



## B A C H E L O R A R B E I T

in der Fachrichtung  
Wirtschaftsinformatik

## T H E M A

### **Konzeption einer DSL zur Beschreibung von Benutzeroberflächen für profil c/s auf der Grundlage des Multichannel-Frameworks der deg**

Eingereicht von:	Niels Gundermann (Matrikelnr. 5023) Woldegker Straße 34 17033 Neubrandenburg E-Mail: gundermann.niels.ng@googlemail.com
Erarbeitet im:	7. Semester
Abgabetermin:	13. Februar 2015
Gutachter:	Prof. Dr. Johannes Brauer
Co-Gutachter:	
Betrieblicher Gutachter:	Dipl.-Ing. Stefan Post Woldegker Straße 12 17033 Neubrandenburg Tel.: 0395/5630553 E-Mail: stefan.post@data-experts.de

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Tabellenverzeichnis</b>	<b>iv</b>
<b>Listings</b>	<b>vi</b>
<b>1 Motivation</b>	<b>1</b>
<b>2 Problembeschreibung und Zielsetzung</b>	<b>3</b>
2.1 Allgemeine Anforderungen an Benutzeroberflächen von pro- fil c/s . . . . .	3
2.2 Umsetzung der Benutzerschnittstellen für mehreren Plattfor- men in der deg (Ist-Zustand) . . . . .	5
2.3 Probleme des Multichannel-Frameworks . . . . .	6
2.4 Zielsetzung . . . . .	7
<b>3 Domänenspezifische Sprachen</b>	<b>8</b>
3.1 Begriffsbestimmungen . . . . .	8
3.2 Anwendungsbeispiele . . . . .	13
3.3 Model-Driven Software Development (MDSD) . . . . .	13
3.4 Abgrenzung zu GPL . . . . .	14
3.5 Vor- und Nachteile von DSL gegenüber GPL . . . . .	14
3.6 Interne DSL . . . . .	14
3.6.1 Implementierungstechniken . . . . .	14
3.7 Externe DSL . . . . .	14
3.7.1 Implementierungstechniken . . . . .	14
3.7.2 Parser . . . . .	14
3.8 Textuelle vs. Nicht-Textuelle DSL . . . . .	14

<b>4</b>	<b>Entwicklung einer Lösungsidee</b>	<b>15</b>
4.1	Allgemeine Beschreibung der Lösungsidee . . . . .	15
4.2	Architektur . . . . .	16
4.3	Vorteile gegenüber dem Multichannel-Framework . . . . .	16
<b>5</b>	<b>GUI-DSL</b>	<b>17</b>
5.1	Beschreibung der Anforderung an die GUI . . . . .	17
5.2	Vorstellung ausgewählter DSL zur Beschreibung von GUIs . .	17
5.2.1	The Snow . . . . .	17
5.2.2	glc-dsl . . . . .	17
5.2.3	Sculptor . . . . .	17
5.3	Bewertung . . . . .	17
<b>6</b>	<b>Evaluation des Frameworks zur Entwicklung der DSL</b>	<b>18</b>
6.1	Vorstellung ausgewählter Frameworks . . . . .	18
6.1.1	PetitParser . . . . .	18
6.1.2	Xtext . . . . .	18
6.1.3	MPS . . . . .	18
6.2	Vergleich und Bewertung der vorgestellten Frameworks . . .	18
<b>7</b>	<b>Aufteilung der Anforderungen auf Sprache und Generator</b>	<b>19</b>
7.1	Anforderung an die neue DSL . . . . .	19
7.2	Anforderung an den Generators . . . . .	19
<b>8</b>	<b>Entwicklung einer DSL zur Beschreibung der GUI in profil c/s</b>	<b>20</b>
8.1	Analyse der Metadaten der GUI . . . . .	20
8.2	Syntax . . . . .	20
8.3	Semantik . . . . .	20
<b>9</b>	<b>Entwicklung des Generators für das Generieren von Klassen für das Multichannel-Framework</b>	<b>21</b>
9.1	WAM-GUI Architektur . . . . .	21
9.2	Syntax und Semantik für die Beschreibung der GUIs . . . . .	21
9.3	Umsetzung des frameworkspezifischen Generators . . . . .	21
<b>10</b>	<b>Zusammenfassung und Ausblick</b>	<b>22</b>

---

<b>Titel anhang a</b>	<b>xi</b>
<b>Glossar</b>	<b>xii</b>
<b>Literaturverzeichnis</b>	<b>xiii</b>

