Machbarkeitsnachweis (POC) für einen DSL basierten Formulargenerator

Josef Rossa

HTW Berlin

s0537867@htw-berlin.de

2. Februar 2014

Zusammenfassung

Diese Arbeit untersucht inwieweit sich eine domänenspezifische Sprache (DSL) als Formulargenerator eignet. Dazu wurde ein *alpha* Prototyp auf der Basis von *Xtext* entwickelt. Der Prototyp erlaubt die Übersetzung eines beschriebenen Formulars in eine *javascript* und *jquery* basierte Webseite. Sämtliche Sprachelemente werden dabei berücksichtigt.
Der Prototyp bestätigt, dass ein DSL basierter Formulargenerator Potential hat. Entwickler finden sich intuitiv in der Sprache zurecht und arbeiten durch den automatisch generierten *Eclipse* Editor sehr effizient. Die Lesbarkeit ist deutlich besser als bei anderen Ansätzen. Auch wenn im Rahmen dieser Arbeit nur ein Übersetzer in eine Zieltechnologie entstanden ist, lässt sich ohne weiteres ein neuer Übersetzer integrieren. Der Ansatz ist deshalb besonders für die *Cross Plattform* Formulargenerierung geeignet.

1. Einleitung

Formulare sind in unserem Leben omnipräsent. Ob Steuererklärungen, Marktumfragen, Anamnesebögen, Feedbackbögen oder Wissensquiz. In vielen Lebensbereichen werden Daten durch Fragebögen erfasst. Die Erstellung dieser Fragebögen ist aufwendig. Neben dem konzeptionellen Teil, stellt die Umsetzung bei digitalen Fragebögen einen beträchtlichen Aufwand dar. Dies gilt insbesondere bei der Bedienung mehrerer Zielplattformen. Soll beispielsweise eine Marktumfrage als *Android App*, *iOS App* und als Webseite bereitgestellt werden, erfordert das Kompetenzen in dreierlei Technologien. Außerdem muss ein und das selbe Produkt drei Mal entwickelt werden.

Nun gibt es verschiedene Ansätze, welche die aufgeführten Probleme adressieren. So sind unter dem Begriff „*Cross Plattform Development*“ eine Vielzahl an Werkzeugen entstanden, die es ermöglichen, durch einmal geschriebenen Code mehrere Zielplattformen zu bedienen.

Daneben existieren mehrere Online Plattformen die, auf der Basis grafischer Editoren, die Erstellung und Bereitstellung von Fragebögen erlauben. Dabei gibt es neben den Plattformen von kleineren Unternehmen[1] auch Angebote von großen Unternehmen wie google[2].

Hinzu kommen mit hoher Wahrscheinlichkeit

[1] bspw.: `https://www.umfrageonline.com/`

[2] siehe `http://www.google.com/google-d-s/createforms.html`

unzählige proprietäre Lösungen von Unternehmen unterschiedlicher Branchen.

Alle aufgezählten Lösungsansätze haben ihre Vor- und Nachteile. Während die *Cross Plattform Tools* einen überwiegend zu generellen Einsatzzweck verfolgen, sind die Online Plattformen zu sehr einer Technologie verschrieben.

Diese Arbeit hat sich der Evaluation eines andersartigen Ansatzes verschrieben. Die Fragebögen werden dabei in einer domänenspezifischen Sprache (DSL) modelliert. Aus dem so generierten Modell des Fragebogens lassen sich anschließend Exemplare beliebiger Zieltechnologien ableiten. Die Evaluation erfolgt am Beispiel eines dafür entwickelten *alpha* Prototypen.

Die Arbeit ist wie wie folgt aufgebaut. Abschnitt 2 erläutert in Kürze, welche Vorteile domänenspezifische Sprachen haben. In Abschnitt 3 sind die Anforderungen an den Prototypen dargestellt. Daraus geht auch der grobe Aufbau des Systems hervor. Der darauf folgende Abschnitt geht auf die Umsetzung des Systems ein. Im letzten Abschnitt wird der Ansatz anhand der gemachten Erfahrungen mit dem Prototypen bewertet.

