Signet farbig_ Format

**Entwurf und Umsetzung einer**

**Beschreibungssprache für mobile Anwendungen**

Bachelorarbeit von Edmund Senkleiter

Bachelorarbeit

Fakultät für Informatik   
Institut für Softwaretechnologie  
Professur für Programmierung kooperativer Systeme  
Prof. Dr. Michael Koch



betreut von:  
Peter Lachenmaier, M. Sc.

Abgabetermin der Arbeit: 14.01.2015

Hiermit versichere ich an Eides statt, dass die vorliegende Arbeit von mir selbständig verfasst wurde und ich alle verwendeten Quellen, auch Internetquellen, ordnungsgemäß angegeben habe.

Datum, Unterschrift (Name)

Kurzzusammenfassung

Hier steht später eine kurze und prägnante Zusammenfassung der Arbeit in ca. fünf bis maximal zehn Sätzen

Inhaltsverzeichnis

Kurzzusammenfassung i

Inhaltsverzeichnis ii

Abbildungsverzeichnis iv

Abkürzungsverzeichnis v

1 Einleitung 1

1.1 Motivation 1

1.2 Problemstellung 2

1.3 Zielsetzung 2

1.4 Aufbau der Arbeit 4

2 Entwicklung plattformübergreifender mobiler Anwendungen 5

2.1 Bewertungskriterien plattformübergreifender Entwicklungsframeworks 5

2.1.1 Allgemeine Anforderungen 6

2.1.2 UIML 6

2.1.3 FXML 6

2.1.4 XAML 7

2.2 Zusammenfassung existierender Sprachen 7

3 Beschreibungssprache 8

3.1 Anforderungen 8

3.2 Entwurf 8

4 iOS Interpreter 9

4.1 Architektur 9

4.2 Entwurf 9

5 Beispiel iOSTemplateLanguage Applikation 10

5.1 Anforderungen 10

5.2 CommunityMashup 10

Umsetzung 10

5.3 /Funktionsweise 10

5.4 Fazit 10

5.4.1 Erreichter Funktionsumfang 10

5.4.2 Verbesserte Performanz 10

6 Zusammenfassung und Ausblick 11

Literaturverzeichnis 12

Abbildungsverzeichnis

[Abbildung 1: Übersicht Risikobewertung und Bewertungsmodell **Fehler! Textmarke nicht definiert.**](#_Toc211259057)

[Abbildung 2: Zuordnung der IT-Risikofaktoren zu den Managementtätigkeiten **Fehler! Textmarke nicht definiert.**](#_Toc211259058)

Abkürzungsverzeichnis

App Applikation

Aufl. Auflage

Bd. Band

bzw. beziehungsweise

ca. circa

d. h. das heißt

et al. et altera

etc. et cetera

f. folgende

ggf. gegebenenfalls

Hrsg. Herausgeber

i. d. R. in der Regel

S. Seite

s. siehe

sog. sogenannt

u. a. unter anderem

usw. und so weiter

v. a. vor allem

vgl. vergleiche

z. B. zum Beispiel

z. T. zum Teil

# Einleitung

## Motivation

Mobile Endgeräte wie Smartphones oder Tablets sind aus unserem Alltag nicht mehr wegzudenken. Die Verkaufsrekorde werden jährlich gebrochen und die klassischen Desktop PCs sowie Laptops langsam verdrängt(Bitkom 2014). Mobile Anwendungen gewinnen dabei natürlich auch immer mehr an Bedeutung und damit auch die Anzahl an Softwareentwicklern in diesem Bereich. Genauso wie für die „klassischen“ Geräte, existieren auch mehrere unterschiedliche Betriebssysteme für mobile Geräte (Apple iOS, Google Android, Microsoft Windows Phone usw.). Um nun eine möglichst große Nutzerzahl abzudecken, ist es erforderlich die Anwendung für mehrere Betriebssystem / Plattformen zu entwickeln, was aber mit erheblichem Mehraufwand verbunden sein kann, gerade auch wenn man an Erweiterungen oder Korrekturen denkt. Die Apps können nativ für jede Plattform extra entwickelt werden, wodurch sie zwar sehr effizient ausgeführt werden und Zugriff auf alle Geräte- /Betriebssystemfunktionen bieten. Dies jedoch eben nur auf der dafür entwickelten Plattform. Im Gegensatz dazu werden Web Apps im Browser ausgeführt was derzeit mit fast jedem mobilen Endgerät möglich ist, sodass hier nicht für jede Plattform extra Code geschrieben werden muss. Die Benutzerschnittstelle entspricht jedoch eher einer gewöhnlichen Webseite als den nativen Elementen. Einen Kompromiss zwischen beiden Varianten bieten hybride Apps, die zwar auch durch Web Technologien wie bspw. HTML5 realisiert werden, zur Laufzeit jedoch nativ in einem WebView Container laufen und somit auch Zugriff auf viele Geräte / Betriebssystemfunktionen bieten. Welche Variante die passendere ist lässt sich an Merkmalen wie Performanz, Nutzung nativer Funktionalitäten, Installation, Erreichbarkeit, offline Nutzbarkeit usw. festmachen(Davit Svand 2014). Ständige Leistungssteigerungen erweitern die Entwicklungs- und somit Einsatzmöglichkeiten, jedoch werden mobile Geräte hauptsächlich zur schnellen Informationsversorgung genutzt(Sonja Knaub 2014), was größtenteils nur einfaches Abrufen und Darstellen bestimmter Informationen aus einer (halb-) öffentlichen Quelle ist.

Im Rahmen dieser Arbeit wird speziell auf mobile Anwendungen eingegangen, die personenzentrierte Daten aus einem sog. Mashup laden und dem Nutzer zur Verfügung stellen. Für die Mensch und Computer Tagung 2014 wurde bereits eine solche App für iOS entwickelt und veröffentlicht. Diese unterstützt die Teilnehmer der Tagung, indem Informationen zu Personen (Teilnehmern), Organisationen und Inhalten (bspw. Vorträgen) anzeigt werden. Informationen sind bspw:

* Kontaktdaten zu Teilnehmern, Referenten, Organisationen
* Überblick sowie detailliertere Angaben zu einem Vortrag
* Auflistung aller Vorträge zu einer bestimmten Person
* Agenda / Programmplan der Konferenz

Den Nutzern wird außerdem noch die Möglichkeit gegeben selbst Informationen, in Form von Kommentaren und Bewertungen, den Inhalten hinzuzufügen. Das eben genannte Mashup, genauer gesagt Community Mashup, stellt eine Schnittstelle zwischen externen sozialen Dienstleistungen, wie z.B Facebook, Twitter, Flickr, und Endbenutzer Anwendungen, wie der eben genannten App Muc2014 dar. Im Mashup werden personenzentrierte Daten gesammelt, gefiltert und konfigurierbar vereinheitlicht (Lachenmaier 2013).

## Problemstellung

Die Mensch und Computer 2014 Tagungsapp ist derzeit nur nativ für Apples Betriebssystem iOS entwickelt worden, sodass eine Portierung auf andere Plattformen wie Android oder Windows Phone mit viel Entwicklungsaufwand verbunden ist. Dasselbe Problem hat man auch bei Änderungen, die mehrere oder sogar alle angebotenen Plattformen betreffen. Außerdem müssen alle Änderungen / Updates bei Apple erneut den sog. „App Review Process“ durchlaufen der mehrere Tage dauern kann, das verhindert kurzfristige Änderungen.

Ein weiteres Problem ist der langwierige erste Start der App, bei dem alle Informationen zunächst abgerufen, lokal gespeichert und // in Beziehung zu einander gesetzt werden müssen //. Das kann bei großen Datenmengen und schlechter Konnektivität mehrere Minuten dauern, währenddessen die App nicht nutzbar ist.

