Ergebnis festzuhalten, dass das Herausstellen von Beziehungen durch Ziehen einer Linie eine verbreitete Handlung ist. Bei der Vorstellung der Ergebnisse aus der späteren Evaluation des Systems weisen Fourney et al. darauf hin, dass die Funktionalität des Zoomens hohen Zuspruch findet [Fourney et al. 2010].

Neben der Betrachtung verwandter Arbeiten ist das Verständnis des Nutzungskontextes eine relevante Informationsquelle für die Ermittlung der Anforderungen des Systems und damit wesentlich für die Systemgestaltung [Heinecke 2012]. So umfasst der Kontext des Interfaceprototyps die Präsentationssteuerung im Hochschulbereich und damit den Präsentierenden sowie das Publikum. In klassischen Vorlesungen tritt der Dozent dabei als Präsentierender vor das Publikum, wohingegen es sich bei Vorlesungen mit seminaristischem Charakter unter dem Dozenten und den Studierenden aufteilt. Gemeinsam ist ihnen, dass die Gestik mittels Kinect an die Anwendung als Eingabe weitergereicht, was einer entsprechenden Reichweite bedarf und als Anzeige über den Beamer ausgegeben wird. Sofern das Publikum nicht in die Präsentationssteuerung miteinbezogen wird, verläuft die Kommunikation zwischen Publikum und Anwendung einseitig über die Anzeige der Präsentation durch den Beamer.

Während der Nutzung von Anwendungssystemen entwickeln Benutzer eine geistige Vorstellung von der Bedienung des Systems. Die Qualität, die Transparenz und die Bedienbarkeit sind abhängig davon, wie gut ein solches mentales Modell mit dem konzeptionellen Modell der Systemdesigner auf der einen Seite und mit dem technischen Modell der Entwickler auf der anderen Seite verträglich ist. Das mentale Modell der Benutzer ist schwer zu fassen bzw. zu modellieren, da jeder Benutzer sein persönliches, sich ständig änderndes Modell von der Bedienung des Systems hat [Herczeg 2009].

Der Präsentierende als Benutzer bildet während der Präsentation aus seinen Empfindungen heraus Ziele, die er in Aktionen bzw. in Form von Gesten an die Kinect als Eingabegerät weitergibt. Die Kinect wiederum wandelt die Aktionen in Signale bzw. Daten um, die sie ihrerseits an das Anwendungssystem weiterreicht. Die systemische Transferleistung besteht nun darin, diese Signale in Systemziele umzuformulieren und diese unter Zuhilfenahme des Beamers an den Benutzer auszugeben. Die Interpretierung der Systemausgabe ist Teil der menschlichen

Transferleistung, die den Zyklus der Mensch-Computer-Interaktion [Bowman et al. 2005] komplettiert.

Aus diesem Zyklus der Mensch-Computer-Interaktion geht hervor, wie elementar die Einbeziehung des Benutzers für die Akzeptanz und die Bedienbarkeit ist, da die Kenntnisse und Fähigkeiten die Dialog- und Interaktionsmöglichkeiten der Schnittstelle bestimmen [Heinecke 2012]. Ansonsten ist für den Benutzer mit höherem Aufwand oder einer fehlerhaften Bedienung des Systems zu rechnen, was wiederum in Frustration oder gar Ablehnung seitens des Benutzers resultiert [Herczeg 2009]. Aus diesen Gründen heraus werden im Folgenden potentielle Benutzer mittels einer Datenerhebung einbezogen.

3 EINBEZIEHUNG DER BENUTZER – QUALITATIVE DATENERHEBUNG

Dix et al. beantwortet die Frage, mit Hilfe welcher Methodik der Benutzer einzubeziehen ist wie folgt: “It is hard to get yourself inside someone else’s head, so the best thing is usually to ask them.” [Dix et al. 2004] Lazar et al. sind ebenfalls der Auffassung, dass Interviews während des Prozesses zum Sammeln der Anforderungen von unschätzbarem Wert sind, um ein Verständnis davon zu erlangen, was die Bedürfnisse und Anliegen der Benutzer tatsächlich sind [Lazar et al. 2008]. Cooper et al. bevorzugen dabei qualitative Methoden, da ihrer Meinung nach lebensnahe Einblicke und derart tief begründete Kenntnisse nicht von einer quantitativen Marktumfrage bezogen werden kann, sondern dafür nur qualitative Forschungsmethoden geeignet sind [Cooper et al. 2007].

Zur Generierung von Erkenntnissen wird daher das teilstrukturierte Interview als Methode der qualitativen Sozialforschung gewählt. Für den Leitfaden werden auf zu enge Fragen oder bereits existierende Tools vermieden und anstelle davon breite Fragen über aktuelle Praktiken, Ziele, Frustrationen und Anliegen der Benutzer gewählt [Lazar et al. 2008]. So gilt es herauszufinden, welche Ziele der potenzielle Benutzer beim Präsentieren verfolgt und welche Handlungen er dafür vollzieht, um seine Ziele zu erreichen. Relevant ist dabei, inwiefern ihn die aktuelle Präsentationssteuerung unterstützt und ob ihn daran etwas stört bzw. frustriert. Welche Erwartungen er daraus folgernd formuliert und wie für ihn die Präsentationssteuerung der Zukunft aussieht, ist für die Ermittlung der Anforderungen zusätzlich von Bedeutung [Cooper et al. 2007].

Bei der konkreten Auswahl der zu befragenden Benutzer ist zu bedenken, dass ein repräsentativer Querschnitt an Nutzern zu betrachten ist, da dies eine entscheidende Bedeutung für die Qualität der Ergebnisse hat. Bei qualitativen Interviews eignen sich Personen mit möglichst unterschiedlichen Perspektiven hinsichtlich des Themas, um eine hohe Variation an Antworten zu erhalten [Moser 2012]. Im Folgenden werden Personen, die ähnliche Eigenschaften aufweisen, zu Benutzerklassen zusammengefasst [Herczeg 2009].

Nach Herczeg können Benutzer vielfältig klassifiziert werden, für ihn optimal aber hinsichtlich organisatorischer Rolle, Erfahrungsstand, Marktsegmenten oder auch ethnografischer Eigenschaften wie dem Lebensstil und dem Lebenskontext [Herczeg 2009]. Übertragen auf den Kontext ist die Klassifizierung nach organisatorischer Rolle (Professor/ Studierende), Präsentationsstil (sachlich-informativ/ belustigend-unterhaltend) und Unterschiede in der Persönlichkeit (introvertiert/ extrovertiert) plausibel. Hierbei werden jeweils zwei Ausprägungen zugelassen, um die Anzahl der Benutzergruppen generell überschaubar zu halten.

Neben diesen Differenzierungsmerkmalen existieren Weitere wie Altersunterschiede oder die Diskrepanz zwischen der Arbeitsweise von weiblichen und männlichen Benutzern. Diese und viele weitere Faktoren sind von Fall zu Fall individuell zu entscheiden [Zühlke 2012]. Im Fall der Präsentationssteuerung wird bei der Auswahl der Teilnehmer das Alter sowie Geschlechtsunterschiede einbezogen, bei der Klassifizierung der Benutzer darauf aber verzichtet.

Auf die Kombination introvertiert als Persönlichkeitsmerkmal und dem Präsentationsstil belustigend-unterhaltend wird aufgrund der Annahme verzichtet, dass diese Benutzergruppe sehr klein ist. Insgesamt bilden diese Benutzerklassen die Grundlage für die Auswahl der Interviewteilnehmer (siehe Tab. 1).

Klasse	Rolle	Persönlichkeit	Präsentationsstil
A	Professor	extrovertiert	belustigend-unterhaltend
B	Professor	extrovertiert	sachlich-informativ
C	Professor	introvertiert	sachlich-informativ
D	Studierende	extrovertiert	belustigend-unterhaltend
E	Studierende	extrovertiert	sachlich-informativ
F	Studierende	introvertiert	sachlich-informativ

Tab. 1. Klassifizierung der Benutzer.

Im Rahmen der Benutzerbefragungen wurden pro Benutzerklasse mindestens eins bis zwei Personen interviewt. Auf Basis der gesammelten Daten aus den Interviews werden im Folgenden die Analyse der Nutzeranforderungen und Aktivitäten vorgenommen.

Qualitative Interviews erlauben keine statistische Auswertung. Es ist wahrscheinlich, dass bei weiteren Befragungen eine Verschiebung der Antworten stattfindet. Ziel ist es, ein grundsätzliches Verständnis der Ziele und Bedürfnisse der unterschiedlichen Benutzergruppen zu erlangen und damit die Qualität der Festlegung der Anforderungen zu erhöhen.

4 ERGEBNISSE - ANALYSE DER NUTZERANFORDERUNGEN UND -AKTIVITÄTEN

Im Folgenden werden die Ziele der Befragten vorgestellt und davon ausgehend die Bedürfnisse. Hinsichtlich der Benutzerziele stellt sich bei der Auswertung der Interviews heraus, dass das Vermitteln von Wissen als oberstes Ziel angesehen wird. Dies entspricht dem Ziel der Lehre. Als Unterziele ist es für die Befragten zielführend zu begeistern, motivieren sowie zu unterhalten und damit die Präsentation interessant zu gestalten. Von Studierenden wird einstimmig das Ziel angeführt, den Inhalt verständlich zu präsentieren. Die Professorenseite betont die Wichtigkeit der interaktiven Gestaltung der Vorlesung als Ziel und Methodik. Zuletzt wird von beiden Rollen das Ziel der Verankerung der Inhalte beim Zuhörer angegeben.

Während der Interviews wurden die Teilnehmer neben der Frage nach den Zielen gefragt, durch welche Maßnahmen sie diese erreichen. Aus diesen Antworten heraus, sind im Folgenden Aktivitäten abgeleitet, welche durch die Präsentationssteuerung zu unterstützen ist. Eine der Hauptaufgaben während einer Präsentation im Hochschulbereich stellt das Navigieren dar. Neben den typischen Aktivitäten beim Navigieren, wie dem Vor- und Zurückblättern, sind laut der Befragten neue Möglichkeiten denkbar wie dem Anzeigen einer Übersicht, dem Wechseln zwischen mehreren Programmen, das Springen zwischen den Folien und dem Verzweigen und Gabeln des Folienflusses.

Für die anschauliche Aufbereitung des Inhalts als zweite Hauptaufgabe ist für die Befragten die Strukturierung ein wesentlicher Aspekt. Deuten, Ausblenden, Aufleuchten, Einkreisen oder Unterstreichen zum Markieren sind typische Aktivitäten beim Hervorheben eines Kernaspektes oder einer relevanten Information. Daneben ist das Aufzählen ebenso eine häufig verwendete Tätigkeit, welche z. B. durch das Auf- und Zuklappen von Stichpunkten oder Sektionen seitens

der Systemsteuerung unterstützt werden kann. Neben der Konzentration auf Kernaspekte ist die Übersicht ein weiteres Mittel der Strukturierung. Sie dienen Lernenden als Einführung in Neues und machen diese mit der zentralen Aussage vertraut [Mietzel 2001]. Dazu zählen der rote Faden, das Anzeigen der wichtigsten Folien, die Anordnung der Folien in einer Baumstruktur, das Zusammenfassen sowie das in Beziehung setzen, indem Inhalte verknüpft, gruppiert und daraus ein Netzwerk gebildet wird. Die Skizzierung eines Beziehungsgeflechtes besitzt eine verständnis- und behaltensfördernde Wirkung. Die Wirkung wird zusätzlich verstärkt, wenn das Verknüpfen vom Lernenden selbst ausgeführt wird, das dann in den Bereich der Interaktion mit dem Publikum fällt [Mietzel 2001].

Der zweite wesentliche Aspekt der anschaulichen Aufbereitung des Inhalts wird von Anknüpfungspunkten ermöglicht wie z. B. durch die Verwendung von Beispielen, Medien und Notizen. Die Integration von Beispielen in die Präsentation gestaltet diese anschaulich, wobei auf aktuelle Themen, Anekdoten oder auch die Praxiserfahrung des Präsentierenden zurückgegriffen werden kann. Die Einbindung von Medien wie Diagramme, Bilder, Comics und Simulationen bieten neben der anschaulichen Aufbereitung auch eine Attraktivität der Präsentation für den Zuhörer. Diese Medien sind durch geeignete Aktionsmöglichkeiten zu unterstützen wie das Erlauben des Zooms, der Rotation, der Verschiebung und der Animation. Dadurch lassen sich zusätzlich Simulationen kreieren wie bspw. die Besichtigung eines Lagers für Logistik-Vorlesungen.

Für die Einbettung eines Videos eignen sich die Aktionen Starten, Vor- und Zurückspulen, Pausieren, Beenden und das Regeln der Lautstärke. Mit Hilfe von Annotationen können der Präsentation Erweiterungen hinzugefügt werden. Hierfür sind systemseitig Handlungsmöglichkeiten, wie das Erstellen, das Speichern, das Anzeigen und Ausblenden, für den Benutzer zu realisieren.

Als dritter Aspekt des Präsentierens ergab sich aus den Interviews die Interaktion mit dem Publikum in Form des Dialogs, des Feedbacks und der Übungen. Diskussionen, Mindmapping und Stellen von Fragen als Dialog können wiederum durch geeignete Aktionsmöglichkeiten unterstützt werden. Beim Mindmapping sind auf die generischen und bereits beschriebenen Aktionen zurückzugreifen wie z. B. das Gruppieren, Verknüpfen und Anfertigen von Notizen. Diese Aktionen sind nicht nur dem Präsentierenden anzubieten, sondern auch den Zuhörern, vorausgesetzt das System unterstützt eine Mehrbenutzerbedienung. Ansonsten sind die Aktionen indirekt über den Präsentierenden unter Anleitung des Zuhörers auszuführen.

Das Stellen von geeigneten Fragen zur Aufarbeitung des Inhalts regt eine Verarbeitung beim Zuhörer an, veranlasst dabei die Aktivierung von Hintergrundwissen und fördert die Integration in bereits vorhandenem Wissen. Diese Prozesse sind wiederum vom System geeignet zu unterstützen, indem die Aktionen zum Hervorheben durch das Publikum hier Einsatz finden.

Übungen als Einzel- oder Gruppenarbeit finden bei klassischen Vorlesungen sowie denen mit seminaristischem Charakter Anwendung. Das Bediensystem bietet die Möglichkeit auch diesen Prozess zu erleichtern, indem es Hilfestellungen beim Gruppen bilden, Themen verteilen und Ergebnisse vorstellen bietet. Diese greifen wiederum auf bereits erwähnte Aktionen zurück oder fügen neue Aktionen wie das Zählen beim Bilden von Gruppen hinzu. Feedback in Form von Umfragen und Bewertungen sind zusätzlich denkbar und mithilfe bereits besprochener Aktivitäten realisierbar.

Die Festlegung der Anforderung für Anwendungssysteme hat oftmals nicht viel mit dem zu tun, was Benutzer tatsächlich brauchen oder wünschen. Viel mehr basieren sie auf Marktanalysen, die Auskunft darüber geben, was gebraucht wird, um wettbewerbsfähig zu bleiben und was Benutzer kaufen werden. Um ein Verständnis der Bedürfnisse potenzieller Benutzer des Systems zu erlangen, wurden sie auch danach befragt, was ihnen an der heutigen Präsentationssteuerung stört und grundsätzlich fehlt.

Nach Angabe der Befragten besitzt der Presenter als vergleichbare Präsentationssteuerung im Grunde wenige Funktionen, was ihn auch wieder sehr zuverlässig und einfach in der Bedienung macht. Sie empfinden es als Nachteil, dass sie gezwungen sind, für die Steuerung ein Gerät in

der Hand halten zu müssen, das zudem Mängel in der Handlichkeit aufweist. Daneben kritisieren sie, dass das Klicken des Presenters häufig zur Geste des Präsentierenden wird. Daher bevorzugen sie eine Präsentationssteuerung, die dem Publikum verborgen bleibt bzw. natürlicher wirkt. Vereinzelt wird angemerkt, dass das Umschalten einer Folie zeitversetzt geschieht, was die Unsicherheit während der Präsentation erhöht.

Die Interaktion mit dem Publikum ist auch Thema hinsichtlich der unerfüllten Bedürfnisse der Benutzer, da sie sich mehr Funktionalität hierfür wünschen. Ihnen fehlt die Unterstützung zum gemeinsamen Entwickeln, um den Zuhörer stärker in die Vorlesung einzubinden. Darüber hinaus besteht der Wunsch nach mehr Flexibilität bei der Abfolge der Präsentationen. So sind klassische Präsentationen mit Power-Point erstellt und basieren auf einer sequenziellen Abfolge. PowerPoint bietet für dieses Bedürfnis bereits eine Reihe von Funktionen an, bei denen allerdings wieder auf die Tastatur und Maus zurückgegriffen werden muss. Schließlich bemängelt die Mehrheit der befragten Nutzer, dass die Einbettung von Medien wie etwa Videos sowie deren Steuerung schwierig und oftmals nur über die Tastatur möglich ist.

5 FAZIT

Neben den gewonnenen Anforderungen und Aktivitäten seitens der Nutzer ist festzuhalten, dass grundsätzlich zwischen der Präsentationsanwendung und der Steuerung der Präsentation zu unterscheiden ist. Die Präsentationssteuerung ruft lediglich die Funktionen der Präsentationsanwendung auf, realisiert sind diese aber innerhalb der Präsentationsanwendung und müssen von dieser auch bereitgestellt werden. Durch eine neue Form der Präsentationssteuerung wie der berührungslose Gestensteuerung werden auf der Seite der Präsentationsanwendung weitere Funktionen überhaupt erst möglich. Diese Funktionen sind dann aber nicht mit vorherigen Präsentationssteuerungen nutzbar. Zusammenfassend lassen sich als Ergebnis aus den Interviews sowie der Betrachtung bereits existierender Systeme zahlreiche funktionale Anforderungen an die Präsentationsanwendung zusammentragen, die im Besonderen für die Präsentationssteuerung mittels Kinect relevant sind (siehe Abb. 2).

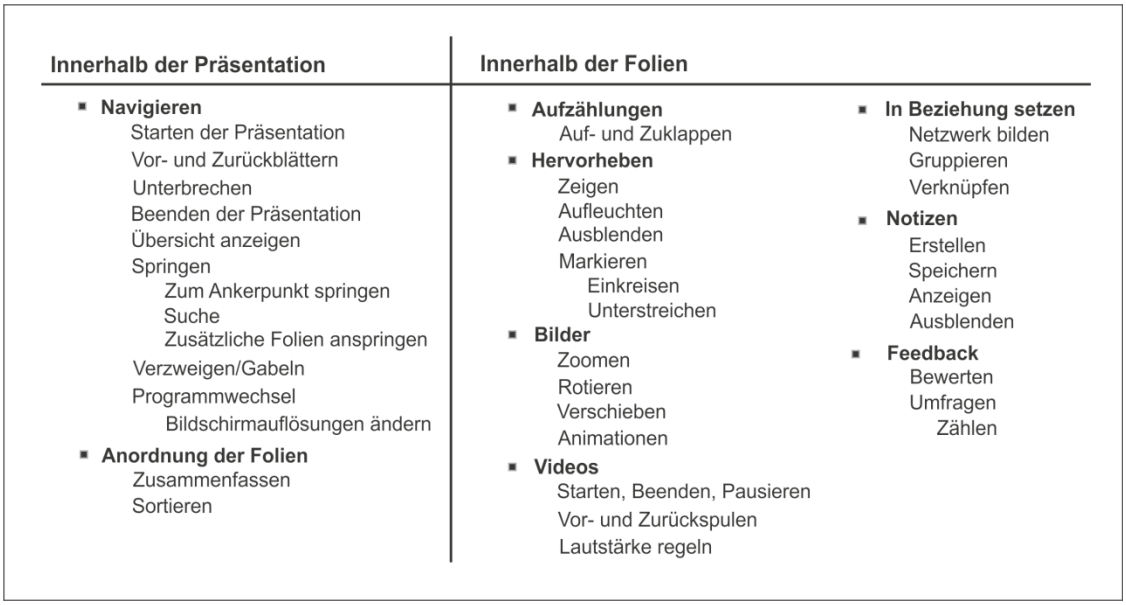

Abb. 2: Funktionale Anforderungen an die Präsentationsanwendung.