2. Potential domänenspezifischer Sprachen

Um Fragebögen zu modellieren bedarf es nicht unbedingt einer domänenspezifischen Sprache. Jede andere Technologie zur Modellierung lässt sich dazu verwenden (bspw. *XML* oder *UML*). Es stellt sich also die Frage, weshalb die Erstellung einer domänenspezifischen Sprache sinnvoll ist.
Während bei *XML* die Notation nicht anpassbar ist, hat *UML* den entscheidenden Nachteil, dass sie nicht domänenspezifisch ist. Das selbe Problem hat man auch bei der Modellierung von Fragebögen in Datenbanken. Generelle Programmiersprachen, sogenannte „*general purpose languages*“ (GPL) sind ebenfalls zu allgemein gehalten (vgl. [1]).
Domänenspezifische Sprachen haben zudem den Vorteil, dass sich die Entwicklungsumgebung ganz auf die Bedürfnisse der Domäne anpassen lässt. So lassen sich beispielsweise logisch inkorrekte Deklarationen in der Sprache durch benutzerdefinierte Validatoren ausschließen.
Dies als ein kleiner Auszug der Vorteile domänenspezifischer Sprachen, welche dazu beitragen, dass diese sich „wachsender Beliebtheit“ (siehe [2]) erfreuen, wie aus einer aktuellen Einschätzung hervorgeht. Für weitere Erläuterungen zu domänenspezifischen Sprachen sei der Leser auch auf [3] verwiesen.

3. Anforderungen an den Prototypen

Der Prototyp soll verdeutlichen wie sich ein Formulargenerator auf der Basis einer domänenspezifischen Sprache umsetzen lässt. Einige Entwicklungsschritte sind dabei klar voneinander zu trennen.

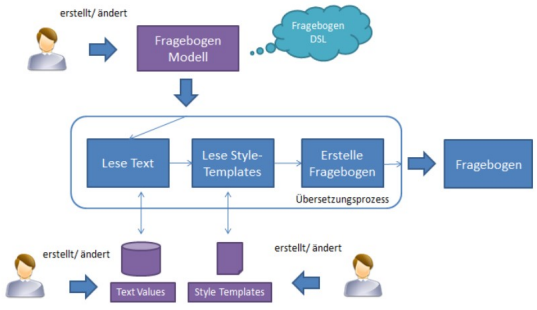

Abbildung 1: Systemaufbau

Abbildung 1 zeigt den Aufbau des Systems. Darin ist erkennbar, dass die tatsächliche Beschreibung der Fragebögen vom Design getrennt ist. Ebenfalls eigenständig ist die Übersetzung, bzw. die Bereitstellung der Fragebogeninhalte in verschiedenen Sprachen. Wie im oberen Teil der Grafik

dargestellt, modelliert der Entwickler den Fragebogen in der domänenspezifischen Sprache. In einem repräsentativen Übersetzungsprozess wird der Fragebogen dann mit Informationen angereichert. Diese sind: zunächst die tatsächlichen textuellen Werte der Variablen in dem Fragebogen Modell. Anschließend wird zu dem Modell eine Designvorlage geladen. Mit diesen Informationen bestückt, wird das Modell dann in die jeweilige Zieltechnologie übersetzt.

Über den Erfolg oder Misserfolg des Systems entscheidet aber in erster Linie die Beschaffenheit der Sprache. Zum Aufbau der Sprache gibt es kein Patentrezept. Alles liegt in der Hand des „Schöpfers“ und so lautet die Anforderung diesbezüglich, dass die Sprache „gefallen muss“. In einem ausführlichen Artikel der Zeitschrift „Programmieren heute“ sind viele Aspekte der Sprachentwicklung thematisiert. Der Autor spekuliert darin, dass der Einsatz gewohnter Sprachelemente Erfolg verspricht (z.B. Blöcke in geschweiften Klammern) (vgl. [4]).

4. Umsetzung des Prototypen

Die erstellte Fragebogen DSL basiert auf *Xtext*. Dabei handelt es sich um ein Sprachen Entwicklungsframework (vgl. [5]). *Xtext* selbst beinhaltet eine DSL, in der sich die Syntax und der Aufbau der eigenen Sprache beschreiben lässt. In dieser Sprache ist die Fragebogen DSL modelliert. Erzeugt man die Sprache aus diesem Modell, generiert *Xtext* auch gleich einen Editor zur Sprache, syntaktische Validierung und vