

*Sehr gut. → Vektordarstellung!*

Duale Hochschule Baden-Württemberg  
Mannheim

## Projektarbeit 1

**Konzeptionelle Entwicklung eines Lokalisierungstools für Movelets**

### Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Verfasser:	Fabian Wolf
Matrikelnummer:	7345461
Firma:	Honeywell International Inc.
Abteilung:	Research & Development
Kurs:	WWI17SEC
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Wissenschaftlicher Betreuer:	Prof. Dr.-Ing. habil. Dennis Pfisterer dennis.pfisterer@dhbw-mannheim.de +49 62 14105-1253
Firmenbetreuer:	Herr Oliver Erlenkaemper Oliver.Erlenkaemper@Honeywell.com +49 62 18545-0237
Bearbeitungszeitraum:	07.08.2018 – 16.11.2018

## Kurzfassung

Softwareprodukte werden heutzutage weltweit vermarktet. Infolgedessen müssen Unternehmen ihre Softwareprodukte effizient lokalisieren, falls sie sich erfolgreich auf dem Weltmarkt behaupten wollen.<sup>12</sup> Aus diesem Grund ist Software zur Automatisierung der Lokalisierung nötig. Derzeit wird von Movilizer keine Software zur Lokalisierung von Movelets eingesetzt. Ziel dieser Arbeit ist die Konzeption einer Software, welche zu übersetzende und übersetzte Strings der Texte eines Movelets automatisiert verarbeitet. Diese Software wird im Folgenden als Lokalisierungstool bezeichnet. Anhand von Literaturrecherche werden verschiedene Konzepte analysiert und diskutiert. Das resultierende Lokalisierungstool extrahiert zu übersetzende Strings in XLIFF-Dateien anhand von ITS und einer \_() Auszeichnung. Die extrahierten Strings erhalten eine Übersetzung, welche von dem Lokalisierungstool genutzt wird um die ausgezeichneten Strings zu ersetzen. Resultierend daraus ist dieses Lokalisierungstool der Anfang der Automatisierung der Lokalisierung von Movelets.

wie gut ist das Ergebnis?

Sehr kurz!

<sup>1</sup>Vgl. Schmitz und Wahle 2000, S. 1.

<sup>2</sup>Vgl. Reineke und Schmitz 2005, S. 1.

# Inhaltsverzeichnis

<b>Abbildungsverzeichnis</b>	<b>iii</b>
<b>Quelltextverzeichnis</b>	<b>iv</b>
<b>Abkürzungsverzeichnis</b>	<b>v</b>
<b>1 Einleitung</b>	<b>1</b>
1.1 Allgemeine Themenrelevanz . . . . .	1
1.2 Ziel dieser Projektarbeit . . . . .	1
<b>2 Begriffe</b> <i>Grundlagen</i>	<b>3</b>
2.1 Globalisierung . . . . .	3
2.2 Internationalisierung . . . . .	3
2.3 Lokalisierung . . . . .	3
2.4 Abgrenzung von Übersetzung, Lokalisierung und Internationalisierung .	4
2.5 Translation Memory . . . . .	4
2.6 Computer Assisted Translation . . . . .	4
2.7 Natural Language Processing . . . . .	5
2.8 Auszeichnung . . . . .	5
2.9 Extensible Markup Language . . . . .	5
2.10 Movilizer . . . . .	5
2.11 Movelet . . . . .	6
2.12 Stammdaten . . . . .	6
2.13 Movilizer Extensible Markup Language . . . . .	6
2.14 Movilizer Expression Language . . . . .	6
2.15 Movilizer Gradle Plug-in . . . . .	7
2.16 Abfragesprache . . . . .	7
<b>3 Verwandte Arbeit</b>	<b>8</b>
3.1 Trennung von Quelltext und Ressourcen . . . . .	8
3.2 Funktionsweisen bestehender Lokalisierungstools . . . . .	8
3.3 Standards der Lokalisierung . . . . .	10
3.3.1 TMX . . . . .	10
3.3.2 XML:TM . . . . .	10
3.3.3 SRX . . . . .	11
3.3.4 GMX-V . . . . .	11
3.3.5 TBX . . . . .	11
3.3.6 XLIFF . . . . .	12

*muß das hier rein?*

3.3.7	Trans-WS . . . . .	12
3.3.8	UTX . . . . .	12
3.3.9	OLIF . . . . .	12
3.3.10	ITS . . . . .	13
<b>4</b>	<b>Anforderungen an das Lokalisierungstool</b>	<b>14</b>
<b>5</b>	<b>Extrahieren zu übersetzender Strings aus dem Quelltext</b>	<b>16</b>
5.1	Auszeichnung zu übersetzender Strings . . . . .	16
5.2	Automatisches Erkennen zu übersetzender Strings . . . . .	18
5.3	Diskussion der Lösungsansätze . . . . .	19
<b>6</b>	<b>Schnittstelle des Lokalisierungstools zu anderer Software</b>	<b>21</b>
<b>7</b>	<b>Ersetzen zu übersetzender durch übersetzte Strings</b>	<b>23</b>
7.1	Identifikation des zu ersetzenden Strings per Name . . . . .	23
7.2	Identifikation des zu ersetzenden Strings per Zeiger . . . . .	24
7.3	Behandlung fehlender Übersetzungen . . . . .	25
7.4	Diskussion der Lösungsansätze . . . . .	25
<b>8</b>	<b>Ansatzzeitpunkt des Lokalisierungstools</b>	<b>27</b>
<b>9</b>	<b>Versionsverwaltung der Lokalisierung</b>	<b>28</b>
<b>10</b>	<b>Fazit</b>	<b>29</b>
<b>11</b>	<b>Diskussion</b>	<b>31</b>
	<b>Literaturverzeichnis</b>	<b>33</b>
<b>A</b>	<b>Anhang</b>	<b>38</b>
A.1	Implementierung der Extraktion zu übersetzender Strings anhand einer Auszeichnung . . . . .	38

# Abbildungsverzeichnis

Abbildung 10.1 Darstellung der Arbeitsschritte des Lokalisierungstools . . . . .	30
--	----

# Quelltextverzeichnis

3.1	Ausgezeichneter Quelltext . . . . .	9
3.2	Erstelltes .po . . . . .	9
3.3	Übersetztes .po . . . . .	9
4.1	MXML-Elemente und -Attribut inklusive Stammdaten . . . . .	15
4.2	MEL-Stringliterale . . . . .	15
4.3	Werte in Movilizer Gradle Plug-in Ressource Dateien . . . . .	15
5.1	Ausgezeichneter zusammengesetzter String . . . . .	17
5.2	Extrahierter zusammengesetzter String . . . . .	18
5.3	Message Screen . . . . .	18
5.4	Epsilon Screen . . . . .	19
7.1	Generischer Name mit Affix . . . . .	24
7.2	Zuordnung von Name zu übersetzten String . . . . .	24

# Abkürzungsverzeichnis

<b>MEL</b>	Movilizer Expression Language
<b>MXML</b>	Movilizer Extensible Markup Language
<b>XML</b>	Extensible Markup Language
<b>XPATH</b>	XML Path Language
<b>.po</b>	Portable Object
<b>.mo</b>	Machine Object
<b>TMX</b>	Translation Memory Exchange
<b>XML:TM</b>	XML Text Memory
<b>XLIFF</b>	XML Localisation Interchange File Format
<b>SRX</b>	Segmentation Rules Exchange
<b>GMX-V</b>	Global Information Management Metrics Volume
<b>UTX</b>	Universal Terminology Exchange
<b>TBX</b>	Term Base Exchange
<b>OLIF</b>	Open Lexicon Interchange Format
<b>Trans-WS</b>	Translation Web Services
<b>ITS</b>	Internationalization Tag Set
<b>GALA</b>	Globalization and Localization Association
<b>LISA</b>	Localization Industry Standards Association
<b>OASIS</b>	Organization for the Advancement of Structured Information Standards
<b>AAMT</b>	Asia-Pacific Association for Machine Translation
<b>W3C</b>	World Wide Web Consortium
<b>NLP</b>	Natural Language Processing
<b>CAT</b>	Computer Assisted Translation
<b>TM</b>	Translation Memory

# 1 Einleitung

↳ Kilot

## 1.1 Allgemeine Themenrelevanz

Softwareprodukte werden heutzutage weltweit vermarktet. In der Konsequenz müssen Unternehmen ihre Softwareprodukte an die Anforderungen verschiedener regionaler Märkte anpassen, falls sie sich erfolgreich auf dem Weltmarkt behaupten wollen. Aufgrund dieser Entwicklung zählt Lokalisierung heute zu den Schlüsselbegriffen einer globalen Marktwirtschaft.<sup>34</sup> Lokalisierung alleine reicht jedoch nicht, um sich gegen Konkurrenten durchzusetzen. Faktoren wie Qualität, Dauer und Kosten sind von großer Bedeutung. Daher ist die Automatisierung von Unterprozessen der Lokalisierung durch Software erforderlich.<sup>5</sup> Derzeit wird von Movilizer keine Software zur Lokalisierung von Movelets eingesetzt.

↳ Def.

was ist das?

## 1.2 Ziel dieser Projektarbeit

Ziel dieser Arbeit ist die Konzeption einer Software, welche zu übersetzende und übersetzte Strings der Texte eines Movelets automatisiert verarbeitet. Resultierend daraus verringert die Software den Aufwand des Prozesses der Lokalisierung. Aufgrund dessen, dass die Software einen Unterprozess der Lokalisierung automatisiert, wird sie im Folgenden als Lokalisierungstool bezeichnet.

Weitere Unterprozesse der Lokalisierung werden von diesem Lokalisierungstool nicht umgesetzt. Hierzu zählen:

↳ woher kommt das?

- Übersetzung
- Verwendung geeigneter Zeichensätze
- Übersetzung verknüpfter Strings
- Anpassung von Grafiken und Symbolen
- Anpassen der Benutzeroberfläche an Textlängen und Flussrichtungen
- Anpassung an geltendes Recht
- Anpassung der Formate für Adresse, Datum <sup>etc.</sup> et cetera

<sup>3</sup>Vgl. Schmitz und Wahle 2000, S. 1.

<sup>4</sup>Vgl. Reineke und Schmitz 2005, S. 1.

<sup>5</sup>Vgl. Zydrón und Saldana Derek 2009, S. 1.



- Anpassung der Maßeinheiten für Gewicht, Währung, et cetera
- Anpassung der Sortierung und Suche

678

Im Verlauf dieser Arbeit werden zunächst die sich aus dem Ziel ergebenden Anforderungen formuliert. Des Weiteren, werden die möglichen Lösungsansätze zu den einzelnen Anforderungen formuliert und unter Berücksichtigung vorausgegangener Erkenntnisse diskutiert. Die Lösungsansätze ergeben sich anhand von Literaturrecherche. Sollten sich aus der Diskussion neue Anforderungen ergeben, wird dieses Verfahren iterativ angewandt. Aus der Gesamtheit aller erarbeiteten Lösungen ergibt sich das Konzept des Lokalisierungstools.

*Außer der Arbeit?*

*In Kap 1 ...*

*#*

---

<sup>6</sup>Vgl. Hassell-Corbiell 2001, S. 423.

<sup>7</sup>Vgl. Asnes 2010.

<sup>8</sup>Vgl. Wagner 2017.

## 2 Begriffe



### 2.1 Globalisierung

Als Globalisierung bezeichnet man alle Aktivitäten eines Unternehmens mit dem Ziel der Vermarktung eines (Software-)Produkts außerhalb des nationalen, lokalen Marktes. Zu diesem Zweck sind technische, wirtschaftliche und gesetzliche Aspekte des Zielmarktes besonders zu berücksichtigen. Infolgedessen ist die Globalisierung im Kontext der betriebswirtschaftlichen und kaufmännischen Unternehmensführung zu betrachten.<sup>9 10</sup>

### 2.2 Internationalisierung

Als Internationalisierung bezeichnet man die Entwicklung von (Software-)Produkten auf eine Weise, die eine möglichst schnelle Lokalisierung mit geringen Aufwand ermöglicht. Zu diesem Zweck muss das (Software-)Produkt mit einer Funktionalität entwickelt werden, die eine Anpassung an technische Konventionen, kulturelle Eigenheiten und Sprache des Zielmarktes ermöglichen. Infolgedessen ist die Internationalisierung immer im Kontext der Entwicklung zu betrachten.<sup>11 12</sup>

### 2.3 Lokalisierung

Als Lokalisierung bezeichnet man die eigentliche Anpassung von (Software-)Produkten an technische Konventionen, kulturelle Eigenheiten und Sprache des Zielmarktes. Zu diesem Zweck müssen (Software-)Produkte zunächst internationalisiert werden, um die Lokalisierung zu ermöglichen.<sup>13 14</sup> In der Konsequenz ist der Aufwand der Lokalisierung geringer, je mehr Aufwand in die Internationalisierung investiert wird.<sup>15</sup>

---

<sup>9</sup>Vgl. Reineke und Schmitz 2005, S. 1.

<sup>10</sup>Vgl. Schmitz und Wahle 2000, S. 2.

<sup>11</sup>Vgl. Reineke und Schmitz 2005, S. 2.

<sup>12</sup>Vgl. Schmitz und Wahle 2000, S. 2.

<sup>13</sup>Vgl. Reineke und Schmitz 2005, S. 2.

<sup>14</sup>Vgl. Schmitz und Wahle 2000, S. 3.

<sup>15</sup>Vgl. Reineke und Schmitz 2005, S. 2.

## 2.4 Abgrenzung von Übersetzung, Lokalisierung und Internationalisierung

Ist ein Text in einer Ausgangssprache verfasst, so wird das Verfassen dieses Texts in einer Zielsprache als Übersetzen bezeichnet. In Abschnitt 2.3 ist die Anpassung von Sprache an einen Zielmarkt als Teil der Lokalisierung definiert. Des Weiteren ist in Abschnitt 2.2 das Entwickeln von Software mit Funktionalität zur einfachen Anpassung an Sprache als Teil der Internationalisierung definiert. Die Anpassung von Sprache beinhaltet unter anderem das Übersetzen von Texten. Daher ist das Übersetzen ein Teil der Lokalisierung. Daraus leitet sich wiederum ab, dass das Übersetzen ein Teil der Lokalisierung und der Internationalisierung ist.

## 2.5 Translation Memory

Translation Memory (TM) ist eine Datenbank voneinander zugehörigen Textpaaren. Die Texte des Textpaares sind in unterschiedlichen Sprachen vorhanden, der Ausgangssprache und der Zielsprache. Das Textpaar besteht aus einem Text in der Ausgangssprache und der Übersetzung dieses Texts in der Zielsprache. Folglich sind die Übersetzungen zwecks Wiederverwendung gespeichert. Diese Wiederverwendung ermöglicht eine Steigerung der Übersetzungsgeschwindigkeit und der Kosteneffizienz, besonders bei repetitiven Texten.<sup>16</sup>

## 2.6 Computer Assisted Translation

Der Einsatz von Software zum Zweck der automatischen Übersetzung erzielt keine hochqualitativen Übersetzungsergebnisse. Eine Möglichkeit, die Qualität automatisch übersetzter Texte zu erhöhen, ist die nachträgliche Verbesserung durch Übersetzer. Bei diesem Prozess kann die eingesetzte Software jedoch nicht von dem Wissen der Übersetzer profitieren. Aus diesem Grund ist es sinnvoll, Übersetzer in der Zusammenarbeit mit Software während des Übersetzungsprozesses einzusetzen. Dies wird als Computer Assisted Translation (CAT) bezeichnet. Auf diese Weise werden lexikalische, syntaktische und semantische Mehrdeutigkeiten in der Interaktion von Mensch und Software gelöst. Resultierend daraus steigen Übersetzungsqualität und Übersetzungsgeschwindigkeit.<sup>17</sup>

---

<sup>16</sup>Vgl. O'Brien, O'Hagan und Flanagan 2010, S. 187.

<sup>17</sup>Vgl. Barrachina et al. 2009, S. 4.

## 2.7 Natural Language Processing

Natural Language Processing (NLP) ist ein interdisziplinäres Feld, welches Algorithmen und Systeme zum Verstehen und Verarbeiten natürlicher Sprache durch Computer entwickelt und erforscht. Aufgabe von NLP ist es menschliche Sprache in gesprochener und geschriebener Form zu analysieren und infolgedessen Kommandos und nützliche Informationen aus dieser zu extrahieren.<sup>18</sup>

## 2.8 Auszeichnung

Textverarbeitungssysteme benötigen typischerweise zusätzliche Information innerhalb des zu verarbeitenden Dokuments. Diese zusätzliche Information wird in natürlichen Text eingebettet und als Auszeichnung <sup>19</sup> bezeichnet. Diese Auszeichnung dient dem Zweck der Trennung der logischen Elemente des Dokuments von den Regeln zur Verarbeitung der jeweiligen Elemente.<sup>20</sup>

## 2.9 Extensible Markup Language

Extensible Markup Language (XML) ist eine Auszeichnungssprache, welche Daten als Inhalte von Elementen mit Tags und Attributen auszeichnet. Diese Elemente sind hierarchisch in einer Baumstruktur geordnet und für Menschen lesbar. Der Aufwand der Implementierung einer XML-erstellenden und/oder -verarbeitenden Software soll gering sein. Des Weiteren soll XML direkt über das Internet genutzt werden können. Aufgrund dessen ermöglicht XML das plattform-unabhängige, einfache Speichern und Austauschen von Daten für Software.<sup>21</sup>

## 2.10 Movilizer

Movilizer ist der Server für den operativen Arbeitsbereich. Movilizer bietet die zentralisierte Vernetzung aller mobilen Geräte und Geschäftsprozessen eines Unternehmens. Zusätzlich ist Movilizer der Name des Unternehmens, welches Movilizer entwickelt und vertreibt.<sup>22</sup>

---

<sup>18</sup>Vgl. Sintoris und Vergidis 2017, S. 135.

<sup>19</sup>Im Englischen Markup

<sup>20</sup>Vgl. Ellison 1994, S. 1.

<sup>21</sup>Vgl. Bray et al. 2008.

<sup>22</sup>Vgl. Nitschkowski 2018c.

## 2.11 Movelet

Das Movelet ist eine zentrale Komponente von Movilizer, welches einen Geschäftsprozess innerhalb eines Geschäftsszenarios abbildet. Das Movelet beinhaltet alle Informationen, welche zur Ausführung dieses Geschäftsprozesses auf einem mobilen Gerät nötig sind. Movelets können kombiniert werden um komplexe und umfassende Geschäftsprozesse auf einer einfach zu nutzenden mobilen Anwendung abzubilden. Von einem technischen Standpunkt sind Movelets eine universelle Basiseinheit, welche auf einer XML-Struktur basieren.<sup>23</sup>

## 2.12 Stammdaten

Stammdaten sind Datenpakete, welche zum Datentransfer zwischen Datenbanken, Movilizer Servern und Movelets verwendet werden. Stammdaten können von Movelets verarbeitet und angezeigt werden. Zu diesem Zweck enthalten Stammdaten Texte, Nummern oder Binärdateien, wie beispielsweise Bilder. Diese Inhalte sind in einer hierarchischen XML-Struktur organisiert.<sup>24</sup>

## 2.13 Movilizer Extensible Markup Language

Movilizer Extensible Markup Language (MXML) ist eine XML-basierte Auszeichnungssprache, welche alle XML-Element und -Attribute im Kontext von Movilizer beschreibt.<sup>25</sup>

## 2.14 Movilizer Expression Language

Movilizer Expression Language (MEL) ist eine auf Ereignissen basierende Programmiersprache, welche in die MXML-Struktur eines Movelets integriert ist. In der Konsequenz ermöglicht MEL das Nutzen von Ereignissen zur Verarbeitung von Benutzereingaben, manipulieren der Benutzeroberfläche und weiteren dynamischen Elementen innerhalb von Movelets.<sup>26</sup>

---

<sup>23</sup>Vgl. Nitschkowski 2018c.

<sup>24</sup>Nitschkowski.2018d.

<sup>25</sup>Vgl. Nitschkowski 2015b.

<sup>26</sup>Vgl. Nitschkowski 2018d.

## 2.15 Movilizer Gradle Plug-in

Def.?

Das Movilizer Gradle Plug-in ist ein Gradle Plug-in mit der Aufgabe Movelet Projekte von der Entwicklung bis zur Veröffentlichung zu unterstützen. Das Movilizer Gradle Plug-in bietet zwei Prozesse, den *Compile Request* und den *Send Request*. Der *Compile Request* ruft alle in einem Movelet referenzierten Dateien ab und erstellt anhand dieser eine Anfrage für die Movilizer Server. Der *Send Request* sendet die von dem *Compile Request* erstellten Anfragen mit den benötigten Einstellungen an die Movilizer Server. Des Weiteren ermöglicht das Movilizer Gradle Plug-in das Nutzen einer Software zur Verarbeitung von Vorlagen. Diese Software ermöglicht unter anderem das Verwenden von Variablensubstitutionen, Kommentaren, konditionalen Blöcken, Schleifen und Importen von externen Quelltexten innerhalb der Vorlage.<sup>27</sup>

## 2.16 Abfragesprache

Eine Abfragesprache ist eine Sprache, in welcher eine Abfrage an ein Informationssystem gestellt werden kann. Ziel dieser Abfrage ist es gesuchte Informationen zu erhalten.<sup>28</sup> Eine Abfragesprache besteht aus den getrennten Definitionen ihrer Syntax und Semantik. Die Syntax definiert das für die Abfrage zur Verfügung stehende Vokabular und seiner erlaubten Kombinationen. Die Semantik definiert die Interpretationsvorschriften einer Abfrage und damit, welches Ergebnis diese zurückliefert.<sup>29</sup>

<sup>27</sup>Vgl. De Mula und Paul Janzen 2018.

<sup>28</sup>Vgl. Reiner 1991, S7f.

<sup>29</sup>Vgl. Willenborg 2001, S. 24.

## 3 Verwandte Arbeit

### 3.1 Trennung von Quelltext und Ressourcen

Eine der ersten Formen der Internationalisierung von Software, ist die strikte Trennung des Quelltexts von den Ressourcen. Hierbei müssen alle Strings in Ressource Dateien getrennt vom Quelltext gespeichert werden. Die Trennung muss also bereits beim Beginn der Softwareentwicklung erfolgen. Dies ermöglicht die Lokalisierung der Software durch Austauschen der Ressource Dateien mit andere Ressource Dateien, die eine andere Sprache enthalten. Die Ressource Dateien bestehen aus Schlüssel-Wert-Paaren. Die Schlüssel enthalten die Bezeichner, welche die Stellen im Quelltext referenzieren, an denen die zugehörigen Werte eingefügt werden. Die Werte enthalten die Strings in der Zielsprache. Um die Ressource Dateien zu übersetzen, müssen also lediglich die Werte übersetzt werden. Die Bezeichner bleiben gleich. Das Übersetzen geschieht in der Regel nicht durch Ändern der originalen Ressource Dateien. Stattdessen werden Kopien erzeugt, welche lediglich den Schlüssel enthalten. In diesen neuen Ressource Dateien werden die Strings in der Zielsprache als Wert eingefügt. Beim Kompilieren des Quelltexts werden die Bezeichner durch die zugehörigen Strings ersetzt. Resultieren daraus wird eine übersetzte Version der Software erzeugen.<sup>30</sup>

Diese Form der Internationalisierung ist in dem Kontext von Movelets möglich. Alle String Ressourcen eines in dieser Form internationalisierten Movelets sind in Stammdaten ausgelagert. Die Stammdaten beinhalten die Schlüssel-Wert-Paare, bestehend aus Strings und deren zugehörige Übersetzungen in den benötigten Zielsprachen. Wie bereits erwähnt, muss jedoch diese Trennung von Beginn der Softwareentwicklung berücksichtigt werden, während ein Lokalisierungstool auch nach fortgeschrittener Programmierung des Movelets ohne große Quelltextänderungen eingesetzt werden kann. Dies wird im folgenden Abschnitt erläutert.

### 3.2 Funktionsweisen bestehender Lokalisierungstools

Seit einigen Jahren existieren Lokalisierungstools, welche keine strikte Trennung von Quellcode und Ressourcen benötigen. Diese Lokalisierungstools verwenden stattdessen eine Auszeichnung, um zu übersetzende Strings im Quellcode zu markieren. Im Zuge der Lokalisierung wird den ausgezeichneten Strings eine Übersetzung zugewiesen. Diese Funktionsweise wird in diesem Abschnitt beispielhaft anhand von *GNU gettext*

---

<sup>30</sup>Vgl. Reineke und Schmitz 2005, S. 145.

dargestellt. GNU *gettext* nutzt `_()` oder *gettext()*, um zu übersetzende Strings im Quelltext auszuzeichnen, siehe 3.1. Die Auszeichnung wird von einem Parser verwendet, welcher durch den Kommandozeilenbefehl *xgettext* gestartet wird. Dieser Parser erzeugt ein *Portable Object (.po)*, welches Schlüssel-Wert-Paare enthält. Dem Schlüssel *msgid* ist der Wert des im Quelltext ausgezeichneten Strings zugewiesen. Dieser String kann der zu übersetzende Text der Ausgangssprache sein oder ein beliebiger Bezeichner, siehe 3.2. Ein Übersetzer fügt die jeweiligen Übersetzungen in das *.po* ein, dies geschieht mithilfe eines beliebigen Texteditors oder einer mit *.po*-kompatiblen Software. Hierbei wird dem Schlüssel *msgstr* der zugehörige Text der jeweiligen Zielsprache als Wert zugewiesen, siehe 3.3. Infolgedessen bilden die Schlüssel-Wert-Paare *msgid* und *msgstr* ein Paar aus Bezeichner im Quelltext und Text in der Zielsprache. Um beim Kompilieren den Wert von *msgid* mit dem Wert von *msgstr* zu ersetzen, muss aus dem *.po* ein *Machine Object (.mo)* erzeugt werden. Dies geschieht mit dem Kommandozeilenbefehl *msgfmt*. In der Konsequenz kann aus dem *.mo* der jeweiligen Zielsprache und dem Quelltext eine übersetzte Version der Software erzeugt werden<sup>31 32 33</sup>

was?

```
1 #include <stdio.h>
2 int main() {
3     printf(_("msg_Greet"));
4     getchar();
5     return 0;
6 }
```

Quelltext 3.1: Ausgezeichneter Quelltext

```
1 msgid "msg_Greet "
2 msgstr ""
```

Quelltext 3.2: Erstelltes .po

```
1 msgid "msg_Greeting"
2 msgstr "Hallo Welt"
```

Quelltext 3.3: Übersetztes .po

Eine Alternative zur Auszeichnung, ist das automatische Erkennen zu übersetzender Strings und dem anschließenden Übersetzen dieser. Die automatische Erkennung zu übersetzender Strings ist durch einen Parser anhand einer kontextfreien Grammatik möglich.<sup>34 35</sup> Aus jeder kontextfreien Grammatik kann ein Parser generiert werden.<sup>36</sup>

<sup>31</sup>Vgl. Tykhomyrov 2002.

<sup>32</sup>Vgl. Mauro 1999.

<sup>33</sup>Vgl. GNU o.D.

<sup>34</sup>Vgl. Wang et al. 2009, S. 556.

<sup>35</sup>Vgl. Leiva und Alabau 2015, S. 6.

<sup>36</sup>Vgl. Unger 1968, S. 240 - 246.



Eine kontextfreie Grammatik ist ein 4-Tupel:

$$G = (N, \Sigma, P, S) \quad (3.1)$$

$N$  bildet das Vokabular aller Nichtterminalsymbole der Grammatik  $G$ .  $\Sigma$  bildet das Vokabular aller Terminalsymbole der Grammatik  $G$ .  $P$  ist eine endliche Menge von Produktionen der Form  $A \rightarrow \omega$ . Es gilt  $A \in N$  und  $\omega \in (N \cup \Sigma)^*$ .  $S \in N$  wird als Startsymbol bezeichnet und bildet den Anfang der Produktion.<sup>37</sup>

Zurzeit existiert jedoch kein Lokalisierungstool für MEL oder MXML.

Alles etwas knapp. Gibt es keine Surveys?

### 3.3 Standards der Lokalisierung

Globalisierung, Internationalisierung und Lokalisierung umfassen einen weiten Bereich von Aufgaben. Um die Ausführung bestimmter Aufgaben<sup>38</sup> zu ermöglichen oder effizienter zu gestalten, wurden seit Gründung der Localization Industry Standards Association (LISA) verschiedene Standards definiert. Mit Gründung verschiedener Organisationen wurden Standards mit teilweise ähnlichen Zweck definiert. Die definierten Standards sind in diesem Abschnitt aufgeführt, um einen Überblick über diese, ihren Zweck und ihre Relevanz für diese Arbeit zu bieten.

#### 3.3.1 TMX

Translation Memory Exchange (TMX) ist ein Standard, der von der *LISA* entwickelt wurde und seit deren Insolvenz von der *Globalization and Localization Association (GALA)* verwaltet wird. Zweck von TMX ist das Bereitstellen einer standardisierten Methode zum Beschreiben von TM Daten, welche zwischen Tools und/oder Übersetzungsdienstleistern ausgetauscht werden. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>39</sup>

#### 3.3.2 XML:TM

XML Text Memory (XML:TM) ist ein Standard, der von der *LISA* entwickelt wurde und seit deren Insolvenz von der *GALA* verwaltet wird. Zweck von XML:TM ist das Bereitstellen einer standardisierten Methode zum Speicher von TM Daten innerhalb

<sup>37</sup>Vgl. Korenjak 1969, S. 614.

<sup>38</sup>Beispielsweise den Austausch von Daten zwischen Lokalisierungstools

<sup>39</sup>Vgl. Savourel 2005.

eines XML-Dokuments. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>40</sup>

### 3.3.3 SRX

Segmentation Rules Exchange (SRX) ist ein Standard, der von der *LISA* entwickelt wurde und seit deren Insolvenz von der *GALA* verwaltet wird. Zweck von SRX ist das Bereitstellen einer standardisierten Methode zum Beschreiben von Segmentierungsregeln, welche zwischen Tools und/oder Übersetzungsdienstleistern ausgetauscht werden. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>41</sup>

### 3.3.4 GMX-V

Global Information Management Metrics Volume (GMX-V) ist ein Standard, der von der *LISA* entwickelt wurde und seit deren Insolvenz von der *GALA* verwaltet wird. Zweck von GMX-V ist das Definieren einer Metrik zum eindeutigen Messen einer gegebenen globalen Informationsverwaltungsaufgabe. Des Weiteren beschreibt GMX-V eine standardisierte Methode zum Austauschen der Metriken zwischen Tools und/oder Übersetzungsdienstleistern. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>42</sup>

### 3.3.5 TBX

Term Base Exchange (TBX) ist ein Standard, der von der *LISA* entwickelt wurde und seit deren Insolvenz von der *GALA* verwaltet wird. Zweck von TBX ist das Bereitstellen einer standardisierten Methode zum Beschreiben terminologischer Daten, welche zwischen Tools und/oder Übersetzungsdienstleistern ausgetauscht werden. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>43</sup>

---

<sup>40</sup>Vgl. Zydrón, Raya M. Rodolfo und Bogacki 2007.

<sup>41</sup>Vgl. Pooley und Raya M. Rodolfo 2008.

<sup>42</sup>Vgl. Zydrón, Walters und Raya M. Rodolfo 2007.

<sup>43</sup>Vgl. GALA 2008, S. vii.

### 3.3.6 XLIFF

XML Localisation Interchange File Format (XLIFF) ist ein Standard, der von der *Organization for the Advancement of Structured Information Standards (OASIS)* entwickelt wurde und seither von dieser verwaltet wird. Zweck von XLIFF ist das Bereitstellen einer standardisierten Methode zum Speichern von Lokalisierungsdaten und deren Austausch zwischen einzelnen Prozessschritten der Lokalisierung. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Ziel des Standards ist es, Kompatibilität zwischen verschiedenen Tools zu ermöglichen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als relevant für diese Arbeit heraus.<sup>44</sup>

### 3.3.7 Trans-WS

Translation Web Services (Trans-WS) ist ein Standard, der von der *OASIS* entwickelt wurde und seither von dieser verwaltet wird. Zweck von Trans-WS ist das Bereitstellen einer standardisierten Methode zur Benutzung und Beschreibung eines Webservices innerhalb der Übersetzungsindustrie. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>45</sup>

### 3.3.8 UTX

Universal Terminology Exchange (UTX) ist ein Standard, der von der *Asia-Pacific Association for Machine Translation (AAMT)* entwickelt wurde und seither von dieser verwaltet wird. Zweck von UTX ist das Bereitstellen eines standardisierten lexikalischen Formats für regelbasierte Übersetzungssoftware und das Bereitstellen eines Formats zur Darstellung von Glossaren, welche in den Bereichen von CAT und NLP eingesetzt werden. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>46</sup>

### 3.3.9 OLIF

Open Lexicon Interchange Format (OLIF) ist ein Standard, der von dem *OLIF Consortium* entwickelt wurde und seither von diesem verwaltet wird. Zweck von OLIF ist das Bereitstellen einer standardisierten Methode zum Beschreiben von lexikalischen

---

<sup>44</sup>Vgl. Schnabel et al. 2014.

<sup>45</sup>Vgl. Reynolds et al. 2006.

<sup>46</sup>Vgl. Yuji et al. 2018, S. 4.

und terminologischen Daten, welche zwischen Tools und/oder Übersetzungsdienstleistern ausgetauscht werden. Während des Austauschprozesses dürfen keine wichtigen Daten verloren gehen. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als nicht relevant für diese Arbeit heraus.<sup>47</sup>

### 3.3.10 ITS

Internationalization Tag Set (ITS) ist ein Standard, der von dem *World Wide Web Consortium (W3C)* entwickelt wurde und seither von diesem verwaltet wird. Zweck von ITS ist das Bereitstellen einer XML-basierten Auszeichnung von Metadaten im Bereich der Internationalisierung, Übersetzung und Lokalisierung. Dieser Standard wurde im Vorfeld dieser Arbeit recherchiert und stellte sich als relevant für diese Arbeit heraus.<sup>48</sup>

---

<sup>47</sup>Vgl. McCormick, M., Susan 2000.

<sup>48</sup>Vgl. Filip et al. 2013.

- Woher kommen die?  
- Anforderungen nummerieren

$A_1$   
:  
 $A_n$

## 4 Anforderungen an das Lokalisierungstool

Zum Zweck der Anforderungsanalyse ist zunächst das Ziel der Arbeit zu betrachten.

Ziel dieser Arbeit ist die Konzeption einer Software, welche zu übersetzende und übersetzte Strings der Texte eines Movelets automatisiert verarbeitet.

Zur automatisierten Verarbeitung von Strings müssen diese zunächst erfasst werden. Hierzu ist zu betrachten, in welchem Kontext Strings in Movelets vorkommen. Der Quelltext von Movelets ist String-basiert und wird in den Programmiersprachen MXML und MEL geschrieben. Für die automatisierte Verarbeitung sind ausschließlich die zu übersetzenden Strings des Quelltexts relevant. Zu übersetzende Strings werden auf der Benutzeroberfläche des Movelets angezeigt. Auf der Benutzeroberfläche angezeigte Strings werden als Benutzeroberflächenstrings bezeichnet. Benutzeroberflächenstrings sind in MXML entweder Inhalt von Elementen oder Werte von Attributen.<sup>49</sup> Eine weitere Form der Auslagerung von Benutzeroberflächenstrings sind die durch MXML ausgezeichneten Stammdaten.<sup>50</sup> Die Programmiersprache MEL zeichnet Benutzeroberflächenstrings in Form von Stringliteralen aus. Zwecks Auszeichnung wird entweder der Apostroph(U+0027):' oder das Anführungszeichen(U+0022):" verwendet. Im Kontext dieser Arbeit werden Apostroph und Anführungszeichen unter dem Begriff Anführungszeichen zusammengefasst.<sup>51</sup> Seit 2018 ist es möglich MXML- und MEL-Quelltext mithilfe des Movilizer Gradle Plug-ins zu erzeugen. Dieses Plug-in ermöglicht die Verwendung von Ressource Dateien, welche Benutzeroberflächenstrings als Wert von Schlüssel-Wert-Paare beinhalten.<sup>52</sup> Folglich muss das Lokalisierungstool folgende Strings erfassen:

- MXML-Elemente inklusive Stammdatenelement, siehe Beispiel 4.1
- MXML-Attribute inklusive Stammdatenelement, siehe Beispiel 4.1
- MEL-Stringlitterale, siehe Beispiel 4.2
- Werte in Movilizer Gradle Plug-in Ressource Dateien, siehe Beispiel 4.3


<sup>49</sup>Vgl. Nitschkowski 2016c.

<sup>50</sup>Vgl. Nitschkowski 2016b.

<sup>51</sup>Vgl. Nitschkowski 2018g.

<sup>52</sup>Vgl. De Mula und Paul Janzen 2018.

```
1 <masterdataPoolUpdate pool="customerData">
2     <update key="user_Cus1" group="USA">
3         <data>
4             <entry name="name">
5                 <valstr>name_Cus1 Doe</valstr>
6             </entry>
7         </data>
8     </update>
9 </masterdataPoolUpdate>
```



Quelltext 4.1: MXML-Elemente und -Attribut inklusive Stammdaten

```
1 <onScreenValueChangeEvent>
2     function(k, cK, v, d) {
3         setAnswerValueByClientKey(k, cK, "msg_Greet")
4     }
5 </onScreenValueChangeEvent>
```

Quelltext 4.2: MEL-Stringlitterale

```
1 Key1: msg_Greet
2 Key2:
3     nestedKey: msg_Bye
```

Quelltext 4.3: Werte in Movilizer Gradle Plug-in Ressource Dateien

Ziel der automatisierten Verarbeitung von zu übersetzenden Strings ist es eine übersetztes Movelet zu erzeugen. Zu diesem Zweck werden zu übersetzende Strings extrahiert und übersetzt. Resultierend daraus müssen extrahierte Strings in einem Format gespeichert sein, welches ohne Kompatibilitätsprobleme an einen Übersetzer oder eine Übersetzungssoftware weitergeleitet werden kann. Die übersetzten Strings müssen anschließend im selben Format empfangen werden. Die extrahierten Strings werden im Quelltext durch die Übersetzten ausgetauscht.

Des Weiteren ist zu bedenken, dass Lokalisierung und Entwicklung des Movelets gegebenenfalls parallelisiert werden, um eine gleichzeitige Veröffentlichung mehrerer Lokalisierungen des Movelets zu ermöglichen. Aus diesem Grund muss das Lokalisierungstool Deltas erkennen, um redundantes Extrahieren zuvor bereits extrahierter Strings zu vermeiden. Zusätzlich ist es gegebenenfalls nötig, eine zuvor lokalisierte und verworfene Version wiederherzustellen. Daher ist die Versionsverwaltung im Kontext des Lokalisierungstools zu beachten.

Was ist das nun? Ggf. Fragen? PW?  
Bezug zu Anforderungen?

## 5 Extrahieren zu übersetzender Strings aus dem Quelltext

Wie in Kapitel 4 erarbeitet, müssen zur automatischen Verarbeitung von zu übersetzenden und übersetzten Strings, die zu übersetzenden Strings aus dem Quelltext extrahiert werden. Diese Extraktion wird durch einen Parser vorgenommen. Der Quelltext eines Movelets ist String-basiert.<sup>53</sup> Infolgedessen müssen zu übersetzende Strings vom restlichen Quelltext unterschieden werden, um diese zu extrahieren. Zum Unterscheiden von Strings gibt es verschiedene Möglichkeiten. Diese sind in diesem Kapitel aufgeführt.

### 5.1 Auszeichnung zu übersetzender Strings

Eine Möglichkeit zu übersetzende Strings vom restlichen Quelltext zu unterscheiden ist das Auszeichnen dieser mit einer eindeutigen Auszeichnung. MXML-Quelltext basiert auf XML.<sup>54</sup> Das *W3C* hat zum Auszeichnen von XML bezüglich Lokalisierung den ITS Standard entwickelt. Aus diesem Grund empfiehlt es sich diesen Standard zum Auszeichnen von zu übersetzenden Strings im Kontext von MXML zu verwenden. ITS bietet zwei grundlegende Ansätze des Auszeichnens: einen Globalen und einen Lokalen. Der globale Ansatz zeichnet Strings nicht direkt aus. Stattdessen wird zu diesem Zweck die Abfragesprache XML Path Language (XPath) verwendet. Ein *rules* Element wird mithilfe eines Parameters ausgezeichnet und erhält eine Abfrage als weiteren Parameter. Alle XML-Elemente, welche Teil des Ergebnisses dieser Abfrage sind, gelten als im gleichen Maße ausgezeichnet, wie das *rules* Element selbst. Der lokale Ansatz zeichnet einzelne XML-Elemente und deren Inhalt direkt mithilfe eines Parameters aus.<sup>55 56</sup>

Die Struktur von MEL und Movilizer Gradle Plug-in Ressource Dateien basiert nicht auf XML.<sup>57 58</sup> Deshalb ist ITS zu deren Auszeichnung nicht ausreichend. Ein globaler Ansatz mithilfe einer Abfragesprache wie von ITS vorgesehen ist ohne die zusätzliche

---

<sup>53</sup>Vgl. Nitschkowski 2018c.

<sup>54</sup>Vgl. Nitschkowski 2018c.

<sup>55</sup>Vgl. Filip et al. 2013.

<sup>56</sup>Notizen zur Implementierung der Extraktion zu übersetzender Strings anhand einer Auszeichnung, sowie einer kontextfreien Grammatik finden sich im Anhang in Sektion A.1

<sup>57</sup>Vgl. Nitschkowski 2018d.

<sup>58</sup>Vgl. De Mula und Paul Janzen 2018.

Entwicklung einer Abfragesprache für MEL und Movilizer Gradle Plug-in Ressourcen Dateien nicht möglich. Ein lokaler Ansatz durch die direkte Auszeichnung von zu übersetzenden Strings hingegen ist möglich. Diese Auszeichnung muss eindeutig sein, da es zu ungewollten Extraktionen kommt, falls diese Auszeichnung auch in einem anderen Kontext von Movelets verwendet wird. Beispielsweise im Kontext von MEL werden Methodenparameter durch die geöffnete(U+0028): ( und geschlossene Rundklammer(U+0029): ) ausgezeichnet.<sup>59</sup> Wird diese Auszeichnung für zu übersetzende Strings verwendet, gelten auch alle Methodenparameter als ausgezeichnet. Eine im Gesamtkontext von Movelets eindeutige Auszeichnung ist die *'GNU gettext'*-artige Auszeichnung. Diese beginnt mit einem Unterstrich(U+005F): \_ gefolgt von einer offenen Rundklammer und endet mit einer geschlossenen Rundklammer \_(). *GNU gettext* selbst kann für Movelets nicht eingesetzt werden.<sup>60</sup> Die Auszeichnung jedoch ist aufgrund von Kürze und Eindeutigkeit vorteilhaft. Wie bereits erwähnt, übernimmt diese Auszeichnung dieselbe Aufgabe wie der lokale Ansatz von ITS. Demzufolge kann auch zu übersetzender MXML-Inhalt mit dieser Auszeichnung ausgezeichnet werden.

Besonders zu beachten ist, dass sich das Extraktionsverhalten innerhalb des Kontextes von MEL-Parametern und von MXML-Attributen im Vergleich zu den anderen Kontexten unterscheidet. Diese beiden Kontexte sind durch den umgebenden MXML-Kontext bestimmbar, da MEL-Quelltext ausschließlich in bestimmten MXML-Elementen vorkommen kann und MXML-Attributen anhand der MXML-Definition bestimmbar sind.<sup>61</sup> In diesen beiden Kontexten sind alle Stringlitterale mit Anführungszeichen ausgezeichnet.<sup>62 63</sup> In der Konsequenz müssen während der Extraktion das erste und letzte Anführungszeichen eines ausgezeichneten Strings entfernt werden. Des Weiteren ist der MEL-Kontext der einzige, in welchem zusammengefügte Strings möglich sind. Diese sind durch die *concat* Methode, in der die Eingabeparameter von Rundklammern umgeben sind, ausgezeichnet: *concat()*. Die Eingabeparameter sind durch Komma(U+002C): , von einander abgetrennt.<sup>64</sup> Daher muss während der Extraktion die Methode, sowie die Kommata und das erste und letzte Anführungszeichen nach einem Komma entfernt werden. Die Variablen müssen als Platzhalter maskiert werden. Variablen sind alle Strings, welche nach einer geöffneten Rundklammer oder einem Komma nicht mit einem schließenden Anführungszeichen beginnen. Das Beispiel 5.2 zeigt, das Ergebnis einer Extraktion eines in 5.1 ausgezeichneten zusammengesetzten Strings.

```
1      varName = call()($global:UDFgetUserName)();
```

<sup>59</sup>Vgl. Nitschkowski 2018e.

<sup>60</sup>Vgl. GNU o.D.

<sup>61</sup>Vgl. Nitschkowski 2015a.

<sup>62</sup>Vgl. Bray et al. 2008.

<sup>63</sup>Vgl. Nitschkowski 2018h.

<sup>64</sup>Vgl. Nitschkowski 2018a.



```
2      _(concat("msg_Greet", varName));
```

Quelltext 5.1: Ausgezeichneter zusammengesetzter String

```
1      msg_Greet %varName
```

Quelltext 5.2: Extrahierter zusammengesetzter String

## 5.2 Automatisches Erkennen zu übersetzender Strings

Eine Alternative zum Auszeichnen zu übersetzender Strings ist die syntaktische und semantische Analyse des Quelltexts. Anhand dieser können Benutzeroberflächenstrings automatisiert aus dem Quelltext extrahiert werden.<sup>65 66</sup> Benutzeroberflächenstrings sind wie in Kapitel 4 festgestellt Inhalte von MXML-Elementen, Werte von MXML-Attributen, MEL-Stringlitterale und Werte in Movilizer Gradle Plug-in Ressource Dateien. Die Entscheidung, welche dieser Strings Benutzeroberflächenstrings sind, ist komplex.

Die Entscheidung über Inhalte von MXML-Elementen und Werte von MXML-Attributen erfolgt anhand des umgebenden MXML-Kontextes. Der MXML-Kontext wird außerhalb von Stammdaten am stärksten durch das `<question>`-Element bestimmt. Das `<question>`-Element kennzeichnet den Beginn eines Subprozesses innerhalb des Movelets. Der Subprozess wird durch das `type`-Attribut des `<question>`-Elements festgelegt. Im Beispiel 5.3 besitzt das `type`-Attribut des `<question>`-Elements den Wert 0. Im Beispiel 5.4 hingegen besitzt das `type`-Attribut des `<question>`-Elements den Wert 41. In der Konsequenz ist der Text *Hallo Welt* im Beispiel 5.3 ein Benutzeroberflächenstring, während er im Beispiel 5.4 keiner ist.<sup>67 68</sup>

```
1 <movelet moveletKey="MOV01 initialQuestionKey="#1">
2     <question key="#1" type="0">
3         <answer key="#1_1" nextQuestionKey="END">
4             <text>Hallo Welt</text>
5         </answer>
6     </question>
```

<sup>65</sup>Vgl. Wang et al. 2009, S. 556.

<sup>66</sup>Vgl. Leiva und Alabau 2015, S. 6.

<sup>67</sup>Vgl. Nitschkowski 2016a.

<sup>68</sup>Vgl. Nitschkowski 2018f.

```
7 </movelet>
```

### Quelltext 5.3: Message Screen

```
1 <movelet moveletKey="MOV01" initialQuestionKey="#1">
2     <question key="#1" type="41">
3         <answer key="#1_1" nextQuestionKey="END"
4             <text>Hallo Welt</text>
5         </answer>
6     </question>
7 </movelet>
```

### Quelltext 5.4: Epsilon Screen

Die Entscheidung über MEL-Stringliterals ist indirekt möglich. Inhalte von MXML-Elementen und Werte von MXML-Attributen werden durch MEL-Methoden abhängig von Methodenparameter verändert. Methodenparameter sind Ausdrücke. Zu diesen Ausdrücke zählen unter anderem MEL-Stringliterals und MEL-Variablen. MEL-Variablen, denen ein Ausdruck mit dem Wert des MEL-Stringliterals zugewiesen wird, liefern den selben Wert zurück. Folglich lassen sich alle Ausdrücke, welche den Wert des MEL-Stringliterals zurückliefern, anhand der Zuweisungen bestimmen. Anhand der Ausdrücke lassen sich alle MEL-Methoden bestimmen, welche diese Ausdrücke als Methodenparameter besitzen. Anhand der MEL-Methoden lassen sich die von ihnen manipulierten Inhalte von MXML-Elementen und Werte von MXML-Attributen bestimmen. Anhand dieser kann die Entscheidung wie bereits erarbeitet getroffen werden. Resultierend daraus wird entschieden, ob das jeweilige MEL-Stringliterals ein Benutzeroberflächenstring ist.

Die Entscheidung über Inhalte von Stammdaten und andere externe Ressource Dateien erfolgt anhand der aufgerufenen MEL-Methoden, welche diese Inhalte auslesen. Anhand der MEL-Methoden wird bestimmt, welche Inhalte abgefragt werden. Hierfür müssen alle externen Ressource Dateien für die Abfrage zugänglich sein. Methoden sind Ausdrücke. Für Ausdrücke kann die Entscheidung wie bereits erarbeitet getroffen werden. Demzufolge wird entschieden, ob die jeweiligen Inhalte Benutzeroberflächenstrings sind.

Die Entscheidung über Werte in Movilizer Gradle Plug-in Ressource Dateien erfolgt nach dem Einfügen dieser in den Quelltext. Nach dem Einfügen sind alle Werte teil des Quelltexts. Für diesen kann die Entscheidung wie bereits erarbeitet getroffen werden. Infolgedessen wird entschieden, ob die jeweiligen Werte Benutzeroberflächenstrings sind.

## 5.3 Diskussion der Lösungsansätze

In diesem Abschnitt sind die Vor- und Nachteile beider Lösungsansätze aufgeführt. Anhand dieser Diskussion wird der passende Lösungsansatz ausgewählt.

Die Auszeichnung zu übersetzender Strings erfolgt manuell, während die syntaktische und semantische Analyse automatisch erfolgt. Deshalb ist die manuell Auszeichnung aufwendiger.

Die Implementierung eines Lokalisierungstools, welches die Auszeichnung verwendet, muss zum Erkennen der Auszeichnung lediglich zwischen MEL-Kontext und anderem Kontext unterscheiden. Bei der syntaktischen und semantischen Analyse hingegen müssen weit mehr Kontexte beachtet und zurückverfolgt werden. Daher ist die Implementierung der syntaktischen und semantischen Analyse weit komplexer und aufwendiger.

Movilizer wird stetig um neue Funktionalität erweitert. Die Auszeichnung ist String-basiert und damit vollkommen unabhängig von neuer Funktionalität einsetzbar. Die einzige Einschränkung ist, dass neue Funktionalität die gewählte Auszeichnung nicht in anderem Kontext verwenden darf. Die semantische und syntaktische Analyse hingegen basiert auf der Deutung der einzelnen Funktionalitäten. Folglich muss die semantische und syntaktische Analyse um jede neue Funktionalität erweitert werden. In der Konsequenz benötigt die Auszeichnung weit weniger Wartung und ist weniger aufwendig.

Anhand der syntaktischen und semantischen Analyse werden Benutzeroberflächen-strings extrahiert. Diese sind großteils jedoch nicht zwangsweise zu übersetzen. Beispielsweise werden *Globale Trade Item Numbers* auf der Benutzeroberfläche einer *Track and Trace* Software angezeigt. Diese sind jedoch nicht zu übersetzen. Die Unterscheidung, ob eine Benutzeroberflächenstring zu übersetzen ist, kann in den beschriebenen Lösungsansätzen komfortabel durch die Auszeichnung ermöglicht werden.

Aufgrund der aufgeführten Argumente, wird für dieses Lokalisierungstool der Lösungsansatz der Auszeichnung verwendet.

## 6 Schnittstelle des Lokalisierungstools zu anderer Software

Wie in Kapitel 4 erarbeitet, müssen zur automatischen Verarbeitung von zu übersetzenden und übersetzten Strings die extrahierten Strings übersetzt werden. Übersetzer werden beim Übersetzen extrahierter Strings in der Regel von Übersetzungssoftware unterstützt. Des Weiteren werden Übersetzungen in TM und/oder Terminologiedatenbanken gespeichert.<sup>69</sup> Zu diesem Zweck werden die extrahierten Strings an andere Software gesendet. Die übersetzten Strings müssen von dem Lokalisierungstool empfangen werden. Deshalb ist ein kompatibles Austauschformat benötigt, in welches das Lokalisierungstool zu übersetzende Strings speichert, beziehungsweise aus welchem das Lokalisierungstool übersetzte Strings ausliest. Dieses Austauschformat bietet die Schnittstelle des Lokalisierungstools zu anderer Software.

Die Standards, welche Austauschformate für Daten der Lokalisierung definieren, wurden in Kapitel 3 vorgestellt. Der in Kapitel 5 für die Extraktion von Strings vorgesehene ITS-Standard unterstützt XLIFF, welches von *OASIS* entwickelt wurde.<sup>70</sup> Des Weiteren ist XLIFF ein Meta-Lokalisierungsformat, welches aus anderen Formaten erzeugbar und in andere Formate umwandelbar ist.<sup>71</sup> Aus diesen Gründen wird XLIFF als Schnittstelle des Lokalisierungstools verwendet. XLIFF bietet zwei Elemente. Das eine Element passt zur Verarbeitung zu übersetzender Strings und das andere Element zur Verarbeitung übersetzter Strings. Das *source* Element enthält zu übersetzenden Text.<sup>72</sup> Infolgedessen werden die von dem Lokalisierungstool extrahierten zu übersetzenden Strings in diesem Element gespeichert. Das *target* Element enthält die Übersetzung zu dem ihm zugehörigen *source* Element.<sup>73</sup> Aus diesem Grund werden in diesem Element die übersetzten Strings gespeichert, mit welchen das Lokalisierungstool die zu übersetzenden Strings ersetzt.

Die Lokalisierung eines Movelets muss gegebenenfalls in unterschiedlichen Sprachen erfolgen. Deshalb müssen für jede Übersetzung eigene XLIFF Inhalte erzeugt werden. In der Konsequenz muss die Sprache der Übersetzungen vor dem Extrahieren und vor dem Ersetzen der Strings angegeben werden. Die Sprache der Übersetzungen der XLIFF Inhalte kann in dem jeweiligen *trgLang* Attribut gespeichert werden.

<sup>69</sup>Vgl. O'Brien, O'Hagan und Flanagan 2010, S. 187.

<sup>70</sup>Vgl. Filip et al. 2013.

<sup>71</sup>Vgl. Reineke und Schmitz 2005, S. 162.

<sup>72</sup>Vgl. Schnabel et al. 2014, S. 21.

<sup>73</sup>Vgl. Schnabel et al. 2014, S. 21.

Die ~~XLIFF~~<sup>74</sup> Inhalte werden in XLIFF Dateien gespeichert. Es ist möglich diese Inhalte in einer einzigen Datei zu speichern. In dem Falle, dass das Movelet eine große Anzahl zu übersetzender Strings beinhaltet, ist diese Datei sehr umfangreich. Zum Zweck der Extraktion zu übersetzender Strings und Ersetzung dieser müssen die jeweiligen Strings eingefügt, beziehungsweise gesucht werden. Zusätzliche Struktur ermöglicht den Einsatz von Sortier- und Suchalgorithmen mit einem geringeren Aufwand.<sup>74 75</sup> Folglich wird das Verarbeiten der Inhalte effizienter. Aufgrund dessen ist es sinnvoll, die Ressourcen innerhalb einer Dateistruktur zu ordnen. Das Herausarbeiten der effizientesten Struktur ist nicht Teil dieser Arbeit und für die Implementierung des Lokalisierungstools offengelassen.