# Abbildungsverzeichnis

2.1	Web-Client . . . . .	4
2.2	Standalone-Client . . . . .	5
2.3	MC-Framework . . . . .	6
4.1	neuerAnsatz . . . . .	15

# Tabellenverzeichnis

# Listings

# Kapitel 1

## Motivation

In der heutigen Zeit werden Programme auf vielen unterschiedlichen Geräten<sup>1</sup> von ausgeführt. Die Benutzeroberfläche neben der internen Umsetzung immer ein wichtiger Faktor, der für den Erfolg einer Anwendung eine große Rolle spielt. [LW] Damit einher geht die Usability<sup>2</sup> einer Anwendung. Denn eine [...] *schlechte Useability führt zu Verwirrung und Miss- bzw. Unverständnis beim Kunden* [Use12]. Dadurch geht letztendlich Umsatz verloren. Wenn ein Programm auf unterschiedlichen Geräten ausgeführt wird, muss der Entwickler bei der traditionellen Entwicklung<sup>3</sup> mehrere Graphical User Interfaces (GUI)<sup>4</sup> bereitstellen. Folglich werden mehrere GUIs mit unterschiedlichen Toolkits oder Frameworks entworfen. Diese Frameworks haben einen starken imperativen Character, sind schwer zu erweitern und sie verhalten sich unterschiedlich abhängig von der speziellen Implementierung. [KB11] Der Entwickler muss das GUI bei diesem Ansatz für jedes Framework explizit beschreiben.

Ein anderer Ansatz zur Beschreibung von Benutzeroberflächen ist das Model-Driven Development. Damit sollen UIs anhand der implementierten Funktionen automatisch erzeugt werden können. [SKNH05] Allerdings wird die Darstellung dieser generierten UIs von der Darstellung von traditionell implementierter Benutzerschnittstellen übertroffen. [MHP99].

Eine Überlegung, die sich daraus ergibt, ist, ob man diese beiden Ansätze

---

<sup>1</sup>Desktop, Smartphone, Tablet

<sup>2</sup>Siehe Glossar: Usability

<sup>3</sup>Siehe Glossar: Traditionelle UI-Entwicklung

<sup>4</sup>Siehe Glossar: GUI



zur Implementierung von UIs (traditionell und Model-Driven) verbinden kann. Somit kann die genaue Beschreibung der Darstellung mit einer höheren Abstraktion verbunden werden.

In dieser Arbeit wird versucht diese Idee umzusetzen. Bei der Umsetzung wird sich auf die UIs der Anwendung *profil c/s*. Profil c/s ist INVEKOS<sup>5</sup>-Programm welches von der deg als Client-Server-Anwendung entwickelt wird. In dieser Arbeit wird versucht diese Idee an einem ausgewählten Beispiel umzusetzen.

---

<sup>5</sup>Siehe Glossar: InVeKoS

## Kapitel 2

# Problembeschreibung und Zielsetzung

### 2.1 Allgemeine Anforderungen an Benutzeroberflächen von profil c/s

Die wichtigste (primäre) Anforderung für diese Arbeit bezieht sich auf den Client von profil c/s. Dieser soll sowohl in Web-Browsern (Web-Client) als auch standalone auf einem PC (Standalone-Client) ausgeführt werden können.

Um eine effiziente Arbeitsweise zu ermöglichen, kamen (sekundäre) Anforderungen wie *Erweiterbarkeit der Frameworks*, *Abstraktion* und die *Ausdruckskraft*<sup>1</sup> der Sprachkonstrukte, die zur Entwicklung verwendet werden.

In Abbildung 2.1 und Abbildung 2.2 ist das GUI eines Zuwendungsblatt<sup>2</sup> eines Förderantrag<sup>3</sup> zu sehen. Für den Aufbau sind nur die Tabelle und die darunter stehenden Buttons, sowies das Bemerkungsfeld (im Web-Client auf der rechten Seite und im Standalone-Client in der Mitte) von Bedeutung. Dass der Aufbau der GUI in beiden Clients ähnlich ist, liegt an der Umsetzung der GUI.

---

<sup>1</sup>Siehe Glossar: Ausdruckskraft

<sup>2</sup>Siehe Glossar: Zuwendungsblatt

<sup>3</sup>Siehe Glossar: Förderantrag

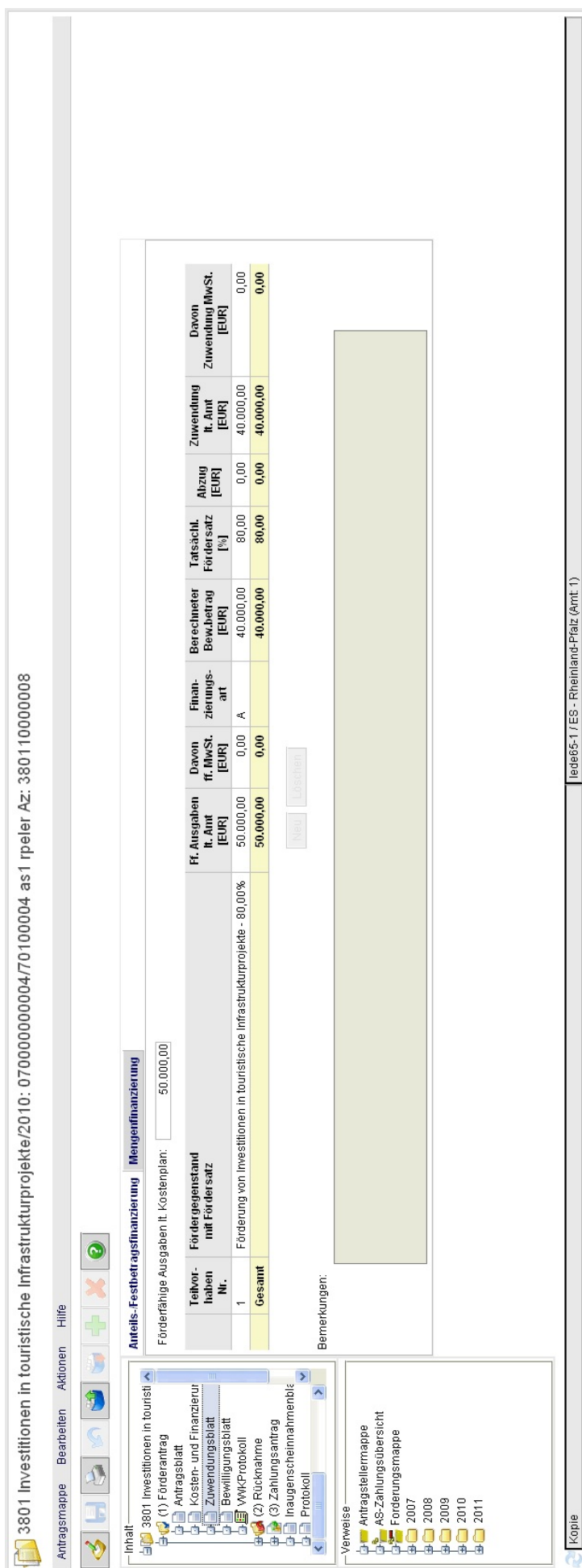


Abbildung 2.1: Web-Client: Zuewndungsblatt [deG07]

Fördergegenstand mit Fördersatz	ff. Ausgaben lt. Amt [EUR]	Finanzierungsart	Berechneter Bew.betrag [EUR]	Tatsächl. Fördersatz [%]	Abzug [EUR]	Zuwendung lt. Amt [EUR]
Erweiterung vereinseigener Sportstätten - 75,0...	50.000,00	A	37.500,00	75,00	0,00	37.500,00
Ausnahmen - 30,00%	20.000,00	A	6.000,00	30,00	0,00	6.000,00
Neubau kommunaler Sportstätten - 75,00%	90.000,00	A	67.500,00	75,00	0,00	67.500,00
Modern. vereinseigener Sportstätten - 75,00%	80.000,00	A	60.000,00	75,00	0,00	60.000,00
Instand. vereinseigener Sportstätten - 75,00%	50.000,00	A	37.500,00	75,00	0,00	37.500,00
<b>Gesamt</b>	<b>290.000,00</b>		<b>208.500,00</b>	<b>71,90</b>	<b>0,00</b>	<b>208.500,00</b>

Abbildung 2.2: Standalone-Client: Zuwendungsblatt [deG07]

## 2.2 Umsetzung der Benutzerschnittstellen für mehreren Plattformen in der deg (Ist-Zustand)

Für die Umsetzung der primären Anforderung wäre es möglich gewesen für den Web-Client und dem Standalone-Client separate GUIs mit unterschiedlichen Frameworks zu entwickeln. Die deg hat jedoch eine Lösung erarbeitet mit der es möglich ist, ein einmal beschriebenes GUI auf mehrere Plattformen zu portieren. Das reduziert den Aufwand zur Entwicklung neuer GUIs, durch eine höhere Abstraktion. Zugleich fördert die einmalige Beschreibung auch einen ähnlichen Aufbau der GUI im Web- und Standalone-Client. Die Lösung der deg ist das *Multichannel-Framework* (MCF). Die Architektur des Multichannel-Frameworks ist in Abbildung 2.3 zu entnehmen. Innerhalb dieses Frameworks werden die GUIs mittels so genannter *Präsentationsformen* beschrieben. Aus Präsentationsformen mithilfe der *Component-Factories* GUIs erzeugt werden, die auf unterschiedlichen Frame-

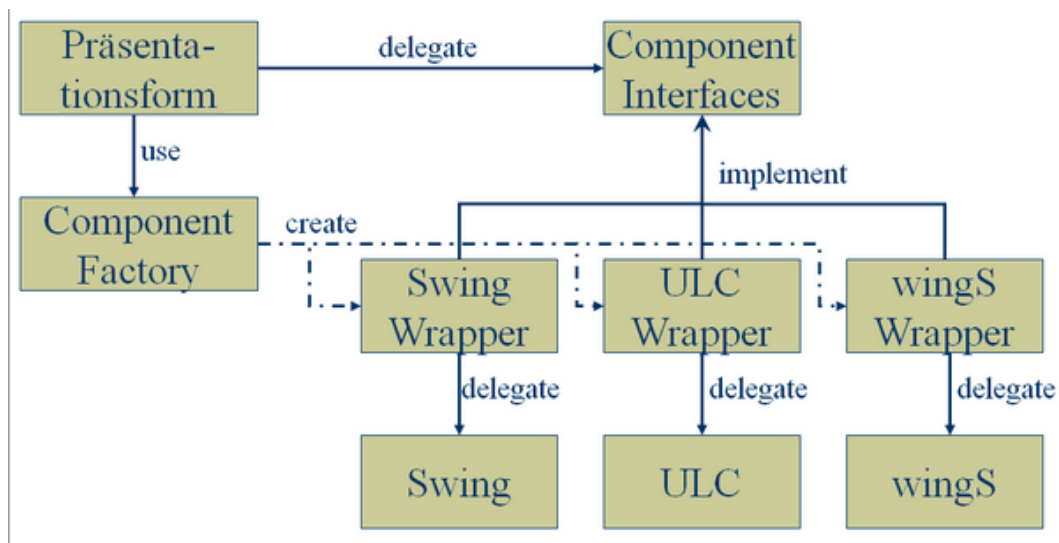


Abbildung 2.3: Architektur des Multichannel-Frameworks [Maa07]

works basieren<sup>4</sup> und das *Component-Interface* implementieren. Das *Component-Interface* wird für die Interaktion mit den Komponenten der unterschiedlichen Frameworks benötigt. Mit dem MCF ist die deg in der Lage ihre GUIs für das *Swing*<sup>5</sup>-Framework und für das *wingS*<sup>6</sup>-Framework mit nur einer GUI-Beschreibung zu erzeugen.

## 2.3 Probleme des Multichannel-Frameworks

Beim Einsatz des MCF treten jedoch Probleme auf. Das erste Problem bezieht sich auf die integrierten Frameworks (Swing und wingS). Beide Frameworks sind verwaltet und werden nicht mehr gewartet. Um auch in der Zukunft den Anforderungen der Kunden nachkommen zu können müssten beide Frameworks von den Entwicklern der deg selbst weiterentwickelt werden. Eine andere Möglichkeit wäre es, wenn die deg andere und modernere Frameworks einsetzt um den nötigen Support der Framework-Entwickler nutzen zu können.

Das MCF ist in der Theorie so konzipiert, dass es leicht sein sollte neue Frameworks zu integrieren (siehe Abbildung 2.3. In der Praxis wurde die Einfachheit einer solchen Integration jedoch widerlegt. Ein Problem, wel-

<sup>4</sup>Hier: Swing, ULC und WingS. Wobei ULC bei der deg nicht mehr im Einsatz ist.

<sup>5</sup>Siehe Glossar: Swing

<sup>6</sup>Siehe Glossar: wingS

ches bei der Integration neuer Frameworks aufkommt, ist, dass sich das MCF sehr stark an Swing orientiert und die GUIs vor allem vom GridBagLayout<sup>7</sup> stark beeinflusst sind. Ein solches Layout steht nicht in allen Frameworks zur Verfügung. Da die Beschreibung der GUI über ein solches Layout vollzogen wird, ist es der Umgang mit dem GridBagLayout<sup>8</sup> innerhalb des Frameworks eine Voraussetzung für die Integration in das MCF. Zusammenfassend sind folgende Probleme des MCF zu nennen:

- verwendeten Frameworks sind inaktuell
- Starke Orientierung an Swing

## 2.4 Zielsetzung

Das langfristige Ziel der deg bzgl. des MCF ist es, eine Lösung zu entwickeln, welche das MCF ablösen kann. Anzustreben ist eine Lösung, die neben den primären Anforderungen die sekundären Anforderung besser umsetzt als das MCF.

In dieser Arbeit wird ein Ansatz untersucht, bei dem es möglich ist, die oben genannten sekundären Anforderungen besser umzusetzen. Kern des Ansatzes ist eine DSL, mit deren Hilfe die UIs beschrieben werden sollen. Eine DSLs kann so konzipiert werden, dass sie ausreichend abstrakt, erweiterbar und ausdrückstärker ist als das MCF. Die primären Anforderungen dürfen dabei nicht außer Acht gelassen werden.

Die genaue Lösungsidee mittels DSL, welche in dieser Arbeit verfolgt wird, ist in Kapitel 4 beschrieben.

---

<sup>7</sup>Siehe Glossar: GridBagLayout

<sup>8</sup>Siehe Glossar: GridBagLayout

# Kapitel 3

## Domänenspezifische Sprachen

### 3.1 Begriffsbestimmungen

#### Sprache/Programmiersprache

Rein formal betrachtet ist eine Sprache ist eine beliebige Teilmenge aller Wörter über einem Alphabet. *Ein Alphabet ist eine endliche, nichtleere Menge von Zeichen (auch Symbole oder Buchstaben genannt)* [Hed12, S.6]. Zur Verdeutlichung der Definition einer Sprache sein  $V$  ein Alphabet und  $k \in \mathbb{N}^1$ . *Eine endliche Folge  $(x_1, \dots, x_k)$  mit  $x_i \in V (i = 1, \dots, k)$  heißt Wort über  $V$  der Länge  $k$*  [Hed12, S.6].

Bei Programmiersprachen grenzt die Bestandteile einer Sprache wie folgt ab:

#### abstrakte Syntax

Die abstrakte Syntax ist eine Datenstruktur, welche die Kerninformationen eines Programms beschreibt. Sie enthält keinerlei Informationen über Details bezüglich der Notation. Zur Darstellung dieser Datenstruktur werden abstrakte Syntaxbäume genutzt. [VBK<sup>+</sup>13, S.179].