## Zielsetzung

Ziel dieser Arbeit ist der Entwurf sowie die Umsetzung einer Beschreibungssprache für mobile Anwendungen um die zuvor genannten Probleme zu lösen. Mit der auf JSON basierenden Beschreibungssprache werden Aussehen, Inhalte sowie mögliche Funktionalitäten der App plattformunabhängig definiert. Damit liegt für alle Plattformen eine gemeinsame Beschreibung vor, die jedoch plattformspezifisch interpretiert wird, sodass nur der Interpreter plattformabhängig entwickelt werden muss. Im Implementierungs- bzw. Umsetzungsteil dieser Arbeit wird nur der Interpreter für iOS als Beispiel und zur Evaluation entwickelt. Ein natives Framework an abstrakteren Grundfunktionalitäten ermöglicht das Zuweisen von Aktionen zu Benutzerschnittstellenelementen und definiert damit die erwähnten möglichen Funktionalitäten der App. Unter abstrakten Grundfunktionalitäten kann man sich Einträge ins Telefonbuch, das Öffnen eines Links, den Wechsel zu einer anderen Ansicht vorstellen. Darauf wird später noch genauer eingegangen. Die Anwendungsbeschreibung soll von einer externen Quelle wie dem Community Mashup heruntergeladen und zur Laufzeit interpretiert werden, was bedeutet, dass das Mashup keine reine Informationsquelle ist, sondern auch das Aussehen sowie die möglichen Funtionalitäten der App mithilfe von Templates über eine REST Schnittstelle definiert. Auf diese Art und Weise können jegliche Korrekturen oder Verbesserungen, die nicht den Sprachumfang bzw. dessen Interpretation betreffen, schnell und mit geringem Aufwand für alle Plattformen gleichzeitig umgesetzt werden. Ein weiteres Ziel ist die Verkürzung der App Initialisierung durch sog. „Caching“. Das wird erreicht indem nur dann Beschreibungsinformationen heruntergeladen werden, wenn sie das erste Mal gebraucht werden oder eine Aktualisierung gewünscht bzw. notwendig ist. Bereits geladene Informationen sind lokal gespeichert und können auch offline abgerufen werden. Damit reduzieren sich die benötigten Informationen bei Anwendungsstart auf das Notwendigste, nämlich der ersten Ansicht. Ein weiteres Ziel ist die entwickelte Sprache sowie den dazugehörigen Interpreter für iOS zu nutzen, indem eine App nach Vorbild der MuC2014 umgesetzt wird. Dabei liegt der Schwerpunkt nicht darauf, die App in der Form einzusetzen bzw. zu Veröffentlichen sondern zu Zeigen, dass mit der Sprache alle grundlegenden Funktionen umgesetzt und die Benutzbarkeit, insbesondere die Ladezeiten sich verbessert haben.

Entwurf sowie Implementierung der UI Beschreibungssprache.

* Beschreibungssprache
  + Aussehen
  + Daten
* Übertragung Server-Client
  + Zur Laufzeit änderbar
* Interpretation auf mobile Clientseite
  + Framework
  + Verschiedene Plattformen
* Fähigkeiten
  + Caching
  + Aktualisierung
  + Listen von den oben motivierten Objekten
  + Details dazu
  + Miteinander verlinkt
  + => Browsing App
* Implementierung/Test für iOS bzw. iPhone

## Aufbau der Arbeit

Am Anfang werden aktuelle Beschreibungssprachen anhand einiger Bewertungskriterien näher untersucht:

* Bewertungskriterium a
* B

Abschließend wird begründet warum eine eigene Beschreibungssprache entwickelt werden soll.

Die Entwicklung der Beschreibungssprache wird in Kapitel 3 vorgestellt. Dazu werden zunächst alle Anforderungen beschrieben, die sich aus Kapitel 2 und den Zielen der Arbeit ergeben. Im folgenden Entwurf werden diese dann umgesetzt.

Die Evaluation der entworfenen Sprache beginnt bereits in Kapitel 4 mit der Entwicklung eines Interpreters für iOS. $Architektur , $Entwurf

Nachdem der Interpreter nun entworfen ist, folgt die Umsetzung einer Beispielanwendung nach Vorbild der MuC2014 App. Hierbei werden wieder Anforderungen und die Verbindung zum Community Mashup erläutert. Anschließend folgen Details zum Aufbau und der Funktionsweise der App. Zum Schluss gibt es noch ein Fazit, dass insbesondere auf den erreichter Funktionsumfang und die verbesserte Performanz eingeht.

Am Ende der Arbeit werden die zu Beginn genannten Ziele in Zusammenhang mit den erreichten Ergebnissen gebracht und ein Ausblick auf weitere Entwicklungen sowie Verbesserungen gegeben

# Entwicklung plattformübergreifender mobiler Anwendungen

Für die Entwicklung plattformübergreifender mobiler Anwendungen gibt es 3 verschiedene Ansätze (Heitk, Hanschke 2013):

1. Nutzung einer Beschreibungssprache dir zur Laufzeit interpretiert wird
2. Nutzung einer gemeinsamen Codebasis, die plattformspezifisch übersetzt wird
3. Modellgetriebener Ansatz

Der erste Ansatz entspricht den sog. Web und hybriden Apps. Die App wird durch eine gemeinsame Beschreibungssprache für alle Plattformen entwickelt und erst zur Laufzeit interpretiert. Dazu muss jedoch für jede Plattform der Interpreter nativ entwickelt worden sein. Web Apps nutzen diesen Ansatz, wobei als Beschreibungssprachen HTML5, CSS und JavaScript verwendet werden und der Interpreter der native Webbrowser ist. Um deutlich mehr Geräte- / Betriebssystemfunktionen nutzen zu können, kann hier ein eigener Interpreter entwickelt werden, worauf in Kapitel 4 näher eingegangen wird.

Der zweite Ansatz ähnelt dem ersten, jedoch wird die Beschreibung zunächst plattformspezifisch übersetzt und damit gleichzeitig für alle Plattformen eine native Anwendung erstellt. Das hat natürlich den Vorteil, dass nichts zur Laufzeit interpretiert werden muss und die Anwendung somit deutlich effizienter ausgeführt wird.

Der dritte Ansatz versucht, wie der Name schon sagt, über eine Modelldefinition plattformübergreifende Apps zu generieren.

Im Folgenden werden aktuelle Beispielframeworks die dem ersten Ansatz folgen vorgestellt. Ein bekanntes Open Source Framework für die Entwicklung von hybriden Apps ist PhoneGap (PhoneGap 2014). Die Anwendungen werden mit HTML, CSS und JavaScript entwickelt, jedoch wird der Webinterpreter um eine JavaScript API in Kombination mit einer Brücke zur Hardware erweitert. Ein Beispiel für die Nutzung eines eigenen Interpreters liefert das Framework Titanium Mobile (Titanium Mobile 2014). Die Benutzerschnittstelle, Logik und Daten werden in JavaScript programmiert und zur Laufzeit interpretiert.

Evtl DSLs für mobile Apps

## Bewertungskriterien plattformübergreifender Entwicklungsframeworks

App

UI

Einsatzzeit im Entwicklungsprozess

Generierung?

Interpretation zur Laufzeit?

### Allgemeine Anforderungen

-Generisch / Plattformunabhängig

-UI Entwicklung ohne spezifische Kenntnisse über die Zielplattformen

-Trennung von UI und Logik Code (Struktur, Style, Inhalt und Verhalten)

-Beschreibung von Präsentation sowie Nutzung der Daten (Darstellungsart, Event ausführung) / Bietet neben reiner Beschreibung der Darstellung, Möglichkeit Aktionen/ Events aufrufen zu lassen

-Vordefinierte abstrakte Aktionen / Event Bspw. Text senden, Kontakt hinzufügen, Rufnummer wählen, Öffnen anderer Applikation, zu anderen Ansichten wechseln / diese öffnen

-konkrete Implementierung durch jeweilige Zielplattform

### UIML

Allgemeines

Ziele der Sprachentwickler

1. Plattform unabhängige Beschreibung von Benutzerschnittstellen
2. Generische Verbindung zwischen Benutzerschnittstelle und Applikationslogik

Besondere Eigenschaften / Vorteile gegenüber anderen Sprachen

Konkrete Beispiele

Benutzer Interaktion

### FXML

Quelle:<http://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html#overview>

### XAML

Quelle: <http://msdn.microsoft.com/en-us/library/ms752059.aspx>

## Zusammenfassung existierender Sprachen

Schluss ziehen warum eigene Sprache

Gründe für den Entwurf einer neuen Sprache

# Beschreibungssprache

## Anforderungen

Siehe Kapitel 2

Erweiterbarkeit

Generisch für mobile Plattformen

## Entwurf

# iOS Interpreter

Sprachmerkmal -> UITableViewController

## Architektur

## Entwurf

Technologieauswahl

Restkit

# Beispiel iOSTemplateLanguage Applikation

Um die entwickelte Beschreibungssprache einer ersten Evaluation zu unterziehen, bedarf es einer Beispiel App, deren Entwurf und Umsetzung im Folgenden nun beschreiben wird. Als Vorlage dient die App Muc2014[src], welche bereits im Rahmen einer Masterarbeit entwickelt wurde. Ziel der Beispiel App ist es, möglichst viele unterschiedliche Funktionalitäten der Vorlage mithilfe der Beschreibungssprache zu realisieren.

## Anforderungen

Das sind Anforderungen an konkrete App

* Datenquelle: CommunityMashup MUC
* JSON Parsing und Mapping
* Persistenz der Daten
* Geschwindigkeit, Verwendung mit geringer Bandbreite
* UI Design stark an Vorlagen App orientiert

## CommunityMashup

Architektur

Funktionsweise

Schnittstellen

Templates

## Umsetzung/Funktionsweise

Hier Schreenshots der App mit Beschreibung

## Fazit

### Erreichter Funktionsumfang

### Verbesserte Performanz

# Zusammenfassung und Ausblick

Literaturverzeichnis

Phanouriou, C. (2000). UIML: A Device-Independent User Interface Markup. Retrieved from http://vtechworks.lib.vt.edu/handle/10919/28625