---

<sup>74</sup>Vgl. Suresh und George 2018, S. 51.

<sup>75</sup>Vgl. Hoda 2015, S. 1723.

## 7 Ersetzen zu übersetzender durch übersetzte Strings

Wie in Kapitel 4 erarbeitet müssen zur automatischen Verarbeitung von zu übersetzenden und übersetzten Strings die extrahierten Strings durch die Übersetzten ersetzt werden. Die Ersetzung erfolgt nicht im Quelltext selbst, sondern in einer Kopie dieses Quelltexts. In der Konsequenz kann dieser Quelltext für die Erzeugung vieler lokalisierter Quelltexte eines Movelets verwendet werden. Resultierend daraus fungiert der Quelltext als Vorlage und ist mit dem Movilizer Gradle Plug-in kompatibel, da dieser ebenfalls mit einer Vorlage des Quelltexts arbeitet.<sup>76</sup> Zum Zweck der Ersetzung der Strings ist eine Zuordnung nötig. Diese Zuordnung ordnet einem übersetzten String einen eindeutigen Bezeichner des zu ersetzenden String des Quelltexts zu. Der übersetzte String ist in der ~~XLIFF~~-Datei gespeichert. Deshalb wird die ~~XLIFF~~-Datei für die Zuordnung verwendet. Grundlegend existieren zwei Möglichkeiten ein Objekt und damit einen String zu identifizieren. Die erste Möglichkeit ist die Identifikation per Name, die zweite Möglichkeit ist die Identifikation per Zeiger.<sup>77</sup>

### 7.1 Identifikation des zu ersetzenden Strings per Name

Der Name eines zu ersetzenden Strings muss eindeutig einem übersetzten String zugeordnet sein. Des Weiteren muss der Name dem zu ersetzenden String zugeordnet sein, der durch den Namen identifiziert wird. Ist der zu ersetzende String gleich seinem Namen, so erfolgt die Zuordnung von zu ersetzenden String zu Name implizit. Der zu ersetzende String ist infolge der Extraktion in einem *source*-Element einer XLIFF-Datei gespeichert. Der zugehörige übersetzte String befindet sich im zugehörigen *target*-Element. In der Konsequenz erfolgt auch diese Zuordnung implizit. Um Kollisionen zu vermeiden müssen Namen eindeutig sein, das bedeutet jeder Name darf nur in einem *source*-Element vorhanden sein. Jedem Namen ist also genau ein übersetzter String zugeordnet.

Werden mehrere Namen dem gleichen übersetzten String zugeordnet, entstehen keine Kollisionen sondern Redundanzen. Daher sollte für den gleiche übersetzten String der gleiche Name verwendet werden.

---

<sup>76</sup>Vgl. De Mula und Paul Janzen 2018.

<sup>77</sup>Vgl. Berners-Lee, Fielding und Masinter 2005, S. 6.

Eine Möglichkeit für kurze, eindeutige Namen sind mit Affixen versehene generische Namen. In Beispiel 7.1 wird der mit Affix versehene generische Bezeichner aus Beispiel 7.2 eindeutig einem übersetzten String zugeordnet.

```
1 <movelet moveletKey="MOV01" initialQuestionKey="#1">
2     <question key="#1" type="41">
3         <answer key="#1_1" nextQuestionKey="END"
4             <text>_(msg_Greet)</text>
5         </answer>
6     </question>
7 </movelet>
```

Quelltext 7.1: Generischer Name mit Affix

```
1 <source>
2     msg_Greet
3 </source>
4 <target>
5     Hallo Welt
6 </target>
```

Quelltext 7.2: Zuordnung von Name zu übersetzten String

Gegebenenfalls muss jedoch ein bereits erstelltes Movelet nachträglich lokalisiert werden. In diesem Falle sind die zu übersetzenden Strings des Movelets keine generischen Namen sondern Texte in einer Ausgangssprache. Diese Texte können jedoch auch als Namen verwendet werden. Gleiche Text benötigen in bestimmten Kontexten unterschiedliche Übersetzungen. Folglich entstehen Kollisionen. Des Weiteren benötigen unterschiedliche Texte in bestimmten Kontexten die gleiche Übersetzung. Infolgedessen entstehen Redundanzen. Redundanzen und Kollisionen können vermieden werden, indem zunächst die Texte selbst ausgezeichnet werden. Diese Texte werden dann in den XLIFF Dateien generischen Bezeichnern zugeordnet und durch diese ersetzt. In der Konsequenz entsteht eine Movelet Vorlage mit generischen Namen.

## 7.2 Identifikation des zu ersetzenden Strings per Zeiger

Der Zeiger auf einen zu ersetzenden String muss eindeutig einem übersetzten String zugeordnet sein. Des Weiteren muss der Zeiger dem zu ersetzenden String zugeordnet sein, auf welchen der Zeiger zeigt. Der zu ersetzende String ist infolge der Extraktion in einem *source*-Element einer XLIFF-Datei gespeichert. Das *source*-Element kommt

in seinem Elternelement einmalig vor. Im Inhalt des Elternelements wird der Zeiger gespeichert, die Zuordnung erfolgt somit implizit. Der zugehörige übersetzte String befindet sich in dem *source*-Element zugehörigen *target*-Element. Folglich erfolgt auch diese Zuordnung implizit.

Es gibt verschiedene Arten von Zeigern. Ein Beispiel ist das geordnete Indexieren des Movelets und das Verwenden des Index als Zeiger. Ein weiteres Beispiel ist es den Pfad eines Objekts des Ableitungsbaums des Movelets als Zeiger zu verwenden. Die Entscheidung über die geeignetste Variante eines Zeigers ist für die Implementierung offengelassen.

## 7.3 Behandlung fehlender Übersetzungen

Die Zuordnung des Bezeichners erfolgt für beide Arten der Identifikation bei der Extraktion. Bei der Extraktion sind die übersetzten Strings noch nicht übersetzt und deshalb leer. Werden die leeren Strings von dem Lokalisierungstool verwendet erzeugt dies ein ungewolltes Ergebnis. Aus diesem Grund muss vor leeren Strings gewarnt werden. Die leeren Strings sollten in der Warnung aufgelistet werden.

Zum Testen muss gegebenenfalls eine unfertige Version des Movelets erzeugt werden. Daher muss es dennoch möglich sein die Warnung zu ignorieren.

## 7.4 Diskussion der Lösungsansätze

In diesem Abschnitt sind die Vor- und Nachteile beider Lösungsansätze aufgeführt. Anhand dieser Diskussion wird der passende Lösungsansatz gewählt.

Wie in Abschnitt 7.1 vorgesehen werden für die gleichen übersetzten Strings der gleiche Name verwendet. Resultierend daraus steigt zwar der organisatorische Aufwand, jedoch werden Redundanzen vermieden. Der organisatorische Aufwand steigt, da sich die beteiligten Entwickler des Movelets auf die verwendeten Namen einigen müssen. Die Vermeidung von Redundanzen verringert den Übersetzungsaufwand, da der gleich zu übersetzende String nicht mehrfach übersetzt werden muss. Infolgedessen ist der Übersetzungsaufwand für die Identifikation per Name geringer.

Zum Zweck der gleichzeitigen Veröffentlichung mehrere lokalisierter Versionen eines Movelets müssen Lokalisierung und Entwicklung parallel stattfinden. In der Konsequenz wird der Quelltext nach dem Erstellen der XLIFF Dateien und damit nach dem Erstellen der Zeiger beziehungsweise Namen verändert. Daher ist es möglicherweise nötig, die Position eines zu übersetzenden Strings zu ändern. Folglich zeigt der



zugehörige Zeiger nicht mehr auf den zu übersetzenden String und muss aktualisiert werden. Auch der Name kann geändert werden, jedoch wird in diesem Falle aufgrund der Extraktion ein neues Element in der XLIFF-Datei angelegt. Deshalb benötigt die Identifikation per Zeiger zusätzliche Logik, falls das Movelet verändert wird.

Aufgrund der aufgeführten Argumente, wird für dieses Lokalisierungstool der Lösungsansatz der Identifikation per Name verwendet.

Bezug zu Aufg.  
Ergebnis?

## 8 Ansatzzeitpunkt des Lokalisierungstools

Generell lassen sich bei der Programmierung verschiedene Zeitpunkte unterscheiden. In dieser Arbeit wird zwischen den Folgenden unterschieden:

- Vorübersetzungszeit
- Übersetzungszeit
- Bindungszeit<sup>78</sup>
- Laufzeit

Der Zeitpunkt an welchem das Lokalisierungstool ansetzt ergibt sich aus den zuvor ausgewählten Lösungsansätzen. Das Ersetzen der extrahierten mit den zu übersetzen String erfolgt auf Basis einer Vorlage des Quelltexts und erzeugt einen lokalisierten Quelltext. Aufgrund dessen, dass das Lokalisierungstool auf nicht kompilierten Quelltext arbeitet und einen nicht kompilierten Quelltext erzeugt, setzt das Lokalisierungstool Vorübersetzungszeit an.

---

<sup>78</sup>Vgl. Presser und White 1972, S. 155.

## 9 Versionsverwaltung der Lokalisierung

Versionsverwaltung wird seit vielen Jahren aktiv in der Software Entwicklung angewendet. Die Aufgabe der Versionsverwaltung ist das zurück verfolgbare Dokumentieren der Änderungen einer Software, um verschiedene Änderungen zusammenzufügen oder zu einer vorherigen Revision der Software zurückzukehren.<sup>79 80</sup>

Bei der Entwicklung von Movelets ist es möglich, dass verschiedene Entwickler und Übersetzer parallel beteiligt sind. Infolgedessen entstehen Änderungen des Quelltexts und des Inhalts der XLIFF Dateien. Diese Änderungen müssen zusammengefügt werden. Möglicherweise werden auch Änderungen vorgenommen, welche sich als unpassend herausstellen. Daher müssen diese Änderungen zurückgesetzt werden, man spricht von der Rückkehr zu einer früheren Revision.

Des Weiteren ist es möglich, dass verschiedene Lokalisierungen eines Movelets gleichzeitig veröffentlicht werden müssen. Aus diesem Grund ist die parallele Entwicklung und Lokalisierung des Movelets notwendig. Folglich, müssen zum Zweck der parallelen Übersetzung die zu übersetzenden Strings aus dem Quelltext mehrfach extrahiert werden. Deshalb müssen bestehende XLIFF Dateien erweitert werden. Um Redundanz zu vermeiden müssen nach Änderungen des Quelltexts ausschließlich neu hinzugefügte zu übersetzende Strings extrahiert werden und bereits extrahierte, nicht mehr verwendete entfernt werden.

Im Kontext der Movelet Entwicklung wird *GIT* zur Versionsverwaltung eingesetzt. Werden die XLIFF Dateien mit ins *GIT Repository* aufgenommen, so übernimmt *GIT* die Aufgabe der Versionsverwaltung für das Lokalisierungstool. Mit einer Anfrage an *GIT* kann das Lokalisierungstool die Information erhalten, welche zu übersetzenden Strings neu hinzugefügt, beziehungsweise entfernt wurden. In der Konsequenz muss das Lokalisierungstool keine eigene Versionsverwaltung implementieren.

---

<sup>79</sup>Vgl. Sommerville 2007, S. 698ff.

<sup>80</sup>Vgl. Pressman 2005, S. 751ff.