---

<sup>1</sup> $\mathbb{N}$  ist die Menge der natürlichen Zahlen einschließlich der Null. [Hed12, S. 6]

**konkrete Syntax**

Die konkrete Syntax beschreibt die Notation der Sprache. Demnach bestimmt sie, welche Sprachkonstrukte der Nutzer einsetzen kann, um ein Programm in dieser Sprache zu schreiben. Die konkrete Syntax wird in so genannten Parse-Bäumen (konkrete Syntaxbäume) dargestellt. [Aho08, S.87]

**statische Semantik**

Die statische Semantik beschreibt die Menge an Regeln bezüglich des Typ-Systems, die ein Programm befolgen muss. [VBK<sup>+</sup>13, S.26]

**ausführbare Semantik**

Die ausführbare Semantik ist abhängig vom Compiler. Sie beschreibt wie ein Programm zu seiner Ausführung funktioniert. [VBK<sup>+</sup>13, S.26]

Programmiersprachen werden dazu verwendet, um mit einem Computer Instruktionen zukommen zu lassen. [FP11, S.27] [VBK<sup>+</sup>13, S.27]

**General Purpose Language (GPL)**

Bei GPLs handelt es sich um Programmiersprachen, die Turing-vollständig sind. Das bedeutet, dass mit einer GPL alles berechnet werden kann, was auch mit einer Turing-Maschine<sup>2</sup> berechenbar ist. Völter et. Al. behauptet, dass alle GPLs aufgrund dessen untereinander austauschbar. Dennoch sind Abstufungen bei der Ausführung dieser Programmiersprachen zu machen. Unterschiedliche GPL sind für spezielle Aufgaben optimiert. [VBK<sup>+</sup>13, S.27f]

---

<sup>2</sup>Siehe Glossar: Turing-Maschine



### Domain Specific Language (DSL)

Eine DSL ist eine Programmiersprache, welche für eine bestimmte Domain<sup>3</sup> optimiert ist. [VBK<sup>+</sup>13, S.28] Das Entwickeln einer DSL ermöglicht es, die Abstraktion der Sprache der Domäne anzupassen. [gho11, S.10] Das bedeutet, dass Aspekte, welche für die Domäne unwichtig sind, auch von der Sprache außer Acht gelassen werden können (Abstraktion). Die Semantik und Syntax sollten dieser Abstraktionsebene angepasst sein. Darüber hinaus sollte ein Programm, welches in einer DSL geschrieben wurde, alle Qualitätsanforderungen erfüllen, die auch bei einer Umsetzung des Programms mit anderen Programmiersprachen realisiert werden. [gho11, S.10f] Eine DSL ist demnach in ihren Ausdrucksmöglichkeiten eingeschränkt. Je stärker diese Einschränkung ist, desto besser ist die Unterstützung der Domäne sowie die Ausdruckskraft der DSL. [FP11, S.27f] In manchen Fällen findet eine Unterscheidung zwischen technischen DSLs und fachlichen DSL statt. Markus Völter unterscheidet diese beiden Kategorien im Allgemeinen dahingehend, dass technische DSLs von Programmierern genutzt werden und fachliche DSL von Personen, die keine Programmierer sind (bspw. Kunden bzw. Personen, die sich in der Domäne auskennen). [VBK<sup>+</sup>13, S.26]

### Grammatik

Grammatiken und insbesondere Grammatikregeln können dazu verwendet um Sprachen zu beschreiben und somit auch den Aufbau eines Computerprogramms. [Hed12, S.23f] Für die Definition einer Grammatik verweise ich auf den Praxisbericht [Gun14, S.5ff]. Grammatiken können in einer Hierarchie dargestellt werden (*Chomsky-Hierarchie*). [S.32f] Bei Programmiersprachen handelt es sich dabei um *kontextfreie Sprachen*, da diese Sprachen entscheidbar sind und somit von einem Compiler<sup>4</sup> verarbeitet werden. [Hed12, S. 16f]

---

<sup>3</sup>Siehe Glossar: Domäne

<sup>4</sup>Siehe Glossar: Compiler

### **Parser**

Ein Parser ist ein Teil der Infrastruktur der DSL. [gho11, S.211] Er ist dafür verantwortlich aus dem DSL-Script ein Output zu generieren, mit dem weitere Aktionen durchgeführt werden können. [gho11, S.212] Der Output wird in Form eines Syntax-Baums (Parse-Baum) (AST<sup>5</sup>) generiert. [FP11, S.47] Ein solcher Baum ist laut Martin Fowler eine weitaus nutzbarere Darstellung dessen, was mit dem DSL-Script dargestellt werden soll. Daraus lässt sich auch das semantische Model generieren. [FP11, S.48]

### **Semantisches Model**

Das semantische Model ist eine Repräsentation dessen, was mit der DSL beschrieben wurde. Es wird laut Martin fowler auch als das Framework oder die Bibliothek betrachtet, welche von der DSL nach außen hin sichtbar ist. [FP11, S.159] In Anlehnung an Ghosh ist das semantische Model mit dem AST gleichzusetzen, der durch eine laxikalische Analyse des DSL-Scripts mithilfe eines Parsers erzeugt wird. Somit wird es als Datenstruktur betrachtet, dessen Struktur von der Syntax der DSL unabhängig ist [gho11, S.214]. Das Gleichsetzen des semantischen Models mit dem AST ist laut Martin Folwer in den meisten Fällen nicht effektiv. Grund dafür ist, dass der AST sehr stark an die Syntax der DSL gebunden, wohingegen das semantische Model von der Syntax unabhängig ist. [FP11, S.48]

---

<sup>5</sup>Siehe Glossar: Abstrakter Syntax Baum

### **Generator**

In Anlehnung an Martin Fowler ist ein Generator ist ein Teil der einer DSL Umgebung<sup>6</sup>, der für das Erzeugen von Quellcode für die Zielumgebung<sup>7</sup> zuständig ist. [FP11, S.121] Bei der Generierung von Code wird zwischen zwei Arten unterschieden.