Zur besseren Übersicht sind die funktionalen Anforderungen in zwei Bereiche aufgeteilt, wobei unterschieden wird, ob die Funktionen innerhalb der Präsentation verfügbar sind oder sich auf den Bereich innerhalb einer Folie beschränken. Im Hinblick auf die Integration des Publikums in die Steuerung einer Präsentation bspw. für Feedback eröffnen sich völlig neue Chancen, aber auch Risiken, die es innerhalb weiterer Forschungsarbeiten herauszuarbeiten gilt. Auf die Zusammenfassung der nichtfunktionalen Anforderungen an die Präsentationsanwendung wird hier verzichtet, da nicht die Präsentationsanwendung, sondern die Steuerung im Fokus dieser Arbeit steht.

Die Unterscheidung und die damit verbundene Abhängigkeit zwischen Anwendung und Steuerung haben Auswirkungen auf die Anforderungen der gestenbasierten Präsentationssteuerung. So werden der Steuerung nur die Anforderungen zugeordnet, die diese betrifft wie die Erkennung der Gesten selbst. Im Vergleich zur Tastatur wird an diese nur die funktionale Anforderung der Erkennung des Tastendrucks gestellt.

Für die Anforderungen der Präsentationssteuerung werden die Bedürfnisse der Benutzer als nichtfunktionale Anforderungen übernommen. Zu den funktionalen Anforderungen der Präsentationssteuerung mittels Kinect zählen die Erkennung von Gesten und Sprache, die Individualisierung in Form einer Konfiguration, um die Gesten auf den Benutzer individuell zu zuschneiden, sowie die Mehrbenutzerbedienung (siehe Tab. 2).

	Nichtfunktionale Anforderung	**Funktionale Anforderung**
Präsentationssteuerung		
- Kinect		
	Entsprechende Reichweite	Gestenerkennung
	Zuverlässig	Spracherkennung
	Intuitive Bedienung	Individualisierung
	Schnelle Erkennung	Mehrbenutzerbedienung
	Kostengünstig	
	Geringe Einarbeitungszeit	
	Einfache Wartung	

Tab. 2. Anforderungen an die Präsentationssteuerung.

Abschließend lässt sich festhalten, dass berührungslose Gesten einen wertvollen Beitrag zum Repertoire der Interaktionstechniken ergeben, aber um aus ihnen den bestmöglichen Nutzen zu ziehen, bedarf die Gestaltung von Raumgesten, wie auch anderen Techniken zuvor, den nötigen Prozess einer langwierigen Entwicklungszeit unter Einbeziehung von potenziellen Benutzern zur Evaluierung und schrittweisen Optimierung [Dix et al. 2004]. Darüber hinaus braucht es neben der richtigen Auswahl und Gestaltung auch Zeit eine entsprechende räumliche Infrastruktur in Bezug auf mehrere Kinects zu entwickeln, das Feedbackverhalten speziell an Raumgesten anzupassen, ein geeignetes Fehlermanagement auszuarbeiten und die Menüsteuerung für berührungslose Gesten auszurichten [Norman 2010].

REFERENZEN

BAUDEL, T. UND BEAUDOUIN-LAFON, M. 1993: Charade - remote control of objects using free-hand gestures. *Communications of the ACM - Special issue on computer augmented environments - back to the real world*. S. 28-35.

BOWMAN, D.A., KRUIJFF, E., LAVIOLA, J.J. UND POUPYREV, I. 2005. 3D User Interfaces: Theory and Practice. *Boston*.

COOPER, A., REIMANN, R. UND CRONIN, D. 2007. About Face 3 – The Essentials of Interaction Design. *Indianapolis/Canada*.

DIX, A., FINLAY, J., ABOWD, G.D. UND BEALE, R. 2004. Human-Computer-Interaction. 3. Auflage. *Harlow et al.*

FOURNEY, A., TERRY, M. UND MANN, R. 2010. Gesturing in the Wild: Understanding the Effects and Implications of Gesture-Based Interaction for Dynamic Presentations. *Proceedings of the 24th BCS Interaction Specialist Group Conference*. S. 230-240.

HEINECKE, A. M. 2012. Mensch-Computer-Interaktion: Basiswissen für Entwickler und Gestalter. 2. überarbeitete und erweiterte Auflage. *Berlin/Heidelberg*.

HERCZEG, M. 2009. Software-Ergonomie - Theorien, Modelle und Kriterien für gebrauchs-taugliche interaktive Computersysteme. 3. Auflage. *München*.

LAZAR, J., FENG, J. H. UND HOCHHEISER, H. 2010. Research Methods in Human-Computer-Interaction. *Chichester.*

MELGAR, E. R., DÍEZ, C. C. UND JAWORSKI, P. 2012. Arduino and Kinect Projects - Design, Build, Blow Their Minds. *New York.*

MOSER, C. 2012. User Experience Design - Mit erlebniszentrierter Softwareentwicklung zu Produkten, die begeistern. *Berlin/Heidelberg*.

NIELSEN, M., STÖRRING, M., MOESLUND, T. B. UND GRANUM, E. 2004. A Procedure for Developing Intuitive and Ergonomic Gesture Interfaces for HCI. *Gesture-Based Communication in Human-Computer Interaction Volume 2915/2004*, S. 409-420.

NORMAN, D. A. 2010. Natural User Interfaces Are Not Natural, 2010. *interactions*. S. 6-10.

O. V. 2009. Handbuch vom Professional Presenter R800.http://www.logitech.com/repository/1400/pdf/26093.1.0.pdf. Zuletzt aufgerufen am 04.04. 2013.

TSE, E., MARENTETTE, L. ,AHMED, S. I., THAYER, A., HUBER, J., MÜHLHÄUSER, M., KIM, S. J. UND BROWN, Q. 2012.: Educational Interfaces, Software, and Technology. *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems*. S. 2691-2694.

ZÜHLKE, D. 2012. Nutzergerechte Entwicklung von Mensch-Maschine-Systemen. 2. neu bearb. Auflage. *Berlin/Heidelberg*.

Native versus HTML5 – where's mobile programming heading to?

PATRICK MENNIG
University of Applied Sciences Wuerzburg-Schweinfurt

The market for mobile applications has grown from a few pre-installed applications on mobile phones to giant stores for different smartphone platforms. Although for smartphones, Android has a much stronger market share than iOS or Windows, a developer can hardly refrain from offering its applications for all operating systems. For him this represents a significant overhead, since the different platforms are addressed with different programming languages. This means concretely that multiple programming effort, multiple tests and calls for a broad expertise of Java, Objective-C and .NET are required. Web technologies known as HTML5, allow to develop applications that are running on all mobile operating systems, without having to be repeatedly developed. Thanks to modern frameworks web technologies are on a level with native development and the approach needs to be carefully considered. This paper investigates the benefits and risks of different development approaches with the help of certain criteria.

Categories and Subject Descriptors: Mobile and Ubiquitous Computing

Additional Key Words and Phrases: Mobile Applications, Apps, Web Applications, Cross-Platform Application Framework, PhoneGap, native App

1 INTRODUCTION

Mobile applications are part of many areas of daily life. The intensive usage of smartphones and tablet computers increasingly permeates the private and business life. Apps advanced from the beginning, which is marked by the first smartphone with a multitouch screen – the Apple iPhone. These small programs are loaded on the smartphone and tablet to enable a variety of actions a user may perform. The typical mindset for any end-user is as induced by the advertising slogan by Apple. It can be transferred as: "If you want to do something, there's an app for it". It remains an open question where these apps come from. The techniques for app development and their differences shall be illuminated further, comparing the different development approaches based on a certain set of criteria.

2 THE MOBILE APPLICATION MARKET

Mobile applications are - as introduced - an important aspect of mobility. Before any smartphones, applications have already been installed on mobile phones. One prominent example is the game "Snake" which was to be found on the phones from Nokia [Alby, T. 2008]. Other than today, the user was dependent on preinstalled applications. It was only in few circumstances, or with huge efforts possible, to install single applications manually. Furthermore, the operating concept of mobile phones was different from today's smartphones and tablet computers. Users controlled early mobile devices by solid mounted keys or touchscreens reacting

Author's address: Patrick Mennig, University of Applied Sciences Wuerzburg-Schweinfurt, Würzburg, Germany.
www.fhws.de.

to pressure (resistive touch technology). The introduction of the first iPhone by Apple in 2007 changed the situation fundamentally. One aspect has been the preinstalled and manifold applications, which resembled today's apps and covered important activities. Additionally it was possible to control this smartphone with the fingers. The touchscreen is reacting to magnetic fields of human skin. The interaction happened with clearly less pressure and was easier to perform for the users. More than one point of interaction (i.e. finger) can be detected which allows so called multitouch gestures.

Advertised was this behavior as the connection of an iPod (a portable music player with a hard drive), a mobile phone and an internet communicator. The slogan describes the impact of the first "real smartphone" insufficient. The concept of small applications within a comprehensive operating system fascinates both, users and developers. The latter have been excluded in the beginning. A commonly available development environment (IDE) for iPhone applications was not available. In August of 2008, Apple published version 3.1 of its IDE Xcode, including the iPhone SDK. Paired with the app store, introduced in early 2008, the baseline for today's ecosystem of small apps from different app stores was built.

In October of 2008 the first smartphone with the operating system "Android" hit the market. This OS is developed and distributed by the Open Handset Alliance under leadership of Google Inc. Contrary to Apple's iOS, it features a visibly more open architecture. Important to highlight is that different vendors of mobile phones can sell their devices with adjusted versions of Android. Contrary, the operating system iOS can only be found on devices sold by Apple. Since October of 2010, a third important actor in the smartphone market is Windows with its OS "Windows Phone". It shows the same concept of small apps that the user may load from the store. Other vendors like BlackBerry (former ResearchInMotion) focus on less typical smartphone functionality and should be excluded from the examination.

The result of this evolution is a smartphone and tablet market that is contested by more than one vendor. Visible differences are found between phones and tablet computers. According to IDC, the market of tablet operating systems in 2012 is partitioned as follows: Apple leads with its operating system iOS with 53.8 percent market share, followed by the OS Android with 42.7 percent. With large distance follows the mobile versions of Windows, reaching 2.9 percent. It can be said that iOS and Android dominate the market. Microsoft has the potential to reach higher market shares with its new system, Windows 8.

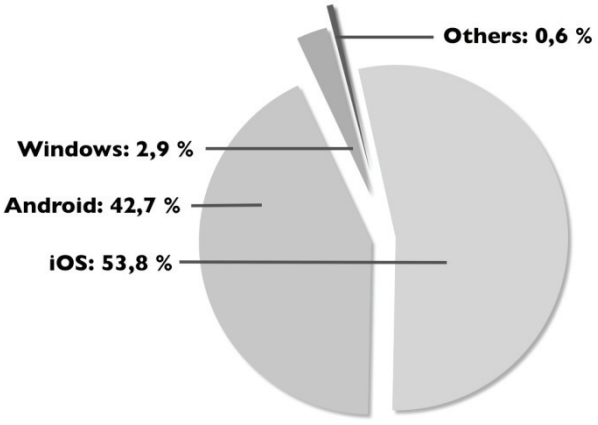

Figure 1: Market share of mobile OS – tablets.

Source: Author's representation, based on IDC, 2013a.

Figure 1 illustrates the tablet computer market shares [IDC, 2013a]. Assuming a developer is only targeting the still dominant iPads with iOS, he is only reaching half of the possible target group. The same applies for Android-only development.

The market of smartphone operating systems shows a different distribution. Android dominates clearly with 70.1 percent. Apple's iOS follows on the second rank with 21.0 percent, behind lies BlackBerryOS with 3.1 percent and Windows with 2.6 percent [IDC, 2013b]. A development for Android may reach circa three quarters of the potential market. The smartphone market is illustrated in Figure 2.

Considering the shown market shares, it follows that when developing an application, more than one target operating system should be chosen. An app that shall reach a market as large as possible cannot be limited to one platform. Rather, it is recommended to create an app that is capable of running on both, smartphones and tablet computers. In case all major operating systems are supported, the greatest part of the potential market is reached. The technological restrictions that appear when choosing a certain development approach for mobile applications, are subject to the following chapters.

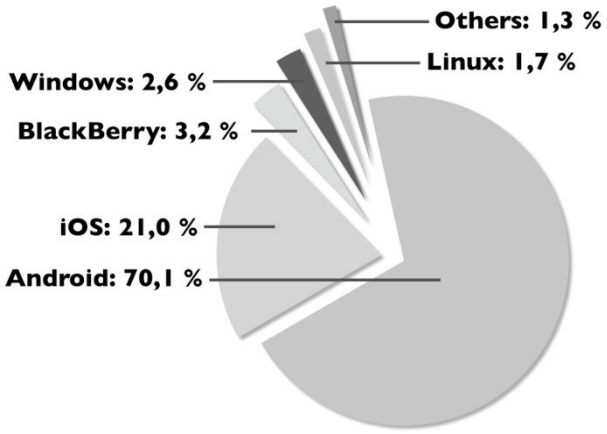

Figure 2: Market share of mobile OS – smartphones

Source: Author's representation, based on IDC, 2013b.

3 MOBILE APPLICATION DEVELOPMENT

Similar to their pendants on normal computers, mobile applications have to be developed with certain programming languages. It applies here, that these differ depending on the targeted platform. Standard iOS apps are developed with the programming language Objective-C. Android apps are to be developed with a certain version of JAVA. For Windows applications, the .NET languages are to be used (C#, C++) [Charland, A. and Leroux, B. 2011]. This includes the creation of the application logic with given tools and methods, as well as the user interface. This is done in different ways. A Mac computer with the development environment Xcode is necessary for iOS development, while Windows apps are created with Visual Studio (which is only running in Windows machines). Android applications can be developed under Mac OS or Windows, for example with the IDE Eclipse. Necessary are only the free available Android Developer, which can be included as a plug in into the development environment. An application created in such a way is called "native". It is written with platform specific code and can only be used on the specific one without adjustments. Porting it to another operating system assumes huge adaptions. Not

only the syntaxes of used languages differ from each other but the semantics, available classes, libraries and the program structure, depending on the targeted operating system.

Apart from native development, another approach called web development exists. Following it, a developer creates so-called web apps. These use the operating systems browser as a runtime environment and are runnable on all mobile operating systems without any installation process. The programming languages to create web applications are HTML, CSS and JavaScript. Those allow platform independent development.[4] The application can be opened by accessing a specific website – nothing else are web applications. A specifically developed and adjusted website mocking a native application in design and functionality [Willnecker, F. et. al. 2012]. With particular techniques, these web apps can also be used without an Internet connection, as long as the user has saved them on the device once and not cleared the browser's cache. In iOS, this is achieved via the function "add to home screen".

A third way of developing mobile applications is to combine native and web development. Connecting a container application written in platform specific code with a web app containing the application logic and the user interface leads to a program referred to as "hybrid". In the most extreme case, the native app contains only a single browser window in full screen, running the web app [Franke, F. and Ippen, J. 2012]. The corresponding element is called UIWebView in iOS and WebView in Android and Windows.

As the different development approaches for mobile applications have been shown, the question remains which of these – depending on the applied criterion – offers advantages and disadvantages. The criteria "Content vs. Experience", "Performance", "APIs", "Distribution" and "Development Cost" are to be considered. Those are only a subset of possible criteria (e.g. [Frank, AM 2011], [Charland, A und Leroux, B. 2011], [Corral, L. et. al. 2011]) and represent those with a clear difference in results.

4 ANALYSIS OF CRITERIA

Above-mentioned criteria should be considered hereafter. Interesting for each are the differences between native, hybrid and web applications. The idea behind the examination is a given firm, which is facing the decision which development approach to choose.

4.1 Content vs. Experience

The first step is to decide what one wants to focus on, when developing an application. Two extremes lie in the mere representation of content and the communication of an experience. As long as the goal is to display specific data and information, it is possible speak of a focus on content. This is especially interesting in the area of enterprise applications. From a developers perspective this focus is possible with all three approaches, i.e., native, hybrid and web. There are no particular requirements on the design options.

This is different, however, with a focus on experience. This term is very vague and hardly palpable. It can be described with the feeling that a user has, when using the application. Experience is influenced, inter alia, by the design, the haptic and responsiveness of the application. As the user manipulates the app with his own fingers, he has a very direct interaction. However, this is affected by sluggish or lack of feedback to touch, due to delays in moving content or too functionally designed user interfaces. A very good example of an app that wants to convey

[4] The browser is used as a container. Hence the platform independence is only restricted by the browsers capabilities. As long as the browsers meet the HTML specifications, independence is given. Adjustments and changes to adapt to different browsers and versions are often inevitable.

an experience is the calendar on the iPad under iOS. Although much content is presented, the frame as well as the background are lovingly designed and give the user the feeling that they have a real calendar of leather and paper in hand. Even details such as curved edges, demolition of old calendar pages in the month view or the shadows when turning pages have been minded.

Native applications have a slight advantage in terms of this criterion because they allow a direct access to system resources and therefore can react very quickly to user input. Furthermore, they are using the operating system's control elements, thus enabling a seamless experience across the entire system. The advantage in resource usage allows for more advanced and skeuomorphic user interfaces due to more power. Basically this also possible with web languages, but it requires huge additional effort in creating the user interface, the transitions between screens and feedback on user interaction. Existing frameworks such as JQueryMobile and SenchaTouch help, but can not offset all the disadvantages of web and hybrid applications. When focusing on communicating a strong experience, native development is the better approach, as it allows reaching quality results with less effort.

4.2 Performance

Depending on the use cases, which the application is based on, different requirements towards the performance of an application arise. If only a certain number of data is shown without being calculated intensively, the app has no special requests to the performance. However, once large amounts of data are displayed, emphasis must be placed on the observance of performance limitations.

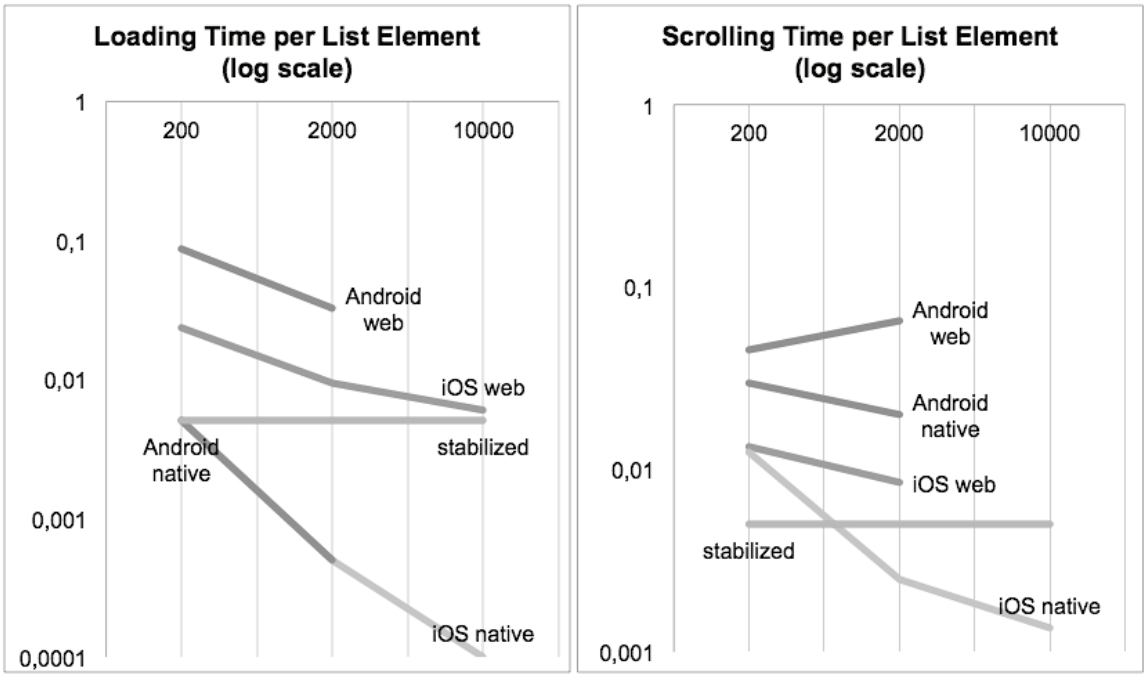

Figure 3: Loading and scrolling time in seconds per list item on mobile devices.

Source: Author's representation, based on [web & mobile developer 2013].

Performance issues have been proven in an experiment published in the journal “web & mobile developer”. It based on the assumption that users expect rapid applications that react quickly to touch input. The set-up is formed by two native applications and a web-application. These are displaying 200, 2,000 and 10,000 list items either natively on iOS and Android or in the web browser on iOS and Android. Goal of this test was to measure the time necessary to load and display the list as well as manual scrolling time from top to the bottom. It was expected that both. Loading and scrolling time will increase with more list items. The experiment came to the conclusion that as early as 2000 list items are shown, significant performance losses are felt. It shows, among other things, that the mobile version of Safari, takes 23 times the time to load the list of 2,000 items, compared to the native application. Also in terms of manual scrolling it shows that in the web application it takes three times the time needed for scrolling than in the native application (from the beginning of the list to the end). The consequence is that the app feels slower and less responsive. At 10,000 list items, a display in the web was no longer possible for most browsers [web & mobile developer 2013].

The illustration Figure 3 shows two diagrams, indicating loading and scrolling time divided by the number of list items for iOS and Android, both web and native (different browsers on one platform have been averaged). The time is shown on a logarithmic scale (in seconds to $\log_{10}$). Both diagrams show a stabilized series, assuming a linear growth of time. In case the display of data or scrolling was impossible, no value is marked for the specific series.

This shows a clear advantage for native development in terms of performance. Considering it as an important criterion for a good user experience, one should be careful in the case of hybrid or web-based apps, estimating early the amount of power necessary. In case very large numbers of records are displayed, complex calculations are carried out or complex 2D and 3D graphics are desired, it is often advisable to develop a native app. In a different application, without these requirements, a hybrid or web approach makes sense too.

4.3 APIs

Smartphones are interesting mainly because of their extensive technical capabilities. These include access to special equipment features such as the camera, the module for determining the position or acceleration sensors. This is generally addressed by so-called application programming interfaces (APIs). In principle, the access to these APIs from the native code is no problem. The functions can be used in accordance with the provided hardware of the device.[5]

Web Code, however suffers from limitations in terms of access to device-specific APIs. The specifications for access to them are not yet available for all functions and therefore not implemented. The module to determine the position can be accessed from the web browser, as well as the camera and the photo browser [Firtman, M. 2013]. One possible way to circumvent the limitations is building hybrid applications that can access the native features with certain technical ways from the web container. A framework that supports the developer with them is Apache Cordova (formerly PhoneGap). It provides a JavaScript library with allows addressing the various operating systems uniformly [Apache Cordova 2013]. The following Table 1 shows a sample of mobile browsers capabilities in terms of API access.

[5] For Android devices, the hardware is very different depending on manufacturer and model. It is therefore possible that, although the same operating system is installed, not all APIs are available on all devices. This restriction does not apply to iOS yet, as iPhones and iPads have homogeneous functions.

Platform	**Browser**	**Safari on iOS**	**Android Browser**	**Internet Explorer**	
	Platform	**iPhone, iPad**	**Phones & Tablet**	**Windows Phone 7.5**	**Windows 8**
	Versions tested	**3.2 to 6.1**	**1.5 to 4.2**	**9**	**10**
Feature	**Geolocation**	yes	yes (2.0+)	yes	yes
	Motion Sensors	yes (4.2+)	yes (3.0+)	no	no
	Touchevents	yes	yes (2.1+)	no	yes
	File API	yes (6.0+)	yes (3.0+)	no	yes
	Media Capture	partial (6.0+)	yes (3.0+)	no	no
	Notifications API	no	no	no	no
	Network Information API	no	yes (2.2+)	no	no

Table 1: API coverage of mobile browsers.

Source: Author's representation, based on [Firtman, M. 2013].

4.4 Distribution

Not only the development of an application in itself should be considered. Also, one should keep in focus that an app has to come to the customer. In the case of enterprise apps, it is easier due to separate stores. But if a developer goes for a "typical" app for the end-user, he needs ways and means to expel the product. The manufacturers of the various mobile operating systems provide this in form of their own app stores. On these central platforms, developers can publish and sell their applications without having to worry about hosting and billing. In return, the suppliers keep a given percentage (usually 30%) of the revenue generated [iOS Developer Program 2013] [Android Developer 2012] [Windows Dev Center 2013].

To mention is that only native and hybrid applications can be set in the stores. A web application is running as pure web site and offers no standard way to distribute them commercially. Although some stores for web apps are created already, they are still little frequented. A separate billing system can certainly be created, but this means extra effort in development.

4.5 Development Cost

In addition to the criteria already considered the cost plays a strong role in the development. It is important to keep in mind that native development only targets a single platform. This means that the same application has to be developed several times in case it should be available on multiple platforms (for example iOS and Android). It follows that this also multiplies development costs. In contrast, web development allows that the app, once created, can be used on different operating systems. Hybrid development is a middle way. Since a large part of the application logic is platform-independent, only slightly higher costs for the preparation of native container arise.

The costs of development for the various operating systems are far apart. An examination of the estimates of sample applications for various platforms shows that creating a native iOS application can be almost twice as expensive as developing a web application with similar functionality. The exact figures are not important at this point, however, the trend is interesting. It

clearly shows that native development is more expensive [Frank, AM 2011]. Figure **4** shows this relationship again.

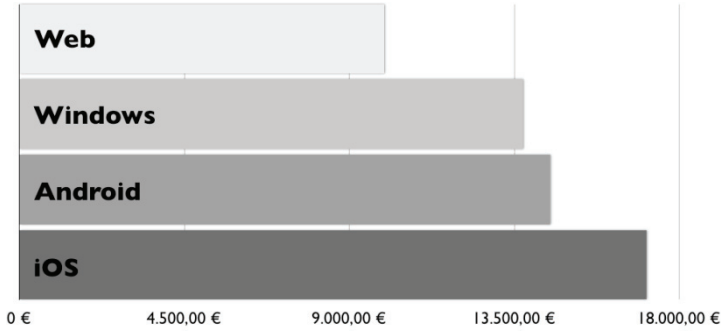

Figure 4: Development cost of a catalogue app.

Source: Author's representation, based on Frank, A. M. 2011.

Furthermore, the majority of developers is developing for iOS and Android (80 percent). These two platforms form the baseline, as the expected revenue on them is the highest. When asked, which system they prefer in terms of development cost, 20 percent of developers answered iOS, compared to 32 percent answering Android. On the other hand, they score iOS the better platform in terms of app discovery and revenue potential (50 & 66 percent iOS vs. 23 & 12 percent Android). Developers, that do not develop for one of the two platforms, earn, on average, only half the revenue of those creating apps for at least one of both. HTML as a platform for app development and deployment is used by 50 percent of developers. 75 percent of all develop for up to three mobile platforms [Developer Economics 2013]. As a conclusion, the sole development cost for web based applications are lower than for Android or iOS but the potential revenue expected is highest on these two native platforms.

5 CONCLUSION

After considering the pros and cons of native, hybrid and web development using various criteria, it's not possible to draw a definite conclusion. On the one hand, native applications offer advantages over web and hybrid applications in factors such as "Content vs. Experience", "Distribution", "APIs" and "Performance". On the other hand, the obvious cost advantage of HTML5 Apps is observed. The set of criteria evaluated in this paper however, does not claim completeness. Certain company-specific conditions can affect the selection. Especially "bring your own device" (BYOD) as a strategy has to solve the problem of integrating various mobile operating systems to support the majority of devices that are used by the employees. The comparison of web and native applications is subject to other studies too, for example [Charland, A und Leroux, B. 2011] or [Corral, L. et. al. 2011]. An outstanding example of web application performance can be found in the app "fastbook" (mimicking the Facebook-app functionality), created by developers from "Sencha" to prove their framework's performance. The web app's performance and user experience exceeds the native app by far [Sencha Fastbook 2012].[6]

[6] The reservation must be made, however, that the native Facebook-app's performance has increased significantly since fastbook was first developed.

In the future, the support of device-specific APIs and the performance of web languages will increase significantly. Especially the emerging HTML5 standard offers many opportunities here. Therefore, the pros and cons, shown in this paper only present a snapshot of the current situation. Nevertheless, currently, it is still imperative to perform a thorough examination of criteria before deciding for a development approach. A pure cost consideration should never be the only basis for a decision.

REFERENCES

ALBY, T. 2008. Das mobile Web. Hamburg, 2008.

ANDROID DEVELOPER 2012. Android Developer Help – Transaction Fees. https://support.google.com/googleplay/android-developer/answer/112622?hl=en. Accessed on 01.06.2013.

APACHE CORDOVA 2013. Apache Cordova is a platform for building native mobile applications using HTML, CSS and JavaScript. http://cordova.apache.org. Accessed on 28.03.2013.

CHARLAND, A UND LEROUX, B. 2011. Mobile Application Development: Web vs. Native. acmqueue Volume 9 Issue 4. New York. 2011.

CORRAL, L. ET. AL. 2011. Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm. Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software. New York. 2011. 181-183.

DEVELOPER ECONOMICS 2013. Developer Economics 2013: The tools report. http://www.visionmobile.com/product/developer-economics-2013-the-tools-report/?image3=1&utm_expid=1534519-18. Accessed on 02.06.2013.

FIRTMAN, M. 2013. Trying to understand HTML5 compatibility on mobile and tablet browsers. http://mobilehtml5.org. Accessed on 28.03.2013.

FRANK, A. M. 2011. Was kostet die Entwicklung von Apps? Ellwangen. 2011.

FRANKE, F./ UND IPPEN, J. 2012. Apps mit HTML5 und CSS3 für iPad, iPhone und Android. Bonn. 2012.

IDC 2013a. IDC Raises Tablet Forecast for 2012 and Beyond As iOS Picks Up Steam, Android Gains Traction, and Windows Finally Enters the Market. http://www.idc.com/getdoc.jsp?containerId=prUS23833612#.UV77q78pG3h. Accessed on 28.03.2013.

IDC 2013b. Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year. http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.UV77J78pG3h. Accessed on 28.03.2013.

iOS DEVELOPER PROGRAM 2013. iOS Developer Program – 3. Distribute. https://developer.apple.com/programs/ios/distribute.html. Accessed on 01.06.2013.

SENCHA FASTBOOK 2012. The Making of Fastbook: An HTML5 Love Story. http://www.sencha.com/blog/the-making-of-fastbook-an-html5-love-story. Accessed on 07.06.2013.

WEB & MOBILE DEVELOPER 2013. Web-Apps oder native Apps. Issue 2/2013.

WILLNECKER, F., ISMAILOVIC, D. UND MAISON, W. 2012. Architekturen Mobiler Multiplattform-Apps. Verclas, S. und Linnhoff-Popie, C. (ed.): Smart Mobile Apps - Mit Business Apps ins Zeitalter mobiler Geschäftsprozesse. Heidelberg, London, New York et. al. 2012. 403-418.

WINDOWS DEV CENTER 2013. Windows Dev Center – Windows Store Apps – App Developper Agreement. http://msdn.microsoft.com/en-us/library/windows/apps/hh694058.aspx. Accessed on 01.06.2013.

Mobile Enterprise Application Development - a Cross-Platform Framework

FLORIAN WOLF, KARSTEN HUFFSTADT
Applied Research Center for Mobile Solutions
University of Applied Sciences Wuerzburg-Schweinfurt

In today's business mobility of information systems is an important topic. Almost no company can get by these days without mobile software solutions. At the same time we are going through a convergence of desktop and mobile, web and native application development. Because of this clear necessarity of mobility and mobile development it is not surprising that there is a large number of different recommendations for software developer using different technologies. This paper is going to point out the basic approach of a cross-platform application framework using the example of PhoneGap and - more importantly - highlights and answers the question if the often promised universal applicability of these frameworks really apply and a one-off source code is enough to deploy an enterprise application.

Categories and Subject Descriptors: Mobile and Ubiquitous Computing

Additional Key Words and Phrases: Mobile Business, Mobile Applications, Apps, Web Applications, Cross-Platform Application Framework, PhoneGap

1 INTRODUCTION

It was already acknowledged in the middle of the last decade that mobile ICT-based capabilities have the potential to increase productivity, efficiency and other important business performance metrics [Gebauer and Shaw 2004; Basole 2005; Nah et al. 2005; Atkins et al. 2006]. Today mobility of information systems is an important business topic. Almost no company can get by these days without mobile software solutions supporting their business.

At the same time we are going through a convergence of desktop and mobile, web and native application development [Mikkonen et al. 2009]. More and more applications run in web browsers and do not necessarily require any explicit installation or native source code. Web applications therefore enable companies to implement software written as a one-off source code in form of the three components HTML5, JavaScript and CSS rendered in a standard web browser. Using that approach, developers do no longer need to support various operating systems writing their applications in native code for iOS, Android, Windows or others. Furthermore web applications are running without the need to be downloaded and installed separately. In addition to that, they can be deployed instantly worldwide. All these factors support companies on their way to easily develop mobile enterprise applications.

What remains is the question of how a complex application for enterprises using only JavaScript, HTML5 and CSS can be supported. With the need of large database handling or access to the API-functionality of the underlying operating systems such as geolocation or accelerometer web techniques reach their limits. Because of that, the so called cross-platform application frameworks were developed which contain native code pieces to interact with the operating systems and run the web application in a webview [Charland and Leroux 2011].

Author's address: Florian Wolf, Karsten Huffstadt, University of Applied Sciences Wuerzburg-Schweinfurt, Germany, http://mobilab.fhws.de.

This paper points out the basic approach of a cross-platform application framework using the example of PhoneGap and - more importantly - highlights and answers the question if the often promised universal applicability of these frameworks really apply and a one-off source code is enough to deploy an enterprise application.

2 CROSS-PLATFORM PARADIGM

Even some years ago Mikkonen et al. [2009] recognized that web applications are the future of mobile application development. Cross-platform techniques like HTML5, JavaScript, CSS and Ajax are the preferred development environment. Willing to reach the goal of making these techniques as powerful and efficient as native software, several development tools and cross-platform frameworks are in development right now and were already developed in the past. We identified a clear distinction of frameworks in two categories.

Browser-centric cross-platform frameworks or so-called user interface frameworks use HTML5-techniques to deploy mobile applications primarily on web browsers. These UI frameworks such as jQuery Mobile or Sencha Touch 2 are HTML5-based user interface systems including a various set of built-in components to develop rich internet applications. With a library of components, widgets, layouts and stylesheets for a high GUI capability already included web applications can be rapidly developed [Yan and Chen 2011; Smutny 2012].

Native-centric cross-platform frameworks use HTML5-techniques - or in some cases a proprietary SDK - to develop mobile applications running on native platforms. Once an application is build up with the framework, it can be deployed on several mobile operating systems without additional expenditure for development. These frameworks such as PhoneGap or Appcelerator can be used in most cases as a hybrid variant of cross-platform frameworks. In this case they are combining a native container with direct access to necessary API-functions supporting a web-based frontend for building web-based user interfaces. These Hybrid-centric cross-platform frameworks provide natively coded containers with for example a simple JavaScript library acting as a bridge between the front end web application and the native API-functionality such as using the camera, the accelerometer or the GPS unit.

All these frameworks share the same idea: to develop a cross-platform mobile application with in most cases web techniques running on different operating systems. Corral et al. [2011] identified in this context an entirely justified evolution in development paradigms in this context. Using these frameworks to create mobile applications for web platforms leads to a web-based multiplatform development paradigm. Due to this significance we analyzed as following described the portability of cross-platform developed web applications using PhoneGap as an example.

3 THE PHONEGAP-THEORY IN DETAIL

Using web techniques is working in a sandbox, which is a jail from lower level APIs [Charland and Leroux 2011]. To access the device storages, sensors and datas this gap has to be bridged. Currently, two ways are available for developers. On the one hand most of today's mobile browsers provide basic API functionality such as geolocation or local storage. Given that the W3C has a Device API Working Group, it can be expected that more APIs reach the browsers in the near future. On the other hand, if developers need API functions, that are not supported directly from the browser, the earlier described hybrid-centric cross-platform frameworks can be used.

The following section exemplarily explains in detail how PhoneGap is working as a hybrid-centric cross-platform framework. For the subsequent analysis of the portability and cross-platform ability we choose iOS as the lead programming platform. Therefore, the first step is to explain the context between iOS and PhoneGap. The basic indicator in iOS is the UIView class which defines a rectangular area on the device screen. This is usually an applications user interface or content that the user is intended to see by the application creator using the device native programming language Objective-C. Some applications not only use native content, but also web content to present information to the user. Therefore the programmer has to create a UIWebView object that has the ability to extend the UIView class to visualize this web content. This is the main principle of how PhoneGap works. Within a native UIWebView there is a UIWebView object, which is able to visualize the HTML pages. The logic of those pages is powered by custom JavaScript code that ensures the functionality of the PhoneGap application. Those JavaScript files are able to make use of the functions provided by the PhoneGap plugin and are therefore able to access all supported APIs.

To get a more substantiated knowledge of how the communication between web based code and native APIs in iOS works, an understanding the basic principles of delegation in iOS is needed. A delegate is based on events and handles the interaction of Objective-C objects and allows them to interact with each other without previously defining dependencies. Using a UIWebView object, the corresponding delegate is called webViewDelegate and inherits from the iOS delegate UIApplicationDelegate. That means whenever web based code is executed within a UIWebView object, the UIWebViewDelegate defines methods that a delegate of this UIWebView object can optionally implement to intervene when web content is loaded. This is where PhoneGap places its own PhoneGapDelegate in between UIApplicationDelegate and UIWebViewDelegate. By doing that, PhoneGap is able to catch specific actions made by a UIWebView object and react as intended in the PhoneGap source code.

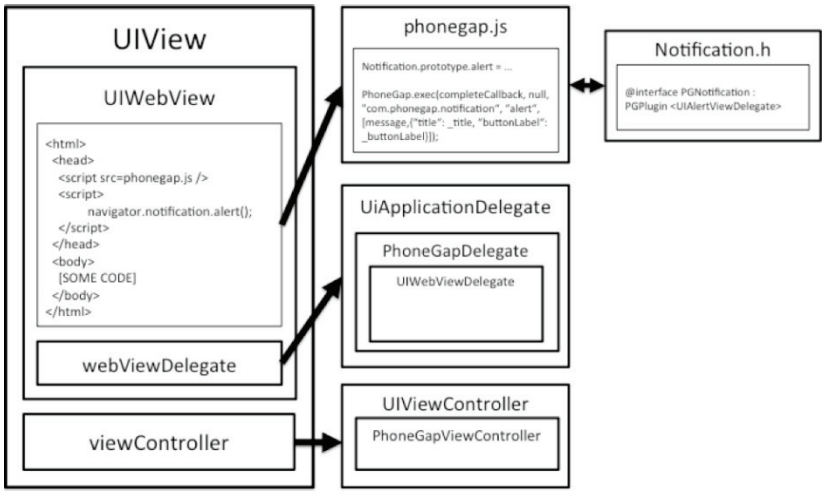

Fig. 1. How PhoneGap works

Similar to that PhoneGap also establishes a PhoneGapViewController as an extension to the iOS UIViewController. This controller provides the fundamental view-management model in iOS.

Using the PhoneGapViewController as a subclass of UIViewController has the advantage of directly controlling the way a PhoneGap based application reacts on certain circumstances other than the native iOS way. Figure 1 provides a quick overview on how it all works together.

4 FRAMEWORK ANALYSIS - A BUSINESS CASE

The main objective was to analyze if the promises made by PhoneGap in matters of multi-system compatibility prove right. It is one of the frameworks big features that a programmer has to create the source code using web technologies only once and then reuse it on any supported platform without the need of further modification.

Therefore a PhoneGap-based application using iOS as lead platform was created, which means that the application was coded in Xcode and the working source was transferred to both the Android and Windows Mobile native programming environments to compile them. In theory, what should be achieved with this approach are three identically working and similar looking applications (see figure 2).