## 10 Fazit

Zusammenfassung  
keine englische  
Version

Ziel dieser Arbeit ist die Konzeption einer Software, welche zu übersetzende und übersetzte Strings der Texte eines Movelets automatisiert verarbeitet.

Im Folgenden wird das im Zuge dieser Arbeit entstandene Konzept eines Lokalisierungstools vorgestellt. Um die Verwendung des Lokalisierungstools zu ermöglichen werden zu übersetzende Strings vom Nutzer ausgezeichnet. Zu diesem Zweck wird ITS und die an *GNU gettext* angelehnte Auszeichnung `_()` verwendet. Das Lokalisierungstool nutzt die Auszeichnung um zu übersetzende von nicht zu übersetzenden Strings zu unterscheiden. Infolgedessen extrahiert das Lokalisierungstool die ausgezeichneten Strings und erstellt XLIFF Dateien. Die extrahierten Strings werden in den *source* Elementen der XLIFF Dateien gespeichert. XLIFF wird als Schnittstelle zu externer Software verwendet. In der Konsequenz kann dieses Format ohne Kompatibilitätsprobleme von in- und/oder externen Übersetzern und deren Software übersetzt werden. Die übersetzten Strings werden in den *source* Elementen zugehörigen *target* Elementen der XLIFF Dateien gespeichert. Aus diesem Grund enthalten die *source* Elemente die zu übersetzenden und die *target* Elemente die zugehörigen übersetzten Strings. Aufgrund der Extraktion stimmt der Inhalt eines *source* Elements mit einem ausgezeichneten String des Quelltexts des Movelets überein. Folglich wird durch das Ersetzen der ausgezeichneten Strings durch die Strings in den zugehörigen *target* Elementen eine übersetzte Version des Quelltexts erzeugt. Zu diesem Zweck werden nicht die Strings im Originalquelltext ersetzt, stattdessen wird eine Kopie erzeugt, in welcher die Ersetzung erfolgt. Resultierend daraus fungiert der Originalquelltext als Vorlage, aus welcher beliebig viele übersetzte Versionen erzeugt werden können.

In Kapitel 9 wurde erarbeitet, dass dieses Lokalisierungstool keine Versionsverwaltung implementiert. Es ist jedoch empfehlenswert Software zur Versionsverwaltung einzusetzen.

Zum Zweck des besseren Verständnisses ist die Funktion des Lokalisierungstools in Abbildung 10.1 veranschaulicht. Die rote Linie zwischen Movelet-Vorlage und XLIFF Datei stellt die Extraktion der zu übersetzenden Strings dar. Die Linien zwischen Übersetzer und XLIFF Datei stellen den Vorgang der Übersetzung dar und die grüne Linie zwischen XLIFF Datei und dem lokalisierten Movelet-EN stellt das Ersetzen der zu übersetzenden Strings in der Vorlage durch die übersetzten Strings dar.

Das erarbeitete Lokalisierungstool greift bestehende Konzepte auf. Zu diesen Konzepten zählt die Auszeichnung von zu übersetzenden Strings, ITS, XLIFF und das Verwenden des Namens als Bezeichner. Die Auszeichnung von zu übersetzenden Strings,

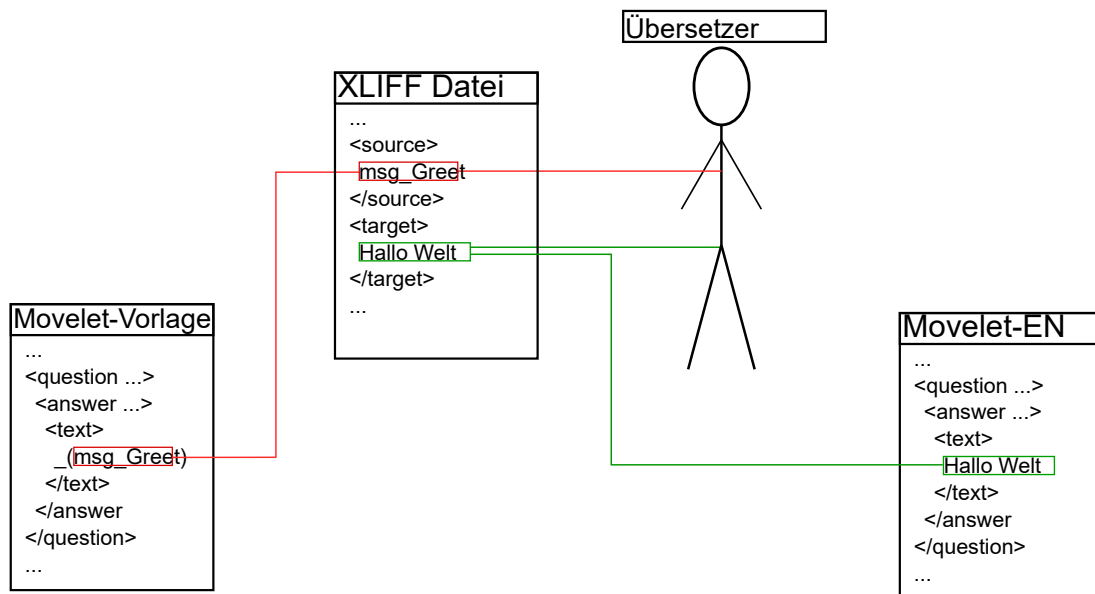


Abbildung 10.1: Darstellung der Arbeitsschritte des Lokalisierungstools

um diese von anderen Strings zu unterscheiden, wird von verschiedenen Lokalisierungstools verwendet. XLIFF wurde von OASIS für den Austausch von Daten der Lokalisierung entwickelt. Den Namen als Bezeichner zu verwenden ist ein grundlegendes Konzept der Bezeichnung.

Das Alleinstellungsmerkmal des erarbeiteten Lokalisierungstools ist seine Einsatzmöglichkeit und Anpassung im Kontext von Movelets. Daher ist es die erste Software, die zum Zweck der Lokalisierung von Movelets eingesetzt werden kann. Aufgrund dessen hat dieses Lokalisierungstool das Potential den Prozess der Lokalisierung von Movelets zu revolutionieren.

# 11 Diskussion

Im Verlauf dieser Arbeit wurden die Anforderungen an das Lokalisierungstool anhand des Ziels dieser Arbeit formuliert. Für diese Anforderungen wurden Lösungsansätze anhand der Literatur recherchiert und diskutiert. Infolgedessen lassen sich Anforderungen und Lösungsansätze anhand der angegebenen Literatur nachvollziehen. Die Entscheidung für den jeweiligen Lösungsansatz wurde anhand einer Diskussion belegt.

Ziel dieser Arbeit ist die Konzeption einer Software, welche zu übersetzende und übersetzte Strings der Texte eines Movelets automatisiert verarbeitet. Das erarbeitete Lokalisierungstool automatisiert die Extraktion ausgezeichneter Strings und das Ersetzen dieser durch deren Übersetzungen. Auf diese Weise wird ein hoher Grad der Automatisierung erzielt. Die Auszeichnung und Übersetzung der zu übersetzenden Strings erfolgt jedoch manuell und bietet folglich weitere Möglichkeit der Automatisierung. Resultierend daraus ist dieses Lokalisierungstool der Anfang einer Automatisierung der Lokalisierung von Movelets. Des Weiteren sind zum Zweck der Implementierung und Verwendung des Lokalisierungstools weitere Aspekte zu beachten. Diese sind die Konzeption der Benutzeroberfläche des Lokalisierungstools, die Auswahl der Übersetzer, beziehungsweise der Übersetzungssoftware und eine Dateistruktur für generierten XLIFF Dateien. Zur Konzeption der Benutzeroberfläche zählt insbesondere das Auflisten und Behandeln von Kollisionen.

Der Umfang dieser Arbeit ist aufgrund des von Movilizer vorgegebenen Zieles und dem vorgesehenen Umfang einer Projektarbeit begrenzt. Die Lokalisierung selbst jedoch umfasst weit mehr Aspekte. Zu diesen zählen.

- Übersetzung
- Verwendung geeigneter Zeichensätze
- Übersetzung verknüpfter Strings
- Anpassung von Grafiken und Symbolen
- Anpassen der Benutzeroberfläche an Textlängen und Flussrichtungen
- Anpassung an geltendes Recht
- Anpassung der Formate für Adresse, Datum et cetera
- Anpassung der Maßeinheiten für Gewicht, Währung, et cetera
- Anpassung der Sortierung und Suche

In der Konsequenz bietet die Lokalisierung viel weiteres Automatisierungspotential für Movilizer. Bezogen auf dieses Lokalisierungstool selbst lassen sich weitere Forschungsspekte empfehlen. Hierzu zählt die automatisierte Übersetzung der generierten XLIFF Dateien. Anstelle von einer automatisierten Übersetzung bietet sich die Konzeption der Möglichkeit einer Übersetzung in der Benutzeroberfläche des Movelets selbst an. Dies ermöglicht die Verfügbarkeit aller Kontextinformationen bezüglich der Übersetzung. Des Weiteren ist es wissenswert, wie der Einsatz der in den XLIFF Dateien gespeicherten Übersetzungen zum Aufbau einer Terminologiedatenbank oder eines TM möglich ist, welche die Übersetzungskosten senken und die Übersetzungsqualität steigern.

# Literaturverzeichnis

- Asnes, Adam (2010). *Internationalization Basics for Non-Engineers*. URL: <https://www.gala-global.org/ondemand/internationalization-basics-non-engineers>.
- Barrachina, Sergio et al. (2009). „Statistical Approaches to Computer-Assisted Translation“. In: *Computational Linguistics* 35.1, S. 3–28. ISSN: 0891-2017. DOI: 10.1162/coli.2008.07-055-R2-06-29.
- Berners-Lee, Fielding und Masinter (2005). *Uniform Resource Identifier (URI) Generic Syntax*. Hrsg. von IETF. URL: <https://www.ietf.org/rfc/rfc3986.txt>.
- Bray, Tim et al. (2008). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Hrsg. von W3C. URL: <https://www.w3.org/TR/xml/>.
- Crocker, D. und P. Overell (2008). *Augmented BNF for Syntax Specifications: ABNF*. Hrsg. von IETF.
- De Mula, Jesús und Paul Janzen (2018). *Movilizer Gradle Plugin: For Movelet Development*. Mannheim. URL: <https://honeywellprod.sharepoint.com/teams/MovilizerTraining/Movilizer%20Knowledge%20Transfer/Sessions/2018-08%20Movilizer%20Gradle%20PlugIn/2018-08-24%20Gradle%20PlugIn.mp4?csf=1&e=unAxzt>.
- Ellison, Paul (1994). „Introduction to SGML Concepts“. In: URL: <https://ieeexplore.ieee.org/document/369727>.
- Filip, David et al. (2013). *Internationalization Tag Set (ITS) Version 2.0*. Hrsg. von W3C. URL: <https://www.w3.org/TR/its20/#overview>.
- GALA, Hrsg. (2008). *Systems to manage terminology, knowledge, and content - Term-Base eXchange (TBX)*. URL: [https://www.gala-global.org/sites/default/files/uploads/pdfs/tbx\\_oscar\\_0.pdf](https://www.gala-global.org/sites/default/files/uploads/pdfs/tbx_oscar_0.pdf).
- GNU, Hrsg. (o.D.). *GNU gettext utilities*. URL: <https://www.gnu.org/software/gettext/manual/gettext.html#Aspects>.
- Hassell-Corbiell, R. (2001). „Developing Globally Correct Content for International Audiences“. In: *IPCC 2001*. Piscataway, NJ: IEEE, S. 421–424. ISBN: 0-7803-7209-3. DOI: 10.1109/IPCC.2001.971590.