#### **Transformer Generation**

Bei der Transformer Generation wird das semantische Model als Input verwendet, woraus Quellcode für die Zielumgebung generiert wird. [FP11, S.533f] Eine solte Generation wird oft verwendet, wenn ein Großteil des Output generiert wird und die Inhalte des semantischen Models einfach in den Quellcode der Zielumgebung überführt werden können. [FP11, S.535]

#### **Templated Generation**

Bei der Templated Generation wird eine Vorlage benötigt. In dieser Vorlage befinden sich Platzhalter, an deren Stelle der Generator speziellen Code generiert. [FP11, S.539f] Diese Art der Cod degenerierung wird oft verwendet, wenn sich in der generierte Quellcode für die Zielumgebung viele statische Inhalte befinden und der dynamisch generierte Anteil sehr einfach gehalten ist. [FP11, S.541]

---

<sup>6</sup>Siehe Glossar: DSL Umgebung

<sup>7</sup>Siehe Glossar: Zielumgebung

## 3.2 Anwendungsbeispiele

Die Anwendungsbereiche für DSLs sind sehr unterschiedlich. Die bekanntesten DSL sind Sprachen wie *SQL* (zur Abfrage und Manipulation von Daten in einer relationalen Datenbank), *HTML* (als Markup-Sprache für das Web) oder *CSS* (als Layoutbeschreibung). [gho11, S.12] Alle Sprachen besitzen eine eingeschränkte Ausdrucksmöglichkeiten und sind von der Abstraktion her direkt auf eine Domäne (jeweils dahinter in Klammern genannt) zugeschnitten. [gho11, S.12f]

Weitere Beispiele für DSL befinden sich im Bereich der Sprachen für Parser-Generatoren (*YACC*, *ANTLR*) oder im Bereich der Sprachen für das Zusammenbauen von Softwaresystemen (*Ant*, *Make*). [gho11, S.12]

Für den Bereich der UI-Entwicklung gibt es ebenfalls Anwendungsbeispiele. Diese werden in Kapitel 5 genauer beleuchtet.

## 3.3 Model-Driven Software Development (MDSD)

In der Einleitung wurde schon der Model-Driven Ansatz in Verbindung mit UI-Entwicklung erwähnt. Dieser Ansatz versucht den technischen Lösungen der Industrie einen gewissen Grad an Agilität zu verleihen. [SKNH05] Das ist damit verbunden, dass die Produktion von Softwareprodukten schneller und besser von statten geht und mit weniger Kosten verbunden ist. [DM14, S.71] Erreicht wird dies indem die Modelle formaler, strenger, vollständiger und konsistenter beschrieben werden. [VBK<sup>+</sup>13, S.31] Die Kernidee ist, dass die Modelle Quellcode oder Funktionalitäten beschreiben und diese in der Evolution der Software immer wiederverwendet werden können. [DM14, S.72] Somit wird wiederkehrender oder schematischer Quellcode vermieden und es ist möglich diese Modelle auch in anderen Anwendungen zu verwenden. [DM14, S.71] Daraus lassen sich folgende Ziele des MDSD ableiten:

## **3.4 Abgrenzung zu GPL**

## **3.5 Vor- und Nachteile von DSL gegenüber GPL**

## **3.6 Interne DSL**

### **3.6.1 Implementierungstechniken**

## **3.7 Externe DSL**

### **3.7.1 Implementierungstechniken**

#### **3.7.2 Parser**

## **3.8 Textuelle vs. Nicht-Textuelle DSL**

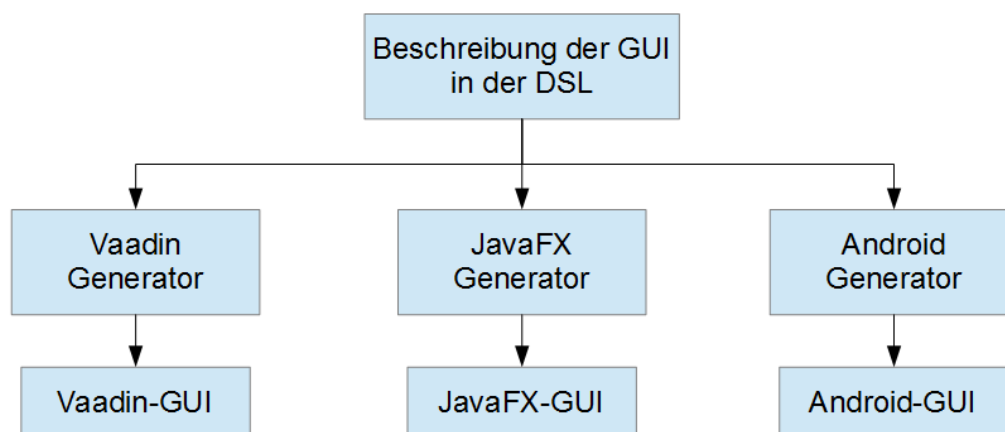


Abbildung 4.1: DSL-Ansatz für gleich GUIs auf unterschiedlichen Plattformen

## Kapitel 4

# Entwicklung einer Lösungsidee

### 4.1 Allgemeine Beschreibung der Lösungsidee

Aufgrund der nur schwer machbaren Integration neuer Frameworks in das bestehende Multichannel-Framework und der Tatsache, dass die derzeit genutzten Frameworks (Swing und wingS) veraltet sind, wird ein neuer Ansatz für die Umsetzung von GUIs auf unterschiedlichen Plattformen gesucht.

Der neue Ansatz basiert auf der folgenden Idee. Die GUIs sollen weiterhin nur einmal beschrieben werden sollen. Diese Beschreibung soll über eine DSL erfolgen und sich nicht an bestehende Frameworks orientieren. Grund dafür ist, dass ansonsten die Gefahr besteht, dass langfristig betrachtet mit diesem Ansatz das gleiche Problem auftritt wie beim Multichannel-

Framework. Aus der Beschreibung der GUIs wird ein Generator speziellen Quellcode erzeugen, der sich auf entsprechenden Plattformen ausführen lässt. Für jedes eingesetzte Framework muss somit ein eigener Generator entwickelt werden. Abbildung 4.1 bildet die aus dieser Idee resultierende Architektur ab<sup>1</sup>.

## 4.2 Architektur

## 4.3 Vorteile gegenüber dem Multichannel-Framework

---

<sup>1</sup>Hier: Vaadin als Web-Framework, JavaFX als Framework für den Standalone-Client und Android als Repräsentant für einen möglichen Mobile-Client

# Kapitel 5

## GUI-DSL

### 5.1 Beschreibung der Anforderung an die GUI

### 5.2 Vorstellung ausgewählter DSL zur Beschreibung von GUIs

#### 5.2.1 The Snow

#### 5.2.2 glc-dsl

#### 5.2.3 Sculptor

### 5.3 Bewertung



## **Kapitel 6**

# **Evaluation des Frameworks zur Entwicklung der DSL**

### **6.1 Vorstellung ausgewählter Frameworks**

#### **6.1.1 PetitParser**

#### **6.1.2 Xtext**

#### **6.1.3 MPS**

### **6.2 Vergleich und Bewertung der vorgestellten Frameworks**

## **Kapitel 7**

# **Aufteilung der Anforderungen auf Sprache und Generator**

### **7.1 Anforderung an die neue DSL**

### **7.2 Anforderung an den Generators**

## Kapitel 8

# Entwicklung einer DSL zur Beschreibung der GUI in profil c/s

### 8.1 Analyse der Metadaten der GUI

### 8.2 Syntax

### 8.3 Semantik

## **Kapitel 9**

# **Entwicklung des Generators für das Generieren von Klassen für das Multichannel-Framework**

### **9.1 WAM-GUI Architektur**

### **9.2 Syntax und Semantik für die Beschreibung der GUIs**

### **9.3 Umsetzung des frameworkspezifischen Gene- rators**

## **Kapitel 10**

### **Zusammenfassung und Ausblick**

## Titel anhang a

# Glossar

**Förderantrag** [...] ist ein Antrag, den der Begünstigte einreicht, wenn er sich eine Maßnahme fördern lassen möchte [dat14]. 5

**GridBagLayout** ist ein Layout Manager innerhalb von Swing, welcher die Komponenten horizontal, vertical und entlang der Grundlinie anordnet. Dabei müssen die Komponenten nicht die gleiche Größe haben [Oraa]. 9

**GUI** ist die Schnittstelle zwischen dem Benutzer und dem Programm. 1

**InVeKoS** ist die Abkürzung für Integriertes Verwaltungs- und Kontrollsystem. Mit einem solchen Sysmten wird im allgemeinen sichergestellt, dass die durch den Europäischen Garantiefonds für die Landwirtschaft finanzierten Maßnahmen ordnungsgemäß umgesetzt wurden. Im speziellen bedeutet dies die Absicherung, von Zahlungen, die korrekte behandlung von Unregelmäßigkeiten und das wieder Einziehen von zu unrecht gezahlter Beiträge [Gen14]. 5

**Swing** ist ein UI-Framework für Java Applikationen [Orab]. xiii, 8, 9

**Traditionelle UI-Entwicklung** Bei der traditionellen UI-Entwicklung wird mit traditionellen UI-Toolkits gearbeitet. Bei diesen Toolkits wird Aufbau der GUI genau beschrieben. Für die Interaktion mit den UI-Widgets, werden Listener implementiert, die auf andere Events reagieren, die von anderen Widgets erzeugt generiert wurden. Events können zu unterschiedlichen Zeitpunkten generiert werden und es wird nicht festgelegt in welcher Reihenfolge sie bei anderen Widgest ankommen. [KB11]. 1

**Usability** beschreibt die Nutzerfreundlichkeit einer GUI, sowie auch die Nutzerfreundlichkeit einer Software. 1

. 8, 9

**Zuwendungs-Berechner** ist ein Werkzeug innerhalb von profil c/s. *Mit diesem Werkzeug kann der Sachbearbeiter die Zuwendung, die dem Antragsteller bewilligt werden soll, nach einem standardisierten Verfahren berechnen (siehe Abschnitt "Algorithmen"). Das Ergebnis wird im Zuwendungsblatt dokumentiert, das auch später mit demselben Werkzeug angesehen werden kann* [deG07]. xiv

**Zuwendungsblatt** ist die grafische Dokumentation der Ergebnisse des Zuwendungs-Berechners innerhalb von profil c/s. xiv, 5



# Literaturverzeichnis

- [Aho08] AHO, ALFRED V: *Compiler: Prinzipien, Techniken und Werkzeuge*. Pearson Studium, 2008.
- [dat14] DATA EXPERTS GMBH: *Förderantrag*. Profil Wiki der deg, März 2014. Zuletzt eingesehen am 02.12.2014.
- [deG07] GMBH DATA EXPERTS: *Detaillkonzept ELER/i-Antragsmappe*, Januar 2007. Letzte Änderung am 01.12.2014.
- [DM14] DANIEL, FLORIAN und MARISTELLA MATERA: *Model-Driven Software Development*. In: *Mashups, Data-Centric Systems and Applications*, Seiten 71–93. Springer Berlin Heidelberg, 2014.
- [FP11] FOWLER, MARTIN und REBECCA PARSON: *Domain-Specific Languages*. Addison-Wesley, 2011.
- [Gen14] GENERALDIREKTION LANDWIRTSCHAFT UND LÄNDLICHE ENTWICKLUNG: *Das Integrierte Verwaltungs- und Kontrollsystem (InVeKoS)*. URL: [http://ec.europa.eu/agriculture/direct-support/iacs/index\\_de.htm](http://ec.europa.eu/agriculture/direct-support/iacs/index_de.htm), November 2014. Zuletzt eingesehen am 02.12.2014.
- [gho11] *DSLs in Action*. Manning Publications Co., 2011.
- [Gun14] GUNDERMANN, NIELS: *Entwicklung einer Grammatik für eine DSL mit xText am Beispiel einer Sprache zur Definition von Pflichtprüfungen in profil c/s*, 2014. Praxisbericht.
- [Hed12] HEDTSTUECK, ULRICH: *Einführung in die Theoretische Informatik*, Band 5. Auflage. Oldenbourg Verlag, 2012.

- [KB11] KRISHNASWAMI, NEELAKANTAN R. und NICK BENTON: *A Semantic Model for Graphical User Interfaces*. Microsoft Research, September 2011. Verfügbar unter URL:.
- [LW] LU, XUDONG und JIANCHENG WAN: *Model Driven Development of Complex User Interface*. Technischer Bericht, Shandong University. Verfügbar unter URL: <http://ceur-ws.org/Vol-297/paper7.pdf>.
- [Maa07] MAASS, DIRK: *JWAMMC - Das Multichannel-Framework der data-experts gmbh*. Vortrag, Dezember 2007.
- [MHP99] MYERS, BRAD, SCOTT E. HUDSON und RANDY PAUSCH: *Past, Present and Future of User Interface Software Tools*. Technischer Bericht, Carnegie Mellon University, September 1999. Verfügbar unter URL: <http://www.cs.cmu.edu/~amulet/papers/futureof-hci.pdf>.
- [Oraa] ORACLE: *Class GridBagLayout*. URL: <https://docs.oracle.com/javase/7/docs/api/java/awt/GridBagLayout.html>. Zuletzt eingesehen am 02.12.2014.
- [Orab] ORACLE: *Swing*. URL: <https://docs.oracle.com/javase/jp/8/technotes/guides/swing/index.html>. Zuletzt eingesehen am 02.12.2014.
- [SKNH05] SUKAVIRIYA, NOI, SANTHOSH KUMARAN, PRABIR NANDI und TERRY HEATH: *Integrate Model-driven UI with Business Transformations: Shifting Focus of Model-driven UI*. Technischer Bericht, IBM T.J. Watson Research Center, Oktober 2005. Verfügbar unter URL: <http://www.research.ibm.com/people/p/prabir/MDDAUI.pdf>.
- [Use12] USERLUTIONS GMBH: *3 Gründe, warum gute Usability wichtig ist*. URL: <http://rapidusertests.com/blog/2012/04/3-gute-grunde-fuer-usability-tests/>, April 2012. Zuletzt eingesehen am 01.12.2014.

- [VBK<sup>+</sup>13] VÖLTER, MARKUS, SEBASTIAN BENZ, LENNART KATS, MATS  
HELANDER, EELCO VISSER und GUIDO WACHSMUTH: *DSL En-  
gineering*. CreateSpace Independent Publishing Platform, 2013.