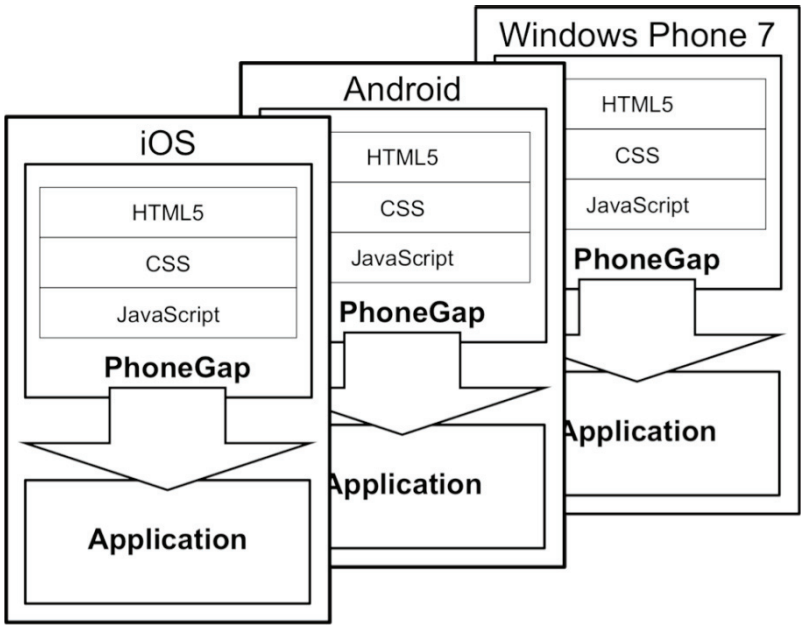

Fig. 2. Building PhoneGap Applications: lead platform iOS

The business case was represented by a logistic function of the maintenance process, especially handling releases and closures of maintenance orders. In terms of functionality, the application uses web services to communicate with an SAP ERP system. The application has to read and write from the ERPs databases to achieve this objective. All error handling and the applications logic is done in JavaScript. Because of the fact that an universal applicability should be considered the business case itself is not in the focus of the analysis and therefore not further described.

In terms of user interface the JavaScript based jQuery Mobile framework was used. Because there are no restrictions within PhoneGap on what kind of JavaScript works (everything should work), this was a possible and valid way to proceed.

JQuery Mobile (jQM) makes use of several HTML5 features to achieve a native like user experience based on general iOS design principles by simulating a native MenuBar and NavigationBar if needed. Another interesting aspect of using jQM is to test the overall PhoneGap

experience is internal page linking. There are two ways of creating a HTML5 based application in jQM, single page templates and multi page templates. The classic way of designing a website, which also applies for most web based apps nowadays, is a multi page template approach. That means that the application consists of a certain number of HTML-pages such as index.html, about.html and so one with (if needed) each linking at each other. The second way of designing the application is by using a single page template. That means that only a single HTML-page with several div data-role=page elements in it exists which are able to simulate an arbitrary amount of different views.

The main reason not to implement the application in a single template way is that the more complex the project gets, the more complicated and unclear in terms of internal structure the single HTML-file becomes. By doing that it was also abstained from using the basic jQM way of simulation page transitions such as the native looking swipe effect on view change and it was created a workaround based on mobile.changePage() which gave back the possibility to get the same effects while using multiple HTML-pages.

As mentioned before the whole application was created within the iOS native development environment Xcode. In general this works well, apart from limited JavaScript debugging possibilities. The best practice in checking JavaScript code in websites and web-based applications is by debugging it directly in a desktop browser. If the lead platform is iOS, than testing in Safari using the standard developer tools works just fine. Other than that almost every major browser nowadays provides developer tools and plugins such as Firebug for Firefox give the developers even more possibilities for debugging and analyzing JavaScript code.

The basic process of creating a working and well functioning iOS PhoneGap application turned out to be pretty straightforward and easy to achieve for any experienced web programmer. Phone-Gaps containers for iOS work very well with the Safari Mobile browser engine leading to a good performing though not quite native user experience.

After creating the PhoneGap application using XMLHttpRequest to handle the HTTP connection and exchange of XML data between server and client the tested and working code was transferred from Xcode to Eclipse with Android SDK. Deploying the application on an Android Samsung Galaxy S2 worked flawlessly and the application showed on the home screen of the phone as expected. Due to the lack of hardware CSS acceleration of Android the application did not perform as well as in iOS at all. Page transitions were stuttering and the overall impression in terms of performance and visual quality was not nearly as decent as in iOS. The main problem was that it was not possible to get XMLHttpRequest to work. The only solution to get the application work in Android after was by replacing all XMLHttpRequest related code with the jQuery function jQuery.ajax() and grant the application the right to communicate with our SAP web services via Manifest.xml in the Android project folder to avoid cross origin conflicts.

Managing to get the PhoneGap application to work both in iOS and Android the process was repeated by transferring the code into the native Windows Mobile development environment. The process was as simple as before in Android and the application ran on the Samsung Omnia 7 test device shortly. Problems began as soon as the application was started. There was a noticeable delay before the initial login screen appeared and none of the implemented functionality such as user login or transition to another page worked. The previously described workaround to apply page transitions to multiple page templates using jQuery Mobiles mobile.changePage() was not recognized by Windows Phones IE Mobile at all, which seems to only accept page transitions via Windows.location.href(). Apart from that all of the efforts to successfully establish either a jQuery.ajax() or XMLHttpRequest connection failed. Due to the lack of documentation and the numerous possibilities to cause this problem there was no way to find the cause and fix it leading to the conclusion that, in this case, the PhoneGap approach in combination with Windows Mobile failed.

The following figure shows yet again the summarized findings comparing iOS, Android and Windows Mobile as the target platform for the deployed application. This overview does not aim to be complete and illustrates, by way of information, the most important aspects of the analysis.

iOS	Android	Windows Mobile
+ best performance + excellent API Support + fixed resolution makes development easier + almost native look&feel + very good JavaScript support + very good online documentation	- lack of CSS hardware acceleration - far away from native look and feel poor performance even on high end phones - some JavaScript functions (e. g. XMLHttpRequest) are not working + very good online documentation	- very poor performance - lack of essential JavaScript and CSS support - PhoneGap APIs work properly, but are useless without proper JavaScript implementation - poor online documentation - very small user-/developer base

Fig. 3. Findings of the analysis

5 CONCLUSION

The intention of our research was to prove the often promised universal applicability of native-centric cross-platform frameworks using HTML5-techniques to develop mobile applications running on various native platforms. In this precise case it was analyzed if the one-off source code created by using the example of PhoneGaps is enough to deploy an application for iOS, Android or Windows Mobile. In this context it becomes clear that we have to differentiate the target operating systems.

In general it can be summarized that the realization of the business case could be implemented without problems on the iOS as the leading platform. The excellent API support allowed a rapid development and implementation.

The transition to Android was problematic and the result was an application with poor performance and a user interface far away from the native look and feel.

A proper implementation on Windows Mobile was practically impossible to achieve. It becomes clear that PhoneGap does not support all JavaScript functions really for cross platforms and in case of Windows there is a high potential of failure. The poor documentation increases the problem and does not allow a simple, uncomplicated and clearly structured implementation.

Based on these findings, the use of a native-centric cross-platform framework can only be recommended partially. The question is whether it is possible to create an application mostly with HTML5-techniques. Even Charland and Leroux [2011] assume that web browser are able to support more and more API functions. In the meantime - when some specific APIs are necessary - a small native container should be developed using the native development environment.

REFERENCES

ATKINS, A. S., PENGIRAN-HAJIALI, A., AND SHAH, H. 2006. Extending e-business applications using mobile technology. Proceedings of the 3rd international conference on Mobile technology, applications systems.

BASOLE, R. C. 2005. Mobilizing the enterprise: a conceptual model of transformational value and enterprise readiness. 26th ASEM National Conference Proceedings, 364-371.

CHARLAND, A. AND LEROUX, B. 2011. Mobile application development: Web vs. native. Communications of the ACM 54, 5, 49-53.

CORRAL, L., SILLITTI, A., SUCCI, G., GARIBBO, A., AND RAMELLA, P. 2011. Evolution of mobile software development from platform-specific to web-based multiplatform paradigm. Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reections on programming and software.

GEBAUER, J. AND SHAW, M. J. 2004. Success factors and impacts of mobile business applications: result from a mobile e-procurement study. International Journal of Electronic Commerce 8, 3, 19-41.

MIKKONEN, T., TAIVALSAARI, A., AND TERHO, M. 2009. Lively for qt: A platform for mobile web applications. Proceedings of the 6th International Conference on Mobile Technology, Application Systems.

NAH, F., SIAU, K., AND SHENG, H. 2005. The value of mobile applications: a utility company study. Communications of the ACM 48, 2, 85-90.

SMUTNY, P. 2012. Mobile development tools and cross-platform solutions. Carpathian Control Conference (ICCC), 2012 13th International, 653-656.

YAN, B. AND CHEN, G. 2011. Appjoy: Personalized mobile application discovery. Proceedings of the 9th international conference on Mobile systems, applications, and services.

Branded Financial Apps – Was erwarten Digital Natives?

SEBASTIAN SPRENGER
Friedrich-Alexander-Universität Erlangen-Nürnberg

Menschen, die mit den technologischen Fortschritten des Informationszeitalters aufgewachsen sind, werden heute als sog. Digital Natives bezeichnet. Die ständige Verfügbarkeit des Internets und die damit einhergehende regelmäßige Nutzung mobiler Datenkanäle verändert das Kaufverhalten junger Erwachsener im Vergleich zu älteren Bevölkerungsschichten signifikant. Diesen Trend haben Unternehmen längst erkannt und versuchen, mit sog. Branded Apps auf den Bildschirmen der Smartphones und Tablet-PCs präsent zu sein. Auch die Finanzdienstleistungsbranche verschließt sich dieser Entwicklung nicht und arbeitet daran, Produkte und Dienstleistungen aus dem Finanz- und Versicherungswesen mobil darzustellen. Allerdings fehlt Unternehmen häufig ein Konzept zur Umsetzung auf diesem neuartigen Gebiet. In Kooperation mit einem Finanzdienstleister zeigt die hier vorgestellte Studie, welche Anforderungen die Zielgruppe künftiger Kunden an ein Unternehmen hat, wenn es um die Betrachtung und Bewertung einer Branded Financial App geht. Zu den wesentlichen Erkenntnissen dieser Umfrage zählt, dass Benutzerfreundlichkeit und Sicherheit einer Applikation als wichtiger erachtet werden wie bspw. Interaktionsmöglichkeiten und Support. Weiterhin stehen zentrale Kundenprozesse wie die Durchführung von Transaktionen im Vordergrund. Unternehmensinformationen und Entertainment gelten eher als nachrangige Erwartungen an eine Branded Financial App.

Kategorie und Themenbeschreibung: Mobile and Ubiquitous Computing

Zusätzliche Schlüsselwörter: Digital Natives, Branded Financial Apps, Mobiles Internet und Finanzdienstleistungen

1 EINLEITUNG

Mobile Daten und die damit verbundenen technischen Möglichkeiten zum unbegrenzten Abruf von Informationen aus dem Internet prägen die letzten und kommenden Jahre dieses Informationszeitalters. Aktuelle Zahlen unterstreichen die Wichtigkeit dieses technologischen Fortschritts. Im vergangen Jahr 2012 waren rund 58% der deutschen Bevölkerung über ein mobiles Endgerät (Smartphone, Tablet-PC oder Netbook) mit dem Internet verbunden [Accenture 2012]. Auch wenn dies bereits auf die beachtenswerte Relevanz mobiler Datenkonzepte hinweist, zeigt ein Blick auf die Altersstruktur der Benutzer, dass mobiles Internet unter jungen Erwachsenen faktisch omnipräsent ist. In der Accenture Mobile Watch Studie 2012 gaben 84% der Befragten Deutschen zwischen 14 und 29 Jahren an, einen regelmäßigen Internetzugang über das Smartphone, einen Tablet-PC oder ein Netbook zu nutzen [Accenture 2012]. Diese Altersgruppe wird heute allgemein hin auch als Digital Natives bezeichnet. Unter jenem Begriff wird diejenige Altersgruppe verstanden, die mit digitalen Technologien des Informationszeitalters − wie bspw. Computer, E-Mail, Internet, Handy − aufgewachsen ist [Prensky 2001]. Auch wenn die strikte Grenze des Geburtenjahrgangs von 1980 in der wissenschaftlichen Literatur durchaus als umstritten gilt, geben Palfrey und Gasser damit zumindest einen Orientierungswert über die Bevölkerungsgruppe, welche heute als Digital Natives bezeichnet wird [Palfrey and Gasser 2008]. Bezieht man diese altersmäßige Eingrenzung der Bevölkerung auf die Ergebnisse der bereits erwähnten Accenture Studie, zeigt sich, dass es im Gegensatz zur gesamten Bevölkerung eben gerade diese Digital Natives sind, welche den täglichen Umgang mit dem mobilen Internet leben.

Adresse des Autors: Sebastian Sprenger, FAU Erlangen-Nürnberg, Deutschland, www.wi3.uni.erlangen.de

Diese Entwicklung wurde von Unternehmen längst erkannt. Auch deswegen spielt das Internet für Firmen sämtlicher Branchen und Wertschöpfungsstufen heute eine zunehmende Rolle. Unternehmen haben damit begonnen, auf das veränderte Kaufverhalten heutiger und künftig potentieller Kunden zu reagieren und sind zunehmend mit verschiedensten Apps in den Applikationsmarktplätzen der wichtigsten Smartphone Hersteller, sog. App Stores, vertreten. Bei solchen mobilen Applikationen handelt es sich um funktionale Programme, versehen mit Logo, Marke oder Namen eines Unternehmens. Im Gegensatz dazu stehen reine Werbeeinblendungen auf einem mobilen Endgerät. Viele Großunternehmen und Konzerne betreiben dieses Konzept zur Etablierung der Marke im mobilen Umfeld bereits sehr erfolgversprechend. Insbesondere die Branche der Automobilindustrie gilt hier als Vorreiter im deutschen Segment.

Aber auch die Finanzdienstleistungsindustrie beschäftigt sich zunehmend mit dem Thema Mobilität und mobilen Zugriffskonzepten. Im Rahmen eines mehrstufigen Forschungsprojektes zwischen der Friedrich-Alexander-Universität Erlangen-Nürnberg und einem Praxisunternehmen aus der Finanzdienstleistungsindustrie soll diese Studie die Bevölkerungsgruppe der Digital Natives besser verstehen lernen und gleichwohl die Erwartungen jener potentieller Neukunden an eine sog. Financial Branded App stellen. Weiterhin dienen die Ergebnisse dieser Studie als Basis für die nächsten Schritte im Bereich des Prototyping.

2 THEORETISCHER HINTERGRUND

Branded Apps als Teil mobiler Marketinginstrumente sind in diesem Kapitel zunächst kurz im breiter gefassten Kontext von E-Business, E-Commerce, M-Business und M-Commerce zu betrachten. Die Abgrenzung und Einordnung der einzelnen Begrifflichkeiten ist nötig, um ein einheitliches Verständnis zwischen Leser und Verfasser der Studienergebnisse zu schaffen. Eine zusammenfassende Darstellung über die Zusammenhänge der einzelnen Begriffe ist in Abb. 1 dargestellt.

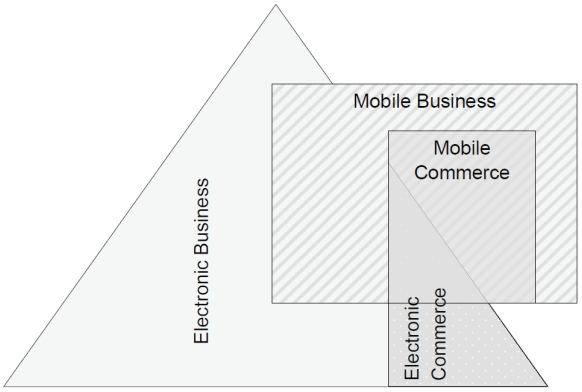

Abb. 1: Abgrenzung E-Business, E-Commerce, M-Business, M-Commerce (Tiwari and Buse 2007)

E-Business betrachtet alle computergestützten Geschäftsprozesse eines Unternehmens, unabhängig ob diese lieferantenorientiert, innerbetrieblich oder kundenorientiert ablaufen. E-Commerce hingegen bezieht lediglich die Transaktionsprozesse der Anbahnung, Vereinbarung und Durchführung auf Kundenseite ein [Mesenbourg 2001]. Der Begriff des M-Business bzw. M-Commerce rückt den zusätzlichen Aspekt der Mobilität in den Vordergrund. Beim M-Commerce werden all diejenigen Prozesse betrachtet, welche auf mobile Weise die Anbahnung, Vereinbarung und Durchführung eines kundenorientierten Ablaufs unterstützen [Tiwari and Buse 2007]. Abb. 1 veranschaulicht diesen Zusammenhang der einzelnen Begrifflichkeiten.

Mobiles Marketing zeichnet sich, angelehnt an die Transaktionsprozesse, in erster Linie durch Audioausgaben und visuelle Darstellungsformen auf mobilen Endgeräten aus [Shankar et al. 2010]. Aufgrund des technologischen Fortschritts sind den Möglichkeiten in diesem Bereich hier heute kaum noch Grenzen gesetzt. Die Ansätze des mobilen Marketings reichen heute von mobilen Webseiten über Gutscheinmodelle bis hin zu mobilem Social Network Management [Shankar et al. 2010].

Eine mit dem steigenden Aufkommen von Smartphones neuartige Möglichkeit des Mobilen Marketings sind sogenannte Branded Apps. Bellman und Potter definieren diese so: „[…] software downloadable to a mobile device which prominently displays a brand identity, often via the name of the App and the Appearance of a brand logo or icon" [Bellman et al. 2011, S.191]. Erica Swallow kategorisiert den Oberbegriff der Branded Applications noch einmal in "Utility Apps", "Product Enhancement Apps" und "Entertainment Apps" [o.V. 2011]. „Utility Apps" betrachten hierbei Applikationen, die das Unternehmen oder deren Produktpalette im Vordergrund sehen. Diese Applikationen werden also vordergründig als Präsentationsfläche genutzt. „Product Enhancement Apps" sehen das Kernprodukt des jeweiligen Unternehmens im Zentrum der Applikation. Es geht darum, das vom Kunden bereits erworbene Produkt um Funktionen und Einsatzmöglichkeiten zu erweitern. Bekannte und erfolgreiche Beispiele lassen sich u.a. in der Automobilindustrie finden (bspw. Keyless Car, Standheizung via App u.v.m.). „Entertainment Apps" zielen auf Unterhaltung, Spiel und Spaß ab. Zahlreiche Unternehmen versuchen durch geschickte Produktplatzierungen innerhalb eines Spiels o.ä. die Marke bzw. den Hersteller zu platzieren und dadurch positive Assoziationen zwischen (potentiellem) Kunde und Unternehmen herzustellen.

Welche Rolle diese verschiedenen Möglichkeiten von Branded Applications im Bereich des mobilen Marketings spielen, betrachtet diese Umfrage. Dabei werden beispielhaft ein Unternehmen der Finanzdienstleistungsbranche und sein Produktsortiment herangezogen und die funktionalen bzw. nicht-funktionalen Anforderungen an eine sog. Branded Financial App erhoben.

3 STUDIENDESIGN UND ERGEBNISSE

Die Daten für die quantitativ ausgelegte Studie wurden mit Hilfe einer Online Umfrage erhoben. Hierfür sind im Zeitraum zwischen Dezember 2012 und Januar 2013 insgesamt 373 Personen, die der Zielgruppe von Digital Natives entsprechen, befragt worden. Die Teilnehmer der Umfrage waren zu 82% zwischen 18-25 Jahren. Betrachtet man die Definition von Digital Natives nach Palfrey und Gasser (s.o.), so entsprechen fast 97% der Befragten jener Gruppe von Personen, die mit den digitalen Technologien aufgewachsen sind.

Der standardisierte Fragebogen ist im Wesentlichen in drei Bereiche aufgeteilt worden. Während in einem ersten Schritt allgemeine Informationen und soziodemografische Daten über den Probanden abgefragt worden sind, enthielten die anderen beiden Teile inhaltliche Fragebogenkonstrukte über die Bekanntheit, Nutzung und Erwartung an Branded Apps im Allgemeinen sowie die Erwartungshaltung und derzeitige Nutzung von Branded Apps speziell im Bereich von Finanzdienstleistungen. Aufgrund des beschränkten Seitenvolumens für diesen Artikel werden im weiteren Verlauf dieses Kapitels nur die Kernergebnisse der Umfrage vorgestellt, was die Betrachtung funktionaler und nicht-funktionaler Anforderungen an eine Branded Financial App umfasst. Die vollständigen Inhalte dieser Studie können jederzeit über den Autor erfragt werden.