Hoda, M. N. (2015). *2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015: New Delhi, India, 11 - 13 March 2015*. Piscataway, NJ: IEEE. ISBN: 978-9-3805-4416-8. URL: <http://ieeexplore.ieee.org/servlet/opac?punumber=7088896>.

ISO, Hrsg. (2006). *ISO/IEC 19757-4:2006: Information Technology - Document Schema Definition Languages (DSDL) - Part 4 Namespace-based Validation Dispatching Language (NVDL)*. URL: <https://www.iso.org/standard/38615.html>.

Korenjak, A. J. (1969). „A Practical Method for Constructing LR (K) Processors“. In: *Communications of the ACM* 12.11, S. 613–623. ISSN: 00010782. DOI: 10.1145/363269.363281. URL: <http://doi.acm.org.ezproxy-dhma-2.redi-bw.de/10.1145/363269.363281>.

Leiva, Luis A. und Vicent Alabau (2015). „Automatic Internationalization for Just In Time Localization of Web-Based User Interfaces“. In: *ACM Transactions on Computer-Human Interaction* 22.3, S. 1–32. ISSN: 10730516. DOI: 10.1145/2701422.

Mauro, Pancrazio de (1999). „Internationalizing Messages in Linux Programs“. In: *Linux J* 1999.59es. ISSN: 1075-3583. URL: <http://dl.acm.org/citation.cfm?id=327697.327701>.

McCormick, M., Susan (2000). „Exchanging Lexical and Terminological Data with OLIF2“. In: *Translating and the Computer 22: Proceedings of the Twenty-second international conference...16-17 November 2000. (London: Aslib, 2000)*. URL: <http://www.mt-archive.info/00/Aslib-2000-McCormick.pdf>.

Nitschkowski, Sabrina (2015a). *Introduction to MEL execution*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/DOC21DRAFT/Introduction++to+MEL+execution>.

Nitschkowski, Sabrina (2015b). *Movilizer XML reference*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/DOC24/Movilizer+XML+reference>.

Nitschkowski, Sabrina (2016a). *Epsilon screen*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/DOC21/Epsilon+screen>.

Nitschkowski, Sabrina (2016b). *masterdata element*. Hrsg. von Movilizer. <https://devtools.movilizer.com/confluence/display/DOC21/masterdata+element>.

Nitschkowski, Sabrina (2016c). *text element*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/DOC23/text+element>.

Nitschkowski, Sabrina (2018a). *concat methode*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/DOC26/concat+method>.

- Nitschkowski, Sabrina (2018b). *Introduction to master data*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/Introduction+to+master+data>.
- Nitschkowski, Sabrina (2018c). *Introduction to the Movilizer*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/Introduction+to+the+Movilizer>.
- Nitschkowski, Sabrina (2018d). *Introduction to the Movilizer Expression Language*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/Introduction+to+the+Movilizer+Expression+Language>.
- Nitschkowski, Sabrina (2018e). *MEL methods*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/MEL+methods>.
- Nitschkowski, Sabrina (2018f). *Message screen*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/Message+screen>.
- Nitschkowski, Sabrina (2018g). *setAnswerValueByClientKey method*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/setAnswerValueByClientKey+method>.
- Nitschkowski, Sabrina (2018h). *String MEL values*. Hrsg. von Movilizer. URL: <https://devtools.movilizer.com/confluence/display/D0C26/String+MEL+values>.
- O'Brien, Sharon, Minako O'Hagan und Marian Flanagan (2010). „Keeping an eye on the UI design of Translation Memory“. In: *Proceedings of the 28th Annual European Conference on Cognitive Ergonomics - ECCE '10*. Hrsg. von Mark Neerinx und Willem-Paul Brinkman. New York, New York, USA: ACM Press, S. 187. ISBN: 9781605589466. DOI: 10.1145/1962300.1962338.
- Pooley, David und Raya M. Rodolfo (2008). *SRX 2.0 Specification*. Hrsg. von GALA. URL: <https://www.gala-global.org/srx-20-april-7-2008>.
- Presser, Leon und John R. White (1972). „Linkers and Loaders“. In: *ACM Computing Surveys* 4.3, S. 149–167. ISSN: 03600300. DOI: 10.1145/356603.356605.
- Pressman, Roger S. (2005). *Software engineering: A practitioner's approach*. 6th ed. Boston, Mass: McGraw-Hill. ISBN: 0-07-285318-2.
- Reineke, Detlef und Klaus-Dirk Schmitz, Hrsg. (2005). *Einführung in die Software-lokalisierung*. Tübingen: gnv Narr. ISBN: 3823361562. URL: <http://www.iim.fh-koeln.de/lion2005>.
- Reiner, Ulrike (1991). *Anfragesprachen für Informationssysteme*. Hrsg. von Deutsche Gesellschaft für Dokumentation. Frankfurt am Main.

- Reynolds, Peter et al. (2006). *Translation Web Services Specification 1.0*. Hrsg. von OASIS. URL: <https://www.oasis-open.org/committees/download.php/19160/trans-ws-spec-1.0.2.pdf>.
- Savourel, Yves (2005). *TMX 1.4b*. Hrsg. von GALA. URL: <https://www.gala-global.org/tmx-14b>.
- Schmitz, Klaus-Dirk und Kirsten Wahle, Hrsg. (2000). *Softwarelokalisierung*. Stauffenburg-Handbücher. Tübingen: Stauffenburg-Verl. ISBN: 3860570714.
- Schnabel, Bryan et al. (2014). *XLIFF Version 2.0*. Hrsg. von OASIS. URL: <https://docs.oasis-open.org/xliff/xliff-core/v2.0/xliff-core-v2.0.html>.
- Sintoris, Konstantinos und Kostas Vergidis (2017). „Extracting Business Process Models Using Natural Language Processing (NLP) Techniques“. In: *2017 IEEE 19th Conference on Business Informatics (CBI)*. IEEE, S. 135–139. ISBN: 978-1-5386-3035-8. DOI: 10.1109/CBI.2017.41.
- Sommerville, Ian (2007). *Software engineering*. 8. ed. International computer science series. Harlow: Addison-Wesley. ISBN: 0321313798.
- Suresh, Aparna und A.K George (2018). „Performance Analysis of Various Combination Sorting Algorithms for Large Dataset to fit to a Multi-Core Architecture“. In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, S. 51–56. ISBN: 978-1-5386-1974-2. DOI: 10.1109/ICICCT.2018.8472956.
- Tykhomyrov, Olexiy Ye (2002). „Introduction to Internationalization Programming“. In: *Linux J* 2002.103, S. 3–. ISSN: 1075-3583. URL: <http://dl.acm.org.ezproxy-dhma-1.redi-bw.de/citation.cfm?id=583910.583913>.
- Unger, Stephen H. (1968). „A Global Parser for Context-free Phrase Structure Grammars“. In: *Communications of the ACM* 11.4, S. 240–247. ISSN: 00010782. DOI: 10.1145/362991.363001. URL: <http://doi.acm.org.ezproxy-dhma-2.redi-bw.de/10.1145/362991.363001>.
- Wagner, Sascha (2017). *Think Global – Act Local: Make use of i18n and l10n*. Hrsg. von GALA. URL: <https://www.gala-global.org/blog/think-global-%E2%80%93-act-local-make-use-i18n-and-l10n>.
- Wang, Xiaoyin et al. (2009). „TranStrL: An Automatic Need-to-translate String Locator for Software Internationalization“. In: *Proceedings of the 31st International Conference on Software Engineering*. ICSE '09. Washington, DC, USA: IEEE Computer Society, S. 555–558. ISBN: 978-1-4244-3453-4. DOI: 10.1109/ICSE.2009.5070554. URL: [http://dx.doi.org/10.1109/ICSE.2009.5070554,%20\[Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen\]](http://dx.doi.org/10.1109/ICSE.2009.5070554,%20[Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen]).

- Willenborg, Josef (2001). „Anfragesprachen für Internet-Informationssysteme“. Dissertation. URL: <http://www.josef-willenborg.de/publications/dissertation.pdf>.
- Yuji, Yamamoto et al. (2018). *UTX 1.20 Specification*. Hrsg. von AAMT. URL: <http://www.aamt.info/english/utx/utx1.20-specification-e.pdf>.
- Zydroń, Andrzej, Raya M. Rodolfo und Bartosz Bogacki (2007). *XML:TM 1.0*. Hrsg. von GALA. URL: <https://www.gala-global.org/xmltm-10>.
- Zydroń, Andrzej und Saldana Derek (2009). *Reference Model for Open Architecture for XML Authoring and Localization 1.0*. Hrsg. von OASIS. URL: [https://www.oasis-open.org/committees/download.php/35736/OASIS%20Open%20Architecture%20for%20XML%20Authoring%20and%20Localization%20Reference%20Model%20\(OAXAL\).pdf](https://www.oasis-open.org/committees/download.php/35736/OASIS%20Open%20Architecture%20for%20XML%20Authoring%20and%20Localization%20Reference%20Model%20(OAXAL).pdf).
- Zydroń, Andrzej, David Walters und Raya M. Rodolfo (2007). *GMX-V 1.0: Global Information Management Metrics Volume (GMX-V) 1.0 Specification*. Hrsg. von GALA. URL: <https://www.gala-global.org/gmx-v-10>.

# A Anhang

## A.1 Implementierung der Extraktion zu übersetzender Strings anhand einer Auszeichnung

Zu dem Zweck der Implementierung eines Lokalisierungstools, welches die ausgezeichneten Strings extrahiert, ist für ITS ein Namespace-based Validation Dispatching Language<sup>81</sup> Dokument gegeben.<sup>82</sup> Die Auszeichnung `__()` lässt sich durch die Produktion einer kontextfreien Grammatik wie in A.1 dargestellt definieren. Die dargestellte Produktion ist Teil einer kontextfreien Grammatik, welche auch eine reguläre Grammatik ist, sie wird in der Augmented Backus-Naur Form<sup>83</sup> dargestellt.

$$\begin{aligned} Q &= \text{""}' | \text{''}'' \\ WS &= (\epsilon | WSP | CR | LF)^* \\ STRING &= (VCHAR | DIGIT | WS)^* \\ MARKUP0 &= \text{""\_(" WS STRING WS ")} \\ &\quad ; 0 \text{ steht für alle ohne ' ausgezeichneten Strings} \\ MARKUP1 &= \text{""\_(" WS Q STRING Q WS ")} \\ &\quad ; 1 \text{ steht für alle mit ' ausgezeichneten Strings} \\ MARKUP2 &= \text{""\_(" WS "concat(" WS (CALL} \\ &\quad | Q STRING Q) (WS \text{""}, \text{"" WS (CALL} \\ &\quad | Q STRING Q) )^+ WS \text{""})" WS ")} \\ CALL &= CHAR STRING^* \\ &\quad ; \text{Aufruf von Variable oder Methode} \end{aligned} \tag{A.1}$$

---

<sup>81</sup>Vgl. ISO 2006.

<sup>82</sup>Vgl. Filip et al. 2013.

<sup>83</sup>Vgl. Crocker und Overell 2008.

# Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Konzeptionelle Entwicklung eines Lokalisierungstools für Movelets* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum

Fabian Wolf