Um Erkenntnissgewinne aus den Umfragewerten ableiten zu können, musste im ersten Schritt diejenige Kohorte der Grundgesamtheit identifiziert werden, welche auch tatsächlich Erfahrung im Umgang mit mobilen Endgeräten, App Stores und mobilen Applikationen hat (siehe Abb. 2). Da diese Frage von 80,43% der insgesamt 373 Befragten bejaht wurde, konnten im weiteren Verlauf

die Einschätzungen von insgesamt 300 Personen in die Erwartungenen an eine Branded Financial App einfließen. Deshalb stützen sich die Kernergebnisse über die Anforderung an eine Branded Financial App (siehe auch Abb. 3 und Abb. 4) auf jene Teilnehmergruppe mit N=300.

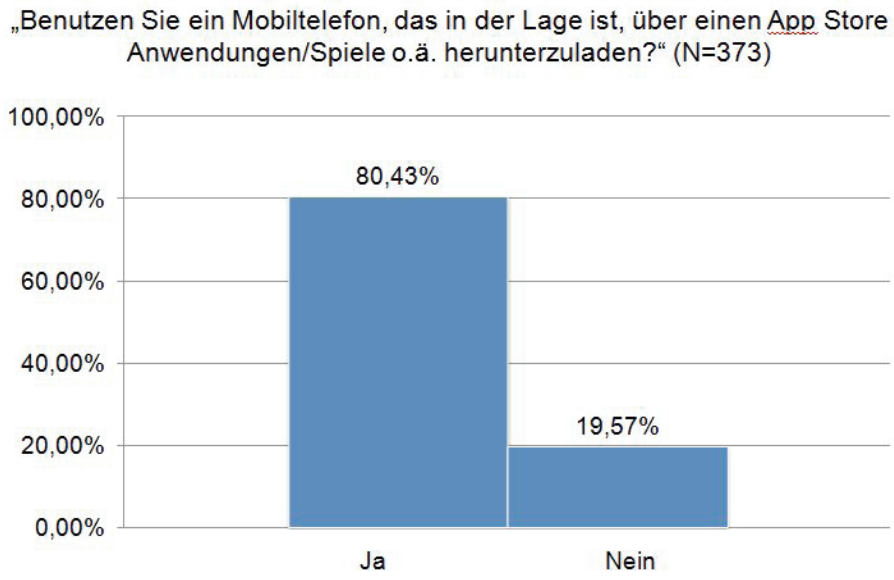

Abb. 2: Abfrage der Nutzung eines mobilen Endgeräts

Funktionale Anforderungen an eine Branded Financial App

Der funktionelle Umfang einer mobilen Applikation im Bereich von Finanzdienstleistern ist insbesondere unter dem Aspekt der drei Kategorien “Utility Apps”, “Product Enhancement Apps” und “Entertainment Apps” betrachtet worden.

Um den Umfrageteilnehmern ein konkreteres Verständnis dieser Funktionen zu geben, sind die Kategorien im auszufüllenden Fragebogenkonstrukt nochmals unterschieden und mit anschaulichen Beispielvorschlägen versehen worden. So konnten die Studienteilnehmer besser verstehen, was unter den jeweiligen Überbegriffen zu verstehen war.

Die Kategorie „Product Enhancement“ ist hier nochmals in Datenabruf und Transaktionsdurchführung unterschieden worden. Beide Konstrukte beziehen sich hierbei auf die konkrete Durchführungsphase im Sinne von Anbahnung, Vereinbarung und Durchführung einer Transaktion. Betrachtet man die beiden Bewertungseinheiten „Sehr interessant“ und „Eher interessant“ als Positivwerte der 5-stufigen Likert-Skala, lassen die Ergebnisse erkennen, dass entgegen der anderen Kategorien „Utility Apps“ und „Entertainment Apps“ im Finanzdienstleistungsbereich vor allem die Transaktion und Geschäftsbeziehung zwischen Kunde und Dienstleister per se im Vordergrund steht und durch mobile Konzepte unterstützt werden soll. Allgemeine Produkt- und Unternehmensinformationen sowie das Thema „Gamification“ spielen hingegen unter den Befragten nur eine untergeordnete Rolle (siehe Abb. 3).

Betrachtet man also die funktionalen Anforderungen einer Branded Financial App, wird unter der befragten Gruppe vor allem die mobile Unterstützung bestehender Transaktionsverhältnisse erwartet. Ein Blick auf derzeit verfügbare Apps in den wichtigsten App Stores (Google Play Store

und Apple App Store) zeigt, dass Finanzdienstleister, wie Banken und Versicherungen, auf diesem Feld bereits tätig sind und der Abruf von Kontoständen sowie Überweisungsdurchführungen, Brokerage etc. bei vielen Unternehmen bereits möglich ist.

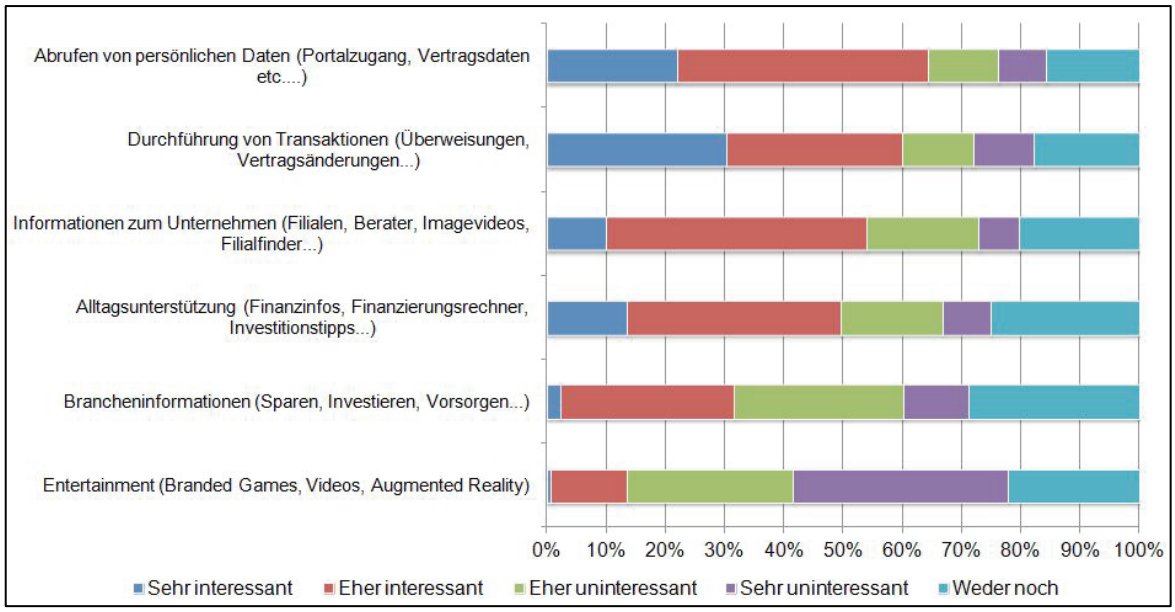

Abb. 3: Funktionale Anforderungen an eine Branded Financial App

Nicht-funktionale Anforderungen an eine Branded Financial App

Die Abfrage nicht-funktionaler Anforderungen ist ebenfalls mit Hilfe einer 5-stufigen Likertskala durchgeführt worden. Allerdings ist bei der Bewertung statt dem Interesse des Probanden (wie bei funktionalen Anforderungen) hier die Wichtigkeit einzelner Faktoren in den Vordergrund gerückt. Auch sind für die Auswertung der Ergebnisse und eine Rangfolgeerstellung die beiden Positivwerte „Sehr wichtig“ und „Wichtig“ zusammengefasst worden.

Wie in Abb. 3 zu sehen, entspricht die Erwartung an nicht-funktionalen Aspekten in etwa der, welche grundsätzlich bei der Entwicklung und Bereitstellung von mobilen Applikationen zu berücksichtigen ist. Interessant hierbei ist allerdings, dass die Bedienbarkeit einer Applikation sogar als wichtiger (~96%) bewertet wird als Vertraulichkeit (~93%), Sicherheit (~91%) und Schutz der Privatsphäre (~92%). Gerade unter der Berücksichtigung, dass es sich um die Transaktionsunterstützung im Bereich von Finanzdienstleistungen halten soll, ist diese Erkenntnis bemerkenswert. Interaktive Elemente wie die Kopplung mit sozialen Netzwerken, Support durch E-Mail und Hotlines sowie der innovative Fortschrittsgrad von Branded Financial Apps spielen nur eine untergeordnete Rolle. Mit Blick auf die Wichtigkeit ordnungsgemäßer Beratung beim Kauf/Verkauf von Finanzprodukten ist die nahezu irrelevante Bewertung von Supportmöglichkeiten ebenfalls ein nicht zu erwartender Punkt.

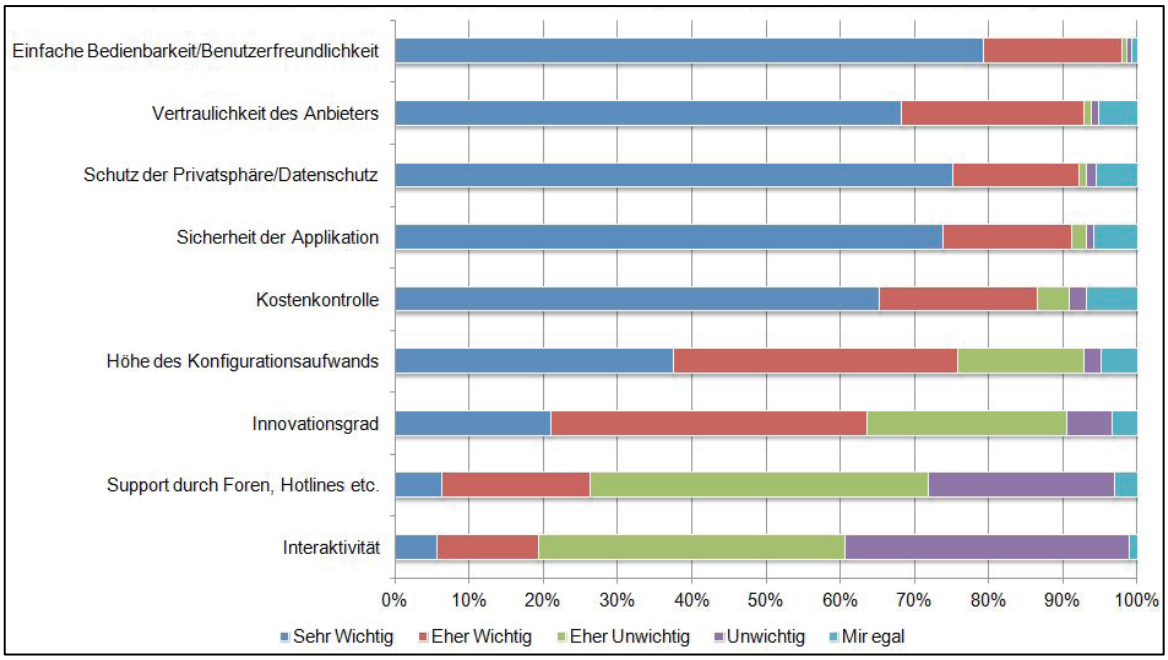

Abb. 4: Nicht-Funktionale Anforderungen an eine Branded Financial App

4 ZUSAMMENFASSUNG UND AUSBLICK

Menschen, die mit den technologischen Fortschritten des Informationszeitalters aufgewachsen sind, werden heute als sog. Digital Natives bezeichnet. Die ständige Verfügbarkeit des Internets und die damit einhergehende regelmäßige Nutzung mobiler Datenkanäle verändert das Kaufverhalten junger Erwachsener im Vergleich zu älteren Bevölkerungsschichten signifikant. Gleichzeitig bieten die erweiterten Möglichkeiten aber für Unternehmen auch neuartige Chancen zur Kontaktaufnahme mit potenziellen Neukunden. Unternehmen erkennen diesen Trend zunehmend und versuchen mit sog. Branded Apps Kundenkontakt über bspw. das Smartphone oder den Tablet-PC aufzubauen. Allerdings fehlen vielen Unternehmen konkrete Ansatzpunkte zur Planung, Entwicklung und Umsetzung von kundenorientierten Branded Apps. Um dieses Defizit schrittweise und praxisorientiert zu lösen, wurde in Zusammenarbeit mit einem Unternehmen aus der Finanzdienstleistungsbranche die hier vorgestellte Studie durchgeführt. Die Finanzdienstleistungsbranche ist auch aufgrund der stark IT-unterstützten Kundenprozesse ein äußerst interessantes Feld zur Untersuchung von Branded Apps und eignet sich deshalb als Untersuchungsumgebung. Die Umfrageergebnisse schaffen den Grundstein zur Entwicklung einer sog. Branded Financial App. Neben globalen Aussagen zum Markt für Branded Apps wurden die funktionalen und nicht-funktionalen Anforderungen für eine kundengetriebene mobile Applikation abgeleitet und interpretiert.

Dabei ist hervorzuheben, dass unter Betrachtung funktionaler Aspekte vor allem das Thema „Product Enhancement“ relevanter erscheint als die anderen beiden Bereiche „Utility Apps“ und „Entertainment“. Eine Branded Financial App sollte also, auf Basis der Aussagen der befragten Gruppe, eher die Durchführung von Transaktionen und deren Beobachtung unterstützen als das Produktsortiment oder Unternehmensinformationen in den Vordergrund zu rücken.

Auf dem Gebiet nicht-funktionaler Anforderungen sind es erwartungsgemäß Faktoren der Benutzerfreundlichkeit, Vertraulichkeit, Sicherheit und Schutz der Privatsphäre, die die Befragten von Applikationen aus dem Umfeld von Finanzdienstleistern erwarten. Diese Faktoren gehen konform mit den grundsätzlichen Ansprüchen an eine erfolgreiche mobile Applikation.

Diese Ergebnisse sind als erster Schritt des bereits erwähnten Forschungsprojekts zwischen der Friedrich-Alexander-Universität und dem Praxispartner zu verstehen. In einem nächsten Schritt erfolgt die Analyse der Business-Seite im Unternehmen, indem die einzelnen Kundenprozesse „Information“, „Beratung“, „Vertragsabschluss“, „Transaktion“, „Service“ und „Auflösung“ auf ihre detaillierten Teilprozesse und deren derzeitige IT-Unterstützung untersucht werden. Mit Hilfe dieser Erkenntnisse kann daraufhin jener kundenorientierte Teilprozess durch eine mobilen Applikation unterstützt werden, bei welchem die geforderten funktionalen und nicht-funktionalen Anforderungen bestmöglich erfüllt und gleichzeitig ressourceneffizient vom Unternehmen bereitgestellt werden können. Die Prototypisierung der Branded Financial App ist für Sommer/Herbst 2013 vorgesehen.

REFERENZEN

ACCENTURE 2012. MOBILE WEB WATCH 2012 SPECIAL EDITION: GERMANY, AUSTRIA, SWITZERLAND

BELLMAN, S., POTTER, R., TRELEAVEN-HASSARD, S., ROBINSON, J. AND VARAN, D. 2011. *GETTING THE BALANCE RIGHT: COMMERCIAL LOADING IN ONLINE VIDEO PROGRAMS.* JOURNAL OF ADVERTISING, 41 (2). S. 5-24.

O.V. 2011. "13 BRANDED IPHONE APPS THAT ENHANCE THEIR COMPANY'S PRODUCTS", HTTP://WWW.OPENFORUM.COM/ARTICLES/13-BRANDED-IPHONE-APPS-THAT-ENHANCE-THEIR-COMPANYS-PRODUCTS/ (ABRUFDATUM 5.04.2013)

MESENBOURG, T. L. 2001. MEASURING ELECTRONIC BUSINESS. S. 1-20.

PALFREY, J AND GASSER, U. 2008. BORN DIGITAL: UNDERSTANDING THE FIRST GENERATION OF DIGITAL NATIVES. BASIC BOOKS PUBLISHING.

PRENSKY, M. 2001. DIGITAL NATIVES, DIGITAL IMMIGRANTS. NCB UNIVERSITY PRESS 9 (5). S. 1-15

SHANKAR, V., VENKATESH, A., HOFACKER, C., AND NAIK, P. 2010. MOBILE MARKETING IN THE RETAILING ENVIRONMENT: CURRENT INSIGHTS AND FUTURE RESEARCH AVANUES. THE JOURNAL OF INTERACTIVE MARKETING. S. 1-31.

TIWARI, R., AND BUSE, S. 2007. THE MOBILE COMMERCE PROSPECTS: A STRATEGIC ANALYSIS OF OPPORTUNITIES IN THE BANKING SECTOR. HAMBURG UNIVERSITY PRESS.

Analysis of Service Level Agreements using Process Mining techniques

CHRISTIAN MAGER
University of Applied Sciences Wuerzburg-Schweinfurt

Process Mining offers powerful methods to extract knowledge from event logs. These event logs can be recorded by any process aware information system and must not conform to a particular format. The main contribution of this paper is to show how Process Mining can be used to analyze the time perspective of processes determined in so called Service Level Agreements (SLA). In terms of processes determined by SLAs, the processing time is extremely crucial. Frequently, delays in processes lead to SLA violations. Therefore, the Process Mining based approach in this paper goes beyond SLA monitoring techniques which simply display the correct execution of a process. In contrast to SLA monitoring, already executed processes can be analyzed from a service provider perspective. The paper describes the creation of an integrated process model which supports a fine-grained investigation of the time perspective. Moreover, this model helps to identify time related bottle-necks in these processes. All Process Mining techniques necessary to create the integrated process model are explained based on a prototypical implementation. The main benefit of this approach is to support a continuous improvement of the quality of service as claimed in ITIL.

Categories and Subject Descriptors: Process Mining

Additional Key Words and Phrases: Process Model Discovery, α-Algorithm, Time perspective, ITIL, Service Level Agreement, Service Level Controlling, SLA Monitoring

1 INTRODUCTION

In the last decade real-time communication via internet enabled a close collaboration between multiple organizations. Flexibility and an improved competitiveness are the major benefits of these collaborations. Yet, if two parties work together, duties and responsibilities need to be defined to enable a fruitful collaboration for all parties. In terms of IT supported processes, the ITIL framework (IT Infrastructure Library) offers a powerful set of concepts and approaches to root a service culture in an organization [Van Bon 2008]. One major objective in every effort regarding ITIL is a sustainable management of processes in a service aware environment. Thereby, the quality of service (QoS) is determined in so called Service Level Agreements. To maintain the quality of these services, an ongoing analysis and controlling of an organization's processes is inevitable. However, it is not addressed in ITIL how to analyze the quality of services in Service Level Agreements (SLA). Most efforts that can be found in the literature are dealing with simple SLA monitoring. These approaches focus on monitoring only and are not suitable for detecting issues in the process. Nevertheless, putting more effort on improving the quality of processes is extremely crucial. Studies like the one presented in [Mendling et al. 2007] prove that organizations tend to overestimate the knowledge of their own processes. Particularly large process models tend to be of poor quality in practice, containing formal flaws. Process Mining can be helpful with providing analysis techniques to improve process quality.

Process Mining is a relatively young research discipline offering powerful methods to extract knowledge form event logs of already executed business processes. Thereby, the major advantage of Process Mining is its' platform and format independence. Event logs of every kind of workflow-engine-like information system can be used and transformed into a Process Mining

Author's address: Christian Mager, University of Applied Sciences Wuerzburg-Schweinfurt, Germany.
www.fhws.de.

conform semantic [Van der Aalst et al. 2012]; [Van der Aalst 2011]. In Process Mining the main focus is on extracting the ordering of events in a process which is commonly called the control-flow perspective. Yet, typically event logs contain richer information such as resource, cost or time information. This makes Process Mining an extremely powerful tool, in particular for organizations aiming to improve their processes continuously.

In this paper an integrated process model on the basis of Process Mining techniques is introduced that allows the analysis of the processes determined by SLAs. In particular this approach allows the analysis of executed process and a fine-grained analysis of the time perspective. First of all, Process Model Discovery techniques are used to extract information of the actual process model from event logs. Afterwards, the model is extended by the time perspective to an integrated process model which allows the detection of time-related bottle-necks in the process.

To specify the details of the approach, some requirements must be complied to successfully implement an analysis instrument for SLA aware processes. First of all, for this kind of effort, the process that is analyzed has to be constant. In other words, processes that are dynamically changing over time are not appropriate for using the techniques presented in this paper. Moreover, a process needs to be executed frequently to allow a Process Model Discovery algorithm to recreate the corresponding process model reliably. Every possible firing sequence of activities has to be recorded in the so called event log. A solid number of process executions increase the chance of getting a complete event log. A complete log is the basis of quality mining results [Van der Aalst 2011]. For instance, the processing of a classical IT-service-helpdesk which is repeated frequently and recorded by a ticketing-system is a good example for Process Mining implementation. One further requirement on a process is the causal dependency between a delay in the execution of the process and a violation of the SLA. This means a delayed delivery of the service provider directly prompts an SLA violation. Control-flow information is the least information necessary to allow Process Model Discovery techniques to work with the event log data [Van der Aalst et al. 2004]. However, for the approach presented in this paper, time information needs to be available in the event log as well. In general, due to particular objectives and questions, it has to be decided individually, what kind of information needs to be available in the event log. When using Process Mining techniques, noise data is a common issue that may distort the results fundamentally and lead to wrong interpretation. Multiple approaches exist to prohibit noise data and ensure quality event logs [Weijters et al. 2006]; [De Medeiros and Weijters 2005]. However, this topic is not addressed in this work.

The following section 2 starts with an introduction on Process Mining as well as on Service Management and SLA monitoring. Afterwards, in section 3 an approach is presented that allows the analysis of the time perspective of processes, determined by an SLA. In section 4 the approach is expounded based on an example process. Finally, the outline in section 5 summarizes all results.

2 OVERVIEW

a) Process Mining techniques

Process Mining is a relatively young research discipline which uses concepts of computational intelligence and data mining on the one hand and process modeling and analysis techniques on the other hand [Van der Aalst et al. 2010]; [Van der Aalst et al. 2012]. Further work dealing with this topic was published under the name "Business Process Intelligence" [Grigori et al. 2004]. Discovering, monitoring and improving real processes are the key objectives of Process Mining. This is accomplished by extracting knowledge form so called event logs. Process Mining provides techniques and tools for discovering process, organizational, social, and performance information from event logs. Today's information systems record an enormous amount of data including

information about all activities that have been executed. However, this data might be unstructured, e.g. scattered over many tables or spread over more than one system. In this case, some effort needs to be done to extract and link data and merge it into a unified event log format. Due to this, specific event log formats like the XES-Standard have been proposed in the past which are suitable for Process Mining [Günther 2009]. The ProM framework is a state of the art open source framework provided by the Technical University of Eindhoven. It supports a variety of Process Mining techniques and uses the XES-Standard [TU/e 2012]; [Van Dongen et al. 2005].

Process Model Discovery is one of the major disciplines of Process Mining. In model discovery, algorithms are provided that try to explain the behavior of a process which is recorded in the event log. Thereby, most of the algorithms do not consider any a-priori information. One simple but very famous algorithm for model discovery is the α-Algorithm [Van der Aalst et al. 2004]. Some more advanced mining algorithms were proposed meanwhile which are able to deal with noise, take heuristics into account and so on. The α-Algorithm suits well to the approach described in this paper because it is simple to apply and still very powerful. The α-Algorithm takes a structured event log as input and computes a Petri net model of the process as output. Therefore, the algorithm scans the event log for patterns in order to find causal dependencies between activities. These causal dependencies are used to identify simple sequences, choices, parallelism or loops between activities in the process. Taking these so called ordering relations into account, a Petri net model is computed. Petri net based process models are commonly used in Process Mining due to their high expressiveness, as well as their well-founded mathematical and graphical modeling notation. The well-defined semantic of Petri nets allows a transformation into other modeling notations like BPMN or YAWL. For this reason, Petri nets suit very well into the context of Process Mining [Van der Aalst 1998]; [Van der Aalst 2011].

Process model enhancement is another discipline in Process Mining. Enhancement aims to enrich a process model with additional information of the event log. Ideally the process model is discovered beforehand using a Process Model Discovery algorithm. The output of any enhancement effort is an improved or extended process model. In [Song and Van der Aalst 2008] the following, perspectives are proposed that can be used to enrich a process model by using enhancement:

- Control-flow perspective: focus on ordering of events.
- Organizational perspective: focus on information about resources.
- Time perspective: focus on timing and frequency of events.

b) IT-Service Management and SLA Controlling

IT-Service Management refers to the implementation and management of quality IT services that meet the needs and requirements of a service consumer. The ITIL framework (IT Infrastructure Library) offers a collection of best-practices for the implementation of an IT Service Management [Van Bon 2008]. Duties and responsibilities between service provider and service customer are defined by Service Level Agreements specified in a service contract. Thereby, SLAs define the Quality of Service (QoS) with the help of quality attributes [Ferguson and Huston 1998]; [Molina-Jimenez et al. 2004]. One fundamental aspect of defining Service Level Agreements is that all agreed details in the contract must be measurable. Therefore, it is important to define statistical metrics about the performance of services. This enables an evaluation of the performance of a service and shows if it meets the agreed level of QoS. However, the quality of services in service level agreements and its metrification are very individual for every organization. Availability, response time or delivery time are attributes that are obviously measurable, for instance. These kinds of attributes can be grouped as performance attributes [Trienekens et al. 2004]. Usually, in case of violations of these SLAs, the accountable party has to pay a compensation agreed on in the contract. In order to avoid violations from the perspective of the provider, the continuous measurement and analysis of its' service delivery processes is crucial. Continual Service Improvement (CSI) is one phase in ITIL dealing with reviewing and

maintaining the processes [Van Bon 2008]. CSI recommends a permanent performance measurement and analysis of service delivery processes. Due to this, the measurement of SLA and the comparison to defined requirements are extremely important. To discover violations and measure the quality of service, SLA monitoring techniques can be used. Multiple technical approaches have been developed in the past to enable SLA monitoring [Ameller and Franch 2008]; [Molina-Jimenez et al. 2004]; [Keller and Ludwig 2003]. However, all of these approaches focus on monitoring only and are not suitable for detecting issues in the processes. In [Grigori et al. 2004] an approach is introduced which considers the idea to measure and control business processes in an integrated framework. This idea comes close to the one addressed in this paper, yet it doesn't allow the dynamical adjustment of process models. State-of-the art Process Mining techniques are able to adjust a model considering additional event logs.

3 PROCESS MINING FOR ANALYZING SERVICE LEVEL AGREEMENTS

One of the most important quality attributes processes determined by SLAs is the time perspective. For this reason, proper management and monitoring of time-related process performance is of upmost importance. Van der Aalst introduced an approach called: "Mining additional Perspectives" in [Van der Aalst 2011]. This approach can be used to extend a process model generated by a Process Model Discovery algorithm with additional perspectives. His idea is generic and can be adopted with changes to the problem addressed in this paper.

To create an integrated process model that contains not only the control-flow perspective but also the time perspective, the following information has to be available. Each process instance in the log needs to be identifiable on the basis of a unique case identifier. Additionally, all events that took place within one instance have to be ordered due to their occurrence. These two pieces of information are enough for the α-Algorithm for model discovery to extract the control-flow of a process. The control-flow of a process illustrates the actual process model as it is documented in the process log. Yet, for enriching the model with the time perspective, a timestamp of start and end time of each activity needs to be recorded in the log. These timestamps are used to calculate the waiting time and the service time of all activities in the model. In this context, waiting time corresponds to the time that passes by between the complete event of an activity and the start event of the following activity. During the waiting time, the process is simply waiting for the next execution step and no progress is made. Waiting time is particularly destructive from the point of view of the service provider as time passes by although the contractually agreed processing time is running. The so called service time refers to the time between the start event and the complete event of an activity. Having an event log with at least this data, the way of creating the integrated process model is explained shortly in the following three steps:

(1) Derive event log suitable for Process Model Discovery. In other words, gather and merge event log data and transform it into a format, e.g. the XES-format, which is suitable for Process Mining.
(2) Use the α-Algorithm for model discovery to compute an appropriate process model. Thereby, the algorithm scans the event log for causal dependencies of activities and puts them together to a Petri net process model. The Petri net model describes the control-flow perspective of the process.
(3) Extend the Petri net model by an additional perspective including time information. The dotted chart analysis, for instance, suits well for comparing and analyzing processing time. The dotted chart analysis shows the occurrence of events over time. This time information for each activity is based on its' start and complete event in the event log. By using start and end time of each activity, the waiting and the service time can be derived. Afterwards, this time information is displayed in the integrated model and compared to the target time.

4 PROCESS MINING APPORACH BASED ON AN EXAMPLE

In this section, a three step approach to generate an integrated process model is described based on the event log in table I.

Case ID	Event name	Ressource	State [lifecycle:transition]	Timestamp
100	Register ticket	system	complete	01.06.2011 06:02
100	Analyze ticket	solver4	start	01.06.2011 06:06
100	Analyze ticket	solver4	complete	01.06.2011 07:12
100	Repair complex	solver2	start	01.06.2011 08:01
100	Repair complex	solver2	complete	01.06.2011 10:32
100	Test repair	tester3	start	01.06.2011 11:05
100	Test repair	tester3	complete	01.06.2011 11:32
100	Submit result	system	complete	01.06.2011 11:40
101	Register	system	complete	01.06.2011 06:08
101	Analyze ticket	solver2	start	01.06.2011 06:34
101	Analyze ticket	solver2	complete	01.06.2011 07:22
101	Repair simple	solver6	start	01.06.2011 08:03
101	Repair simple	solver6	complete	01.06.2011 09:41
101	Test repair	tester3	start	01.06.2011 12:01
101	Test repair	tester3	complete	01.06.2011 12:22
101	Submit result	system	complete	01.06.2011 12:39
102	Register	system	start	01.06.2011 06:10
102	Register	system	complete	01.06.2011 06:10
102	Analyze ticket	solver2	start	01.06.2011 06:45
102	Analyze ticket	solver2	complete	01.06.2011 07:19
102	Repair complex	solver3	start	01.06.2011 08:01
102	Repair complex	solver3	complete	01.06.2011 10:01
102	Test repair	tester5	start	01.06.2011 11:35
102	Test repair	tester5	complete	01.06.2011 12:24
102	Submit result	system	complete	01.06.2011 12:26

Table 1. Event log of help desk process

The approach is implemented in this section by using existing Process Mining techniques available in the ProM framework. Yet, multiple plug-ins have to be used to extract all information necessary for generating the integrated view on a process. So far, there is no plug-in in ProM allowing the creation of integrated process models. The following approach can be seen as an idea for a plug-in unifying the generation of integrated views.

First of all, event log data from a process aware information system needs to be extracted and transformed into a format suitable for Process Mining. The event log used for this example is shown in table I. This log contains all necessary information and is suitable for Process Mining. Only an extract of the full event log is used to generate the process model, as the whole log would be too long to fit in this paper. However, the extract gives a good idea on what the event log data looks like. The first column of the table named "Case ID" describes the case perspective. Each qualifier in this column corresponds to one unique process instance of the process model which has been executed before. In the second column "Event name", the ordering of events within one case is given. These two columns include enough information for computing the control-flow perspective applying the α-Algorithm. Column three "Resource“shows the name of the resource that was executing the event. This column could be added as further perspective to the integrated model. Yet, the resource perspective is not considered in this work. Column four and five show

the "State" and the "Timestamp" of the event. The combined information of these two columns is used later to calculate service and waiting time.

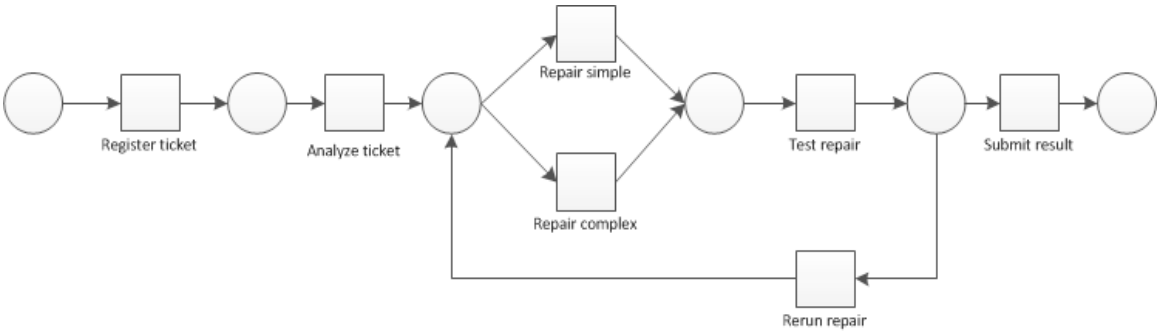

Fig. 1. Petri net model of event log in table I

Assuming this event log is provided as stated in step (1) of the concept, the next task is to compute the process model using the α-Algorithm. The algorithm is provided in ProM by a plug-in. First, the event log as stated in table I is loaded into the mining framework. Therefore, the log needs to be transformed into the XES-format. The α-Algorithm doesn't require any a-priori information provided by the user for computing the Petri net model. The algorithm uses only the case quantifier and the ordering of activities to construct a Petri net model which describes the model of the event log. The output of the algorithm is the Petri net shown in figure 1. Receiving the actual process model alone is already extremely valuable for analyzing processes determined by an SLA. The service provider gets the chance to monitor the actual execution of process instances and not only the process model as it was planned beforehand. In other words, the reengineering approach of Process Mining can be extremely valuable for auditing existing processes. This analysis mechanism is particularly helpful for processes specified by Service Level Agreements, as unforeseen occurrences in these processes may have an undesirable impact for the service provider. Not only the provider is directly affected by contractual penalties, moreover customer dissatisfaction can be prevented. For this reason, information provided by Process Mining can be supportive when aiming to improve services steadily as stated in ITIL.

Figure 2 shows a dotted chart that corresponds to the event log in table I. This chart can be generated using the ProM plug-in called "Dotted Chart Analysis". In this chart the time perspective of an event log is illustrated in a flexible diagram. The dots correspond to the starting point or to the point of completion of an activity in the event log. Thereby, the axis of the diagram can be customized individually to get a static or a relative view.

Fig. 2. Dotted chart analyzer plug-in in ProM6

Finally, in the third step the control-flow and the time perspective are assembled together to an integrated model. This model is illustrated in figure 3. The Petri net based model explains the behavior of the process corresponding to the event log in table I. Thereby, the time perspective is presented in form of a bar chart above places in the Petri net. The waiting time is projected as a red bar and the service time as a blue bar. The bar chart shows waiting and service time for the three process instances 100, 101 and 102.

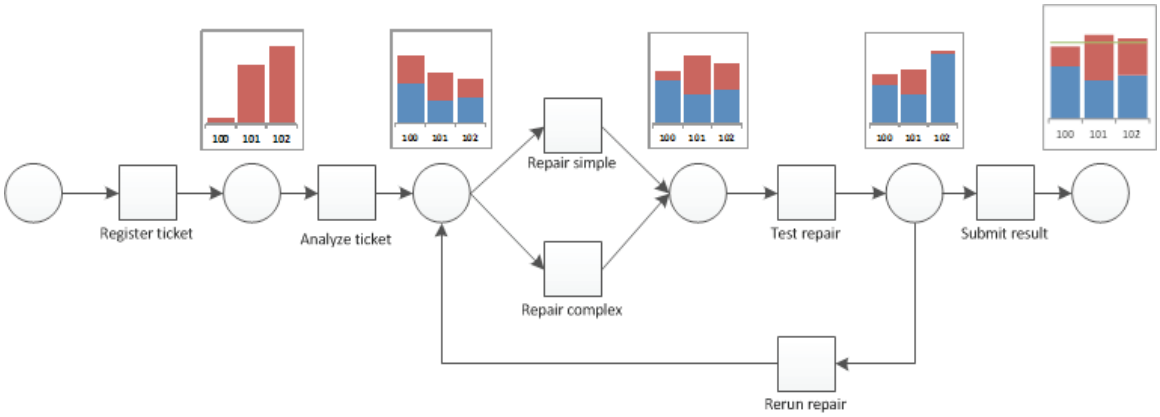

Fig. 3. Integrated process model including control-flow and time perspective

The process in the integrated model in figure 3 operates as follows. First, the activity "Register Ticket" is executed by the system. This activity is executed immediately and doesn't need any service time. However, before the next activity "Analyze Ticket" is processed, waiting time has passed by, as one can see in the bar chart above the place before the activity. The bar chart shows the waiting time for all three instances. The place after the following activity "Analyze Ticket" has two outgoing arcs that correspond to an exclusive or-split. This means either the activity "Repair complex" or the activity "Repair simple" is conducted. The bar chart above this place shows the service time of the previous activity and the waiting time till either one of the repair activities starts. The next place that contains two incoming arcs from both repair activities corresponds to an or-join. Subsequently, the repair is tested by the activity "Test repair". The next

place has again two outgoing arcs. Either the activity "Submit result" is executed and the process terminates or a rerun of the repair activity is performed.

What is particularly interesting about this figure, besides the process model as it is actually executed, is the possibility to analyze the time perspective of the process. The assumption that the processing time is six hours for each ticket is determined in the SLA. This agreement is illustrated in a green line in the bar chart above the final place in the Petri net. At this final place, the delivery of the service to the customer takes place. The bar chart above this final place shows the waiting and service time in total for all instances. One can see that the processing time of instance 100 is below the six hour line and instances 101, as well as 102 are above this line. This monitoring of service processing time is also possible with simple SLA monitoring approaches. However, these approaches don't offer any further details on the violations of the SLA. For improving the processes structure one might be interested in what caused the violations of instance 101 and 102. Therefore, a more fine-grained analysis is necessary. Having a closer look at the integrated model can be helpful here and give some hints on what might have caused the violations. Obviously, when looking at the bar chart above the final node, one can see that not necessarily the service time but the high waiting time of instance 101 and 102 have been problematic. Reviewing the detailed time chart for each processing step, lead to the conclusion that the waiting time before the activities "Analyze Ticket" and "Test repair" has been relatively high in case of instance 101 and 102. The next step would be to use these hints for a deeper analysis within the organization, why the waiting time for these two instances was so high.

5 CONCLUSION

This paper introduced an approach of generating an integrated process model that is suitable for analyzing the control-flow and the time perspective of processes specified by SLAs. This effort can be clearly delimited from classical SLA monitoring techniques, which simply provide an alerting in case of SLA violations. The analysis of SLAs based on Process Mining techniques allows a more precise search for unknown issues in the process composition like deadlocks. The approach presented in this paper enables a fine-grained analysis of the time perspective and the detection of bottle-necks in the process. Delays in the control-flow that may lead to SLA violations can be detected soon after the process execution. Certainly, all information detected by analyzing the integrated model can be extremely valuable for the continuous improvement of a process. The very simple example shows the power of the Process Mining based analysis approach. In case of larger processes with hundreds or even thousands of activities and a much more complex structure these hints derived from an integrated view can be extremely supportive. They might be a valuable first step to improve structure and organization of business processes.

All necessary functionality for this work is available in ProM. Yet, Process Model Discovery techniques and the Enhancement methods are separated over a couple of plug-in. In regard to future work, the approach described in this work could be bound together to one unit in a new plug-in. Furthermore, the next step could be to extend the integrated process model by additional perspectives like organizational or cost perspective. Calculating the approximate waiting and service time or a target time for every activity would enable further time related analysis. A scenario where time information is used to provide operational support in real time to control business processes is imaginable as well.

REFERENCES

AMELLER, D. AND FRANCH, X. 2008. Service level agreement monitor (salmon). Composition-Based Software Systems, ICCBSS 2008, 224-227.

DE MEDEIROS, A. AND WEIJTERS, A. 2005. Genetic process mining. Applications and Theory of Petri Nets 2005, volume 3536 of Lecture Notes in Computer Science.

FERGUSON, P. AND HUSTON, G. 1998. Quality of service: delivering QoS on the Internet and in corporate networks. Wiley New York.

GÜNTHER, C. 2009. Xes standard definition. http://www.xes-standard.org/_media/xes/xes_standard_proposal.pdf (visited on 11/30/2012).

GRIGORI, D., CASATI, F., CASTELLANOS, M., DAYAL, U., SAYAL, M., AND SHAN, M. 2004. Business process intelligence. Computers in Industry 53, 3, 32-343.

KELLER, A. AND LUDWIG, H. 2003. The wsla framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management 11, 1, 57-81.

MENDLING, J. , NEUMANN, G., AND VAN DER AALST, W. 2007. Understanding the occurrence of errors in process models based on metrics. On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, 113-130.

MOLINA-JIMENEZ, C., SHRIVASTAVA, S., CROWCROFT, J., AND GEVROS, P. 2004. On the monitoring of contractual service level agreements. Proceedings of the First IEEE International Workshop on Electronic Contracting, 1-8.

SONG, M. AND VAN DER AALST, W. 2008. Towards comprehensive support for organizational mining. Decision Support Systems 46, 1, 300-317.

TRIENEKENS, J., BOUMAN, J., AND VAN DER ZWAN, M. 2004. Specification of service level agreements: Problems, principles and practices. Software Quality Journal 12, 1, 43-57.

TU/E. 2012. Prom 6. Published by: Process Mining Group, CS department, Eindhoven University of Technology - http://www.promtools.org/prom6/ (visited on 11/30/2012).

VAN BON, J. 2008. Foundations in IT service management: Basierend auf ITIL. Vol. 3. Van Haren Publishing.

VAN DER AALST, W. 1998. The application of petri nets to workflow management. Journal of circuits, systems, and computers 8, 01.

VAN DER AALST, W. 2011. Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer Berlin and Heidelberg.

VAN DER AALST, W., ANDRIANSYAH, A., ALVES DE MEDEIROS, A., ARCIERI, F., BAIER, T., ET AL. 2012. Process mining manifesto. BPM 2011 Workshops Proceedings, 169-194.

VAN DER AALST, W., VAN HEE, K., VAN WERF, J., AND VERDONK, M. 2010. Auditing 2.0: Using process mining to support tomorrow's auditor. Computer 43, 3, 90-93.

VAN DER AALST, W., WEIJTERS, T., AND MARUSTER, L. 2004. Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering 16, 9, 1128-1142.

VAN DONGEN, B., DE MEDEIROS, A., VERBEEK, H. , WEIJTERS, A., AND VAN DER AALST, W. 2005. The prom framework: A new era in process mining tool support. Applications and Theory of Petri Nets 2005, 1105-1116.

WEIJTERS, A., VAN DER AALST, W., AND DE MEDEIROS, A. 2006. Process mining with the heuristics miner algorithm. Technische Universiteit Eindhoven, Tech. Rep. WP 166.

Flow-Design and Event-Based Components: A New Approach in Designing Software Compared to Established Procedures

TOBIAS GEGNER
University of Applied Sciences Wuerzburg-Schweinfurt

Flow-Design is a new approach in developing software architectures. Flow-Design focuses on data processing function units and the communication among them. Event-Based Components represent a programming language independent methodology for implementing Flow-Design and can also be considered as a resulting software architecture. The Event-Based Components software architecture is structured by components that communicate via events. Therefore it is compared to the concept of service-oriented architecture and event-driven architecture which aim to create independent components as well. In conclusion Flow-Design and Event-Based Components provide a platform and programming language independent concept to develop software components. The resulting components are reusable, maintainable and testable. Although it can lead to a more complex coding.

Kategorie und Themenbeschreibung: Software Engineering

Object oriented programming, Software architecture, Software design, Software reusability, Software engineering.

1 INTRODUCTION

Software development has ever since strived for re-usability and maintainability. Manufacturing industry with standards across companies and national borders has reached a high level in reusing and composing parts built by different suppliers. Serving as a role model, manufacturing industry motivated computer scientists explore ways of mapping industry procedures to software development. Therefore in the past years different approaches arisen to develop software components of a high level of reusability. Within this evolution several approaches evolved using events to couple components and arrange communication among them. These event-based architectures are widely discussed and find appliance in distributed software systems as well as within sole software systems or components e.g. [Cugola et al., 2001, Eugster et al., 2003, Carzaniga et al., 1998, Fiadeiro and Lopes, 2006, Kowalewski et al., 2008].

Following the idea of using events to make software components more independent of their environment Ralf Westphal invented Flow-Design and Event-Based Components. By using design patterns and best practices he evolved the way of designing and implementing software. In the following this paper will introduce Event-Based Components to the reader. By examining a running example, the fundamental concepts of Flow-Design will be discussed, as well as differences to object-oriented analysis and design. Further the implementation of Flow-Design with Event-Based Components will be described. Subsequently a comparison of Flow-Design and Event-Based Components to established design processes and software architectures will be given. In the end a conclusion is given as well as a future outlook.

Author's address: Tobias Gegner, University of Applied Sciences Wuerzburg-Schweinfurt, Faculty of Computer Science and Business Information Systems, Sanderheinrichsleitenweg 20, 97074 Würzburg, Germany, www.fhws.de

2 INTRODUCING FLOW-DESIGN AND EVENT-BASED COMPONENTS

The concept of Event-Based Components (EBC) was mainly introduced by Ralf Westphal, who started to post articles in his German and English speaking blogs [Westphal, 2010b, Westphal, 2010d] in the year 2010. He also wrote articles about EBC in professional journals which will also be referenced in this paper. EBC is a new conceptual way to handle data communication between individual software components. It represents the code implementation of Flow-Design (FD), which is a different approach in designing software architectures. As the name Flow-Design already suggest it's about flows. More concrete, FD describes function units and data that flows among them to fulfill business logic. Ralf Westphal evolved FD and the Flow-Orientation on the basis of Flow-Based Programming and Event-Based Programming concepts which are discussed in computer science for quite a few years [Morrison, 2010, Faison, 2006]. His work was also influenced by Business Process Modeling [Freund and Rücker, 2012] and the Event-Driven Architecture [Bruns, Ralf and Dunkel, Jürgen, 2010]. Enriched with own thoughts he brought up a new approach of designing software: Flow-Design.

In this paper the terms component and flow will play a major role. Therefore based on Ralf Westphals usage of the terms flow and component, these two terms will be defined as follows:

Definition 1 (component). *A component is a modular part of a software. It encapsulates a set of related functions or a single functionality, optionally its data. Further it is replaceable within its environment. Interfaces describe required and provided services. Within this paper the terms component and function unit are used synonymously.*

Definition 2 (flow). *A flow represents communication among components of a software. It describes handing over control of data processing between components as well as hereby exchanged data. As an intellectual concept flows are depicted in diagrams.*

2.1 Running Example

To make the difference of the FD approach clearer a sample application is examined. A command line tool has to be developed, that reformats text from a given file. It reads text from a file, rearranges the line breaks to a given line length and writes the resulting text into a new file. To start the application from command line, three parameters have to be provided by the user. First parameter names the file that contains the text. Second parameter defines the maximum line length to which the text is reformatted. Third parameter names the target file into which the result hast to be written. The application must not change nor apply hyphenation in its first version, this is left to future versions. The application has to guarantee, that the maximum line length is not exceeded nor that fewer words than possible are arranged in one line. Punctuations have to be considered as part of the preceding word and therefore must not be changed in any way [Westphal, 2011d].

2.2 Flow-Design

Designing software with the object-oriented analysis and design (OOAD) approach concentrates on building units of data with associated operations. That way the problem is

analyzed for substantives which become objects in the code representation of the solution. Flow-Design turns this the other way round. The problem is analyzed for verbs which become operations (function units). FD concentrates on what should be done and then looks for the data that needs to float between the function units. Seen that way, FD is closer to the Input-Process-Output (IPO) approach than to OOAD. It centres around the processes that need to be done to solve a certain problem, representing the P in IPO. Therefore the data is represented by I and O in IPO. The customer has data (input) and wants it to be transformed (process) into another form or representation (output) [Westphal, 2010b, Westphal, 2011a].

2.2.1 How to Apply Flow-Design

Following the OOAD approach a possible solution to the running example (2.1) might come up with a couple of single objects generated from the substantives of the task: File ; Text ; Line. Then the discovered data objects are filled with methods that implement the business logic. The object-oriented solution would also be well documented with a UML 2.0 class diagram [Booch et al., 2006]. A simplified diagram of the OOAD solution can be seen in figure 1. The outcome of this approach is a static diagram which gives an overview of the data model but has no information about the work flow.

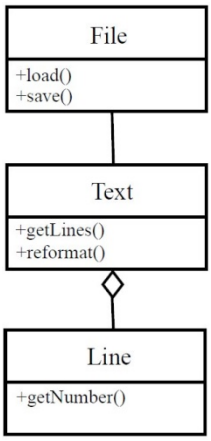

Figure 1. Simplified UML class diagram of the OOAD solution for the Running Example (2.1). Adapted from [Westphal, 2011d]

In contrary, solving Running Example (2.1) with FD would start by looking at the operations that need to be developed. Starting with a rougher first draft of the solution a Flow-Design diagram is denoted that shows the very basic operations represented as function units. In further cycles the solution would be more and more refined. By using FD a possible solution for the problem might identify the following basic function units: *read text from file*; *reformat text*; *write text to file*. Next development cycle the function unit reformat text might be refined to these function units: *fragment to words*; *aggregate to line*; *aggregate to text*. In figure 2 the Flow-Design diagram of the FD solution can be seen.

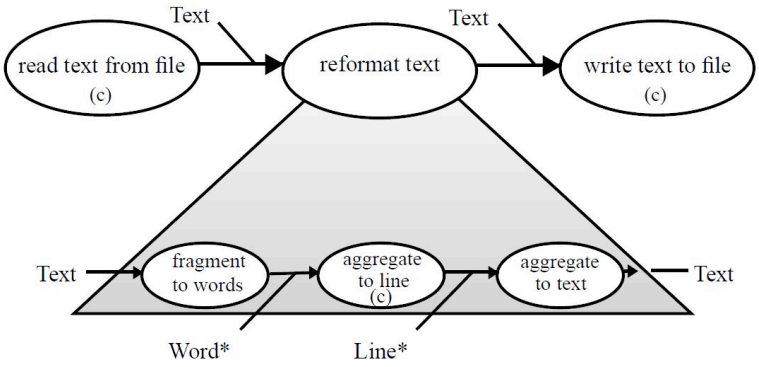

Figure 2: Flow-Design diagram of the FD solution for the Running Example (2.1). Asterisks imply multiplicity one-to-many. Adapted from [Westphal, 2011d]

2.2.2 Flow-Design Diagrams

As Flow-Design introduces a new way of analyzing and designing software and software architectures it also brings a notation with it to depict the design process which was stated by Ralf Westphal as well as FD itself [Westphal, 2010d]. The Running Example (2.1) and its suggested FD solution illustrate how FD follows the process of the solution naturally and depicts the data flow between the function units with FD diagrams. Function units are depicted as circles or ellipses with their name in it. Flows, connecting the function units, are called wires. Therefore connecting function units is called wiring and is denoted with an arrow from the output of a function unit to the input of another function unit. Data only occurs on flows and is denoted with its type and multiplicity. As the FD diagram shows the work flow of the solution one can see how the solution works on different levels of abstraction. It's also possible to variegate the level of detail by switching the layer of abstraction in the diagram to get a deeper view of the solutions details. This way the customer or a new team member can be inducted to the current state of the solution quickly without having to know details about the implementation.

2.2.3 How Flow-Design Structures and Connects Software Components

Going deeper in the level of abstraction, function units have to describe more concrete what they do. Applying this Top-Down approach, FD follows the concept of functional decomposition [Westphal, 2011d]. This way FD enables one to take a look at the solution on different levels of abstraction, giving more or less details about the functionality taking respect of what is needed in the current situation. Taking a closer look at this approach one will see, that in FD there are two different types of function units. FD builds up a hierarchical structure in form of a tree of function units starting at the root, which represents the whole application. Going down the tree there are nodes with further children and leaves at the end of the tree. At the leaves function units implement logic as well as data processing is provided. On the contrary nodes group leaves to form units of a higher level of abstraction. These function units arrange the data flow among the grouped leaves. As seen in the FD solution for the Running Example (2.1), the function unit

reformat text represents a node and decomposes to the three leaves fragment to words, aggregate to line and aggregate to text. This way nodes offer a more cohesive functionality for units on a higher level of abstraction. By distinguishing between nodes and leaves FD follows the concept of Separation of Concerns [Reade, 1989]. The two different kinds of function units in FD handle two different concerns: implementing functionality and aggregating functionality. However the two kinds of function units are depicted with the same symbol in FD diagrams. According to electronic engineering Ralf Westphal named the aggregating function units Boards and the aggregated function units, that implement the logic Parts [Westphal, 2010c]. In figure 3 an example is given of the tree structure with Boards and Parts assembling to the whole application.

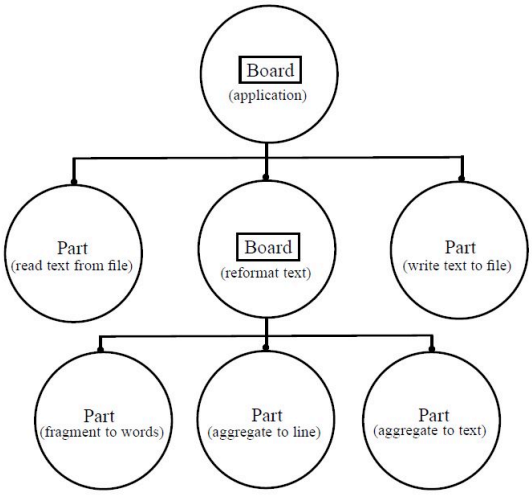

Figure 3: Tree showing the hierarchical structure of FD and two different types of function units. Adapted from [Westphal, 2010c]

By designing Boards and Parts with FD a high granularity can be accomplished and Parts can be build serving just one single task. This way dependency among function units is avoided as well as the Single Responsibility Principle [Martin, 2003] is fulfilled. Parts are independent from functionality of other Parts or Boards due to the fact, that function units only process incoming data and provide the processed data, following the IPO approach. There are no implicit dependencies in form of calls to other FD function units that would break the data flow. While Boards are implicit depended of the Parts they integrate. This implicit dependency is not critical because of the simplicity of Boards. Boards only concern about putting Parts together and should have no business logic implemented.

Beside implicit dependency it is possible that Parts are dependent of existing components or APIs (Application Programming Interface) that provide needed legacy functionality. This is a special form of dependency and should be explicit depicted in FD diagrams [Westphal, 2011d]. To integrate legacy components or APIs in FD and show the explicit dependency the API or components are wrapped into a separate function unit, serving as adapter for FD. This separate function unit takes the needed data for the API or component as input and delivers the result as output data, forming a data flow in FD style. This way legacy components or APIs can be fully integrated into the FD process.

2.2.4 Special Features of Flow-Design and Flow-Design Diagrams

Flow-Design diagrams offer similar ways than UML 2.0 diagrams [Booch et al., 2006] to depict characteristic features in the software design. Later these features will be translated into EBC and therefore executable object-oriented code.

Configurable: As well as in OOAD there is the possibility that function units have to be configured preliminary before they are able to fulfill their function. In Running Example (2.1) that would be read text from file which needs the source file name, aggregate to line which needs the maximum line length and write text to file which needs the target file name. In FD diagrams configurable function units are annotated with a '(C)', as seen in figure 2. The configuration of these function units has to be done before they process any data in the flow. Implementation of the configuration is left to EBC [Westphal, 2010d].

Entry point: To mark where the whole flow and therefore the data processing starts an entry point annotation is available. The function unit which is processed first when the software starts up is annotated with a '(E)'. This is likewise to UML 2.0 activity diagrams [Booch et al., 2006], where the start point of the process is also marked. The implementation details are left to EBC as well [Westphal, 2010d].

Singleton: Since function units are translated into object-oriented code they are multiple instantiated by default. So referring to function units with the same name in a flow means to refer different instances of the same function unit. In example referring to a function unit at the beginning and at the end of a flow would mean pointing to two different instances of the same function unit. This occurs because, as declared previously, function units are independent of their environment and therefore have no knowledge of existing instances of other function units. In case it is needed to reuse the same instance of a function unit at different points of the flow, the Singleton design pattern [Gamma, 1999] can be applied. This has to be denoted in the FD diagram by annotating the function unit with a '(S)' [Westphal, 2010d].

Split and Map: Function units are independent, provide their result to unknown consumers and are connected via flows by Boards. Though it is possible to connect a function unit not only with one but multiple other function units. This is achieved by wiring the output of a producing function unit (P) with the input of every consuming function unit (C). This is called a Split and can be depicted in FD diagrams by just denoting a wire from function unit P to every function unit C. In that case, the same data flows on every wire. Yet the flow is still synchronous and the different branches are not processed in any specific way. Matters of concurrency are left to EBC and its implementation. Letting only Parts of the original data flow pass to connected function units is called a Map. This can be depicted explicitly or implicitly in FD diagrams. Explicit a box is painted into the branch of the flow where the Map is installed. Implicit the data type of the affected branch differs from the original flow. Maps may be applied when a consuming function unit handles only a subset of the flows data at its input [Westphal, 2010d].

Join: While it is possible to create function units with more than one input it may be needed to combine different branches of input flows into one. Especially when all data from the input flows is needed to proceed, a Join should be installed. A Join takes multiple input flows and handles over a tuple of the flow data to its function unit. Flows are considered as separate standard Parts. They are depicted as a vertical line in front of the function units input [Westphal, 2010d].

2.3 Event-Based Components

Event-Based Components are the implementation of Flow-Design architectures and aim at the goal of creating composable function units or components. Designing an application with the FD

approach leads to Boards and Parts, independent function units. These function units work without knowledge of their environment and are therefore independent. In FD diagrams function units are connected via data flows which represent the communication between different Boards and Parts. At that point the function units are composable, at least in terms of FD diagrams. EBC describe how to translate FD in object-oriented code, so that the implemented function units and components are composable and independent as well.

2.3.1 How to Apply Event-Based Components

The basic rules of translating Flow-Design diagrams into runnable object-oriented code are:

- Every function unit, whether Board or Part, becomes a class.
- Every input flow becomes an event-handler method of a class.
- Every output flow becomes an event-sender method of a class.

Following these rules, translating FD diagrams into code is pretty much straight forward. Though the connections between independent Parts have to be created. As well as the assembling of Parts to Boards needs to be solved. To achieve this and preserve the flexibility of FD, best practices and design patterns are applied. For readability of coding, it is helpful to prefix methods for input and output pins accordingly with ‘in ’ and ‘out ’ [Westphal, 2010c].

2.3.2 Issues of Component-Based Software Engineering

If we take a look at Component-Based Software Engineering [Heineman and Councill, 2001] we see that composing different components can be achieved through extracting interfaces from components. The extracted interfaces represent the service contract for the implementing component. Therefore the client components are independent from a concrete implementation of the service interface. Breaking with static dependencies from interface implementations at compile time can be accomplished by making use of the Dependency Injection design pattern [Fowler, 2004]. Thus client components can be bound dynamically at runtime to a concrete service interface implementation. Hereby the client component gets more independent of its environment because it neither has to have knowledge of the implementation of the service interface nor it has to be able to instantiate a concrete implementation of the service interface. Yet a client component is still dependent of the service interface it makes use of. Since interfaces often define more than only one single method and may change over time the Single Responsibility Principle is broken. A client component would not only have to change if the signature of a used method of the interface changes but also if the interface gets decomposed into smaller distinct interfaces. Here the topological dependency of components from their environment shows up. Ralf Westphal named these components Injection-Based Components (IBC) due to the fact, that IBC are injected as whole component. This already points to where EBC are going. EBC resolve topological dependencies and leave components with functional dependencies only, which are uncritical and integral part of client-server relationships [Westphal, 2010e, Westphal, 2010a].

2.3.3 Connecting Function Units with Event-Based Components

Event-Based Components connect different components by using events and event-handlers. A client component that needs a certain service fires an event. A subscribed event-handler of a

service component receives the service request via the fired event and processes the transmitted data. When the service component finished processing the data it fires an event itself and publishes the result that way. That is the basic procedure that maps data flow from FD to EBC. Depending on the used programming language for implementation, different techniques can be used for realizing the events and event-handling. As for example in C# event-handling is an integral part of the programming language and the .Net Framework[11] . Therefore delegates can easily be used to implement events. Whereas the Java Runtime[12] does not support events by default. Here an event-handler has to be implemented manually, for example by implementing the Observer design pattern [Gamma, 1999].

To label the events and create a common code representation for flows, a generic event interface is being created. Both, client and service, use this common event interface to typecast their event data. Depending on the used programming language, solely one interface is needed for the event (e.g. C#) or an interface for event-handler and event-sender is needed to accomplish the same functionality. Except the higher effort in implementation there is no disadvantage for the latter approach and therefore it is not distinct in this paper. For example an event-handler interface in Java would look like:

```
interface IEventHandler<T>{ handle(T event) }
```

That way type safety and flexibility is provided either. Alternatively, if type safety is dispensable or the programming language of choice does not support generics, all events can implement a simple marker interface without a generic type. Through this shared event interface the communication between all components is arranged. Clients send events of a certain type and services provide event-handler methods for that certain event type. With this event based communication between components a client neither gets dependent of a concrete service implementation nor of an aggregated service interface. The client just gets connected to a single service method respectively its interface. Although it looks like the client gets dependent of a certain event or event-handling method, in fact it is the Board which is dependent of the Parts it connects. As mentioned previously, that kind of dependency is nothing to bother about. That way the client component gets topologically independent of the service and the service interface. Standard connectors, likely to pins of electronic parts in electronic engineering, are established through this approach. Due to the similarity to electronic engineering parts, event-sender and event-handler methods are called input pins and output pins. Thus Parts provide certain channels for in and outgoing messages in form of input and output pins. Boards aggregate Parts and other Boards to new units of a higher level of abstraction. Wiring the input and output pins of the included Boards and Parts together to enable the flow means to register event-handler at the corresponding event-sender. Externally the aggregating Board presents input and output pins which are connected to the internally starting and ending point of the flow respectively their pins. Taking a look at Running Example (2.1) the Board reformat text provides an input pin for data of type Text and an output pin of the same data type. Diving into deeper, it is shown that the Boards input pin is connected to the input pin of the Part fragment to words. Connecting input pins of a Board to input pins of an included function unit is accomplished by delegating the incoming event and its parameters to the according event-handler method of the included Board or Part which is in charge of the event-handling. The output pin of fragment to words is wired to aggregate to line which is further wired to aggregate to text. The output pin of the Part aggregate to text is wired to the output pin of the Board by passing through the Boards parameter to the Part. At the beginning and at the end of the internal flow of reformat text is the data type Text. Within the flow itself there are different data types flowing between the internally wired Parts. It is essentially important, to go

[11] Based on .Net Framework 4 published on 21st of February 2011.

[12] Based on Java SE 7 published on 28th of July 2011.

strict with the FD design specifications while implementing EBC. This means, that Boards must not implement any business logic but wiring included Parts and Boards. Otherwise design and implementation drift apart and hidden dependencies may occur. EBC are abstracted through interfaces as well and the previously mentioned Dependency Injection design pattern is used to hand over function units and to aggregate Parts to Boards. Due to the abstraction via interfaces Boards are free of dependencies of concrete implementations of Boards or Parts.

2.3.4 Special Features Implemented with Event-Based Components

As mentioned in 2.2.4, there are several characteristics in software design with FD. The following will depict how to implement these features with common object-oriented techniques. Moreover further concepts and their implementation with EBC are explained. Concrete details how each feature has to be exactly implemented with EBC is depending on the programming language of choice. Therefore this is left up to the implementer.

Entry point: The flow starts, where the entry point in the FD diagram was denoted. Implementing the entry point with EBC is solved with a marker interface which defines the start up method for the flow. Depending on the chosen programming language, the start up method for the application varies. However it is common for start up methods to take a collection of string arguments. Therefore the entry point interface in EBC defines a method named ‘run’ with a parameter typed as string collection as well. For example an entry point interface in Java would look like:

```
interface IEntrypoint{ void run(string[] args)}
```

At application start up the ‘run’ method of the entry point has to be called and the start arguments have to be passed to the method. Acting as the start of the flow, the entry point interface has to be implemented by the Board which aggregates the function units to the whole application. This is at root of the function unit hierarchy tree [Westphal, 2010d].

Configurable: Configurable function units need to be configured at start up at. Yet at least before the take part of the flow. Similar to the entry point, that takes start up arguments, configurable Parts implement a marker interface. This interface defines a method ‘configure’ with a parameter typed as string collection like the entry point interface does. Thereby the interface implementing Part receives the needed parameters to configure itself. In Running Example (2.1) it is required that three parameters are provided at start of the application to define the reformatting details of the text. By implementing the FD diagram, seen in figure 2, with EBC the entry point would get the three parameters at start up and assign them to the according configurable Parts [Westphal, 2010d].

Singleton: If a function unit is only to be instantiated once, the Singleton design pattern [Gamma, 1999] has to be applied. Implementing a singleton of a FD diagram has to be accomplished with instruments of the chosen programming language. Ensuring that only one instance can be created during the program run is mostly achieved through a private class constructor. Only the class itself is able to create a new instance. Though a public static method has to be provided that creates one single instance of the class and provides it to the outside. That way it is guaranteed that, within the same or different flows, every time a singleton function unit is wired it is always the same instance of the function unit [Westphal, 2010d].

Split and Map: Connecting more than one input pin to one output pin is called a split. Data doesn't flow parallel on all branches, but sequential in undefined order. This can be achieved by wiring multiple function units to the output pin. Implementing with EBC that means assigning multiple event-handler to the event-sender. If not all of the data has to flow on all branches of the split flow, a map, that filters data, is needed. To implement a map with EBC a separate class has to be constructed serving as a standard Parts. It takes in the original data and outputs particular Parts of that data filtered by certain criteria. When needed a map is put between two function units while wiring them. So the data flowing from one function unit to the other gets filtered by the map. Summing up, a map helps to connect two function units where the receiving function unit is only able to process a subset of the output data. Therefore a map can also be seen as an adapter [Westphal, 2010d].

Join: A join depicted in a FD diagram aggregates two output pins to one input pin. To implement a join with EBC a separate standard Part, similar to a map, needs to be created. This extra class provides two input pins and one output pin. The provided output pin can either have two parameters or one tuple sticking the two parameters together. By default a join generates an output value whenever a new input is received. Nevertheless for the first output the join class needs all inputs. It is also possible to create joins of different behaviors. Therefore a join can reset all inputs after every output or only produce an output if input on a particular pin is received. Coding details are left to the implementation with the chosen programming language. Installing a join is similar to installing a map. When needed the join is put between the participating function units. The output pins to join are wired with the input pins of the join Part. Then the output pin of the join is wired to the input pin of the receiving function unit [Westphal, 2010d].

Concurrency: Implementing concurrency with EBC can be achieved by using splits and joins. With splits and joins flows can be segmented into multiple branches and brought together as well. As mentioned previously a split is not parallel by default. Therefore available concurrency techniques of the chosen programming language, like threading, threadpools and semaphores, have to be used. This way split flows can be processed in separate threads. With joins the parallelized flows can be back to one thread. In Running Example (2.1) the reformatted text could not only be written into a file but also be printed on the command line as well as directly sent to a printer. Or certain data of the flow could be logged with a logging tool. These further steps could be processed parallel to the original flow and therefore not disturb a user's work flow. As with all parallel processing, concurrency in EBC demands attention during design and implementation to avoid race conditions and dead locks [Westphal, 2010d].

Explicit dependencies: As explicit dependencies of legacy components or APIs can appear in FD, this has also to be handled during implementation with EBC. To make dependencies explicit a generic marker interface is used. This interface defines one method to inject the class of which the implementing function unit is dependent of. The method named 'inject' has one parameter: the required class. Through this interface the dependency is visible in the coding. The dependency is handled at one certain point in the coding. This makes the implementation more maintainable and the coding more readable. By using a method to inject the required class it is independent of object creation. Therefore dependency injection is possible at any time during start-up of the application [Westphal, 2010d].

Exception handling: As most object-oriented programming languages share the principle of exceptions this is also an issue in EBC. Exceptions which can or should not be handled within a function unit are published. Often it is desirable to have a single part in the software where exceptions are handled. Therefore exceptions in function units are published via an extra output pin. Depending on the programming language of choice this output pin can be parameterized with an universal exception type or left untyped.

2.3.5 Tooling

EBC are growing in popularity. Resulting in a community that discusses Flow-Design and Event-Based Components on internet forums. Basic Concepts are discussed as well as possibilities to use tools for EBC development. Meanwhile miscellaneous tools for EBC have been developed to facilitate designing applications with FD and developing with EBC. These tools intend to make routine activities easier for developers or even release them. A graphical presentation of FD should be enabled. And the resulting code from EBC should become unified and therefore gain quality and readability [Westphal, 2011b].

To counter the probably most confusing part of EBC coding, the wiring of components, Ralf Westphal and Stefan Lieser introduced a Domain Specific Language (DSL) for EBC. With the help of this XML based DSL, which is therefore named ebc.xml, wiring Boards and Parts should become more easy. Through ebc.xml aggregating Boards and Parts to lager, more abstract function units is separated from coding. Though it is also separated from business logic which is found within Parts. By using the XML-File ebc.xml no wiring is left to the code but all defined in well-structured and defined XML files. Following the idea of Separation of Concerns [Fowler, 2003] ebc.xml separates implementations details and business logic from structuring components. Being independent of a certain platform or programming language is another advantage of ebc.xml. Finally the code according to the target programming language is generated with an ebc.xml -compiler. The resulting code aggregates Parts to Boards according to the XML file [Westphal, 2011c].

The basis for a common visualizing and wiring of Boards and Parts was built through ebc.xml. Therefore besides the ebc.xml -compiler an ebc.xml -visualizer exists too. This visualizer is able to generate a graphical representation of the wiring defined with ebc.xml files. Hence besides unifying EBC code a unified graphical presentation of EBC is provided with a tool. A better understanding of EBC is also accomplished as well as a clear presentation of wiring and therefore the flow between function units [Westphal, 2011c].

From the EBC developer community an ebc.xml -designer evolved. Mike Bild developed an ebc.xml -designer that enables graphical development of FD and EBC. Boards and Parts are standard units which can be named and combined on the work space. Wires and input respectively output pins are used to connect components graphically. Data flowing on the wires between components can be typed. Eventually the complete design is translated into ebc.xml DSL and is further processed by the ebc.xml -compiler [Westphal, 2011c].

Under the project ‘Event-Based Components Tooling’ [Westphal and Lieser, 2011] Ralf Westphal and Stefan Lieser started an open source project. The projects purpose is to provide tools and standard components for EBC like joins and splits. These pre-assembled components lighten and fasten developing software with EBC. Using standard components the resulting code gets more robust gains readability. These EBC tools are available as compiled programming libraries. Also the source code is available through the projects website. At this point, EBC tools are only available for the programming language C#.

3 CONSIDERATION OF FLOW-DESIGN AND EVENT-BASED COMPONENTS IN CONTEXT OF SOFTWARE ARCHITECTURES

Flow-Design introduced a new way of engineering software architectures while Event-Based Components specified the rules of implementation. Next, established software architectures are considered in contrast. In which way FD and EBC differ from event-driven architecture, service-oriented architecture and object-oriented analysis and design is discussed in the following.

3.1 Separating Design and Implementation

Flow-Design is an approach to design software architectures. Therefore it is compared to object-oriented analysis and design. By use of FD approach new architectures respectively possibilities to design architectures arise. Diagrams, like UML for OOAD, are provided to support and document the design stage. Where OOAD puts objects and therefore data into foreground, FD focus on function units which process data. The exchange of input and output data between components is called flow. OOAD maps reality into software by using objects. In doing so, objects are units of data and operations. They encapsulate knowledge and therefore data. Operations represent a particular behavior. Though a software system is built of and described by cooperating objects. The relationships between these objects causes a dependency among them. When changing or maintaining an object-oriented programming (OOP) implementation, this dependency comes to relevance. If the signature of an objects method changes, changes may have to be done in many parts of the software to react on the change. Dependent objects have to be adjusted to regain the systems operability. Implementations of complex software systems result in vast amount of objects. Relations and dependencies between them is hard to see in coding an often not well documented in diagrams. Therefore it is often unclear how much a maintenance or change will affect the software system. Documentation is often not up to date with coding and is itself of high complexity. A OOAD designed software can be described with 13 UML 2.0 diagrams.

Component-based software engineering (CBES) focuses on the idea of reusability. Based on OOAD, CBES targets on creating bigger, more abstract elements. These components are meant to be reused without knowing any details of their implementation. By a high grade of reusability and therefore a higher number of users, components are thought to be more robust. Over time it is supposed to create a portfolio of standard components which can be reassembled and reused in following software projects. These components have fixed contracts, represented through interfaces describing exactly the functional range. Distributed components create the basis for service-oriented and event-driven architectures.

EBC is a methodology for implementing FD as well as an architecture similar to event-driven architecture (EDA) and service-oriented architecture (SOA). It is independent of a particular programming language or platform. Although it builds up rules and specifications for implementation. OOP is the programming paradigm used for implementing EBC, because events are part of many OOP languages, but at least can be implemented through design patterns. Though, unlike EDA and SOA exchanging messages on system level between independent software, EBC is by default an architecture limited to the boundaries of a software.

3.2 Similarities and Differences to Established Software Architectures

SOA builds up on the concept of CBES. CBES manages components via a repository to be reused during software development. In contrary SOA installs components as independent services. Instead of administrating components with a repository, the service components are managed by a service broker within a service catalogue. Service provider publish their provided services to the service broker and service consumer place their service request to the service broker. Message exchange is managed by a service-mediation system, also called enterprise service bus. Communication is synchronously, bidirectional and message oriented. SOA enables communication across independent software as well as using existing services during software development [Masak, 2007, Reussner and Hasselbring, 2009].

EDA uses events to exchanges messages. EDA is not about administrating components or services, but rather to manage the event flows. Events arise from state changes and are triggered by a source to be processed by a sink. Source and sink are independent from each other.

Components, software systems but also hardware like sensors can be source of an event. Via an event channel the events reach the middleware that distributes events to registered sinks. Event receivers, the sinks, have to register themselves with the middleware for certain events. Communication in EDA is asynchronously, unidirectional and event oriented [Bruns, Ralf and Dunkel, Jürgen, 2010].

EBC has commonalities with SOA and EDA. Similar to SOA, EBC builds up components to fulfill a certain service. This service is defined by a contract represented through an interface. Hence EBC is not about building service components providing several services like SOA. EBC creates components to provide one certain, cohesive functionality. This has the effect of loose coupling and minor dependencies. Here lies the analogy to EDA. Where EDA operates across system boundaries, EBC is limited to a single software system. Therefore in contrary to EDA, EBC doesn't need a middleware to transport events from source to sink. EBC solves this via components called Boards. These aggregate the smallest function units, Parts, to Boards. Wiring source and sink, called output and input pins, is arranged by Boards enabling the events to flow between function units. Similar to EDA the communication is unidirectional and event oriented. Although communication is by default synchronously the flow can be made asynchronously and concurrency with instruments of the programming language chosen for implementation [Bruns, Ralf and Dunkel, Jürgen, 2010, Masak, 2007, Reussner and Hasselbring, 2009].

4 CONCLUSION

Flow-Design changes the view in designing software. In contrary to OOAD, FD concentrates on processes respectively function units. It solves dependency and coupling issues by generating a design of separate components which are connected via flows and therefore are independent of each other. Independent components build with FD are more easily to test because they don't make calls to other components. Instead they take input and deliver output. Though no mock-ups are needed for testing. The function units can be tested separately without influence of other components. With Event-Based Components it is possible to translate the design into code and keep design and implementation aligned. Teams and developers can implement the design self-responsible. The internal logic of the function units is not part of the design, this is left to the implementers. FD defines the flow and the contract between components, not the internal structure. The communication between components is accomplished through events. Therefore it is unidirectional, the message flow takes only effect in one direction. Common client-server practices in which the client calls for a service and gets the response are not part of FD and EBC. Therefore this new approach requires software designers and developers to adapt to a new thinking. Aggregating Parts to Boards by wiring input and output pins, means creating components of a higher abstract level. The more function units are aggregated, the more complex the wiring gets. Hence the resulting coding of EBC can be complex and unclear. But tools for designing and the implementation for EBC are already developed. Although C# is the only tool supported programming language at the present time and Ralf Westphals coding examples are all presented in C#.

Ralf Westphal keeps on working on FD and EBC as well as he spreads his new way of thinking in software development. Tending towards automated creation of EBC out of FD models Ralf Westphal introduced the Flow Execution Engine [Westphal, 2010d] which generates Boards and Parts from a design. Afterwards the Parts need to be filled with logic. This relieves developers from wiring Parts to Boards. Hereby he also developed a runtime which is able to execute flows. Although up to now limited to German speaking areas, supported by a growing community FD and EBC gain popularity and attention.

REFERENZEN

BOOCH, G., RUMBAUGH, J., AND JACOBSON, I. 2006. Das UML-Benutzerhandbuch: Aktuell zur Version 2.0. *Addison-Wesley*, München [u.a.].

BRUNS, RALF AND DUNKEL, JÜRGEN 2010. Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. *Springer-Verlag*, Berlin and Heidelberg.

CARZANIGA, A., DI NITTO, E., ROSENBLUM, D. S., AND WOLF, A. L.1998. Issues in Supporting Event-Based Architectural Styles. *Proceedings of the third international workshop on Softwarearchitecture ISAW 98*, pages 17–20. Available: http://portal.acm.org/citation.cfm?doid=288408.288413

CUGOLA, G., DI NITTO, E., AND FUGGETTA, A. 2001. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850.

EUGSTER, P. T., FELBER, P. A., GUERRAOUI, R., AND KERMARREC, A.-M. 2003. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131.

FAISON, E. W. 2006. Event-Based Programming: Taking events to the limit. *Apress and Springer*, Berkeley and Calif. and New York.

FIADEIRO, J. L. AND LOPES, A. 2006. A formal approach to event-based architectures. In Baresi, L. and Heckel, R., editors, *Fundamental Approaches to Software Engineering*, pages 18–32. *Springer-Verlag*, Berlin and Heidelberg.

FOWLER, M. 2003. Patterns of enterprise application architecture. *Addison-Wesley*, Boston.

FOWLER, M. 2004. Inversion of Control Containers and the Dependency Injection pattern. *Available: http://www.martinfowler.com/articles/injection.html*

FREUND, J. AND RÜCKER, B. 2012. Praxishandbuch BPMN 2.0. *Hanser, Carl*, München, 3 edition.

GAMMA, E. 1999. Design patterns: Elements of reusable object-oriented software. *Addison-Wesley*, Reading and Mass, 17 edition.

HEINEMAN, G. T. AND COUNCILL, W. T. 2001. Component-based software engineering: Putting the pieces together. *Addison-Wesley*, Boston.

KOWALEWSKI, B., BUBAK, M., AND BALI´S, B. 2008. An Event-Based Approach to Reducing Coupling in Large-Scale Applications. In Bubak, M., van Albada, G., Dongarra, J., and Sloot, P., editors, *Computational Science – ICCS 2008, volume 5103 of Lecture Notes in Computer Science*, pages 358–367. *Springer-Verlag*, Berlin and Heidelberg.

MARTIN, R. C. 2003. Agile software development: Principles, patterns, and practices. *Prentice Hall*, Upper Saddle River and N.J.

MASAK, D. 2007. SOA? Serviceorientierung in Business und Software. *Springer-Verlag*, Berlin and Heidelberg.

MORRISON, J. P. 2010. Flow-Based Programming: A new approach to application development. *J.P. Morrison Enterprises*, Unionville and Ont., 2 edition. *Available: http://www.worldcat.org/oclc/694201092*

READE, C. 1989. Elements of functional programming. *Addison-Wesley*, Wokingham and England.

REUSSNER, R. AND HASSELBRING, W. 2009. Handbuch der Software-Architektur. *Dpunkt-Verl.*, Heidelberg, 2 edition.

WESTPHAL, R. 2010A. Nicht nur außen schön: Event-Based Components. *dotnetpro*, (8.2010):126–133.

WESTPHAL, R. 2010B. One Man Think Tank Gedanken. *Available: http://ralfw.blogspot.de/search/label/Event-based%20Components*

WESTPHAL, R. 2010C. Stecker mit System: Event-Based Components. *dotnetpro*, (7.2010):126–133.

WESTPHAL, R. 2010D. The Architect's Napkin: .NET Software Architecture on the Back of a Napkin. *Available: http://geekswithblogs.net/theArchitectsNapkin/category/11899.aspx*

WESTPHAL, R. 2010E. Zusammenstecken - funktioniert: Event-Based Components. *dotnetpro*, (6.2010):132–138.

WESTPHAL, R. 2011A. Lass es fließen: IEnumerable<T> richtig einsetzen. *dotnetpro*, (3.2011):128–134.

WESTPHAL, R. 2011B. Tools für Event-Based Components, Teil 1: Auf dem Weg zum Autorouter. *dotnetpro*, (1.2011):136–140.

WESTPHAL, R. 2011C. Tools für Event-Based Components, Teil 2: Das Navi für Signale. *dotnetpro*, (2.2011):136–140.

WESTPHAL, R. 2011D. Von Substantiven, Verben und EBCs: Abhängigkeiten in Flow Designs. *dotnetpro*, (4.2011):132–138.

WESTPHAL, R. AND LIESER, S. 2011. Event-Based Components Tooling. *Available: http://ebclang.codeplex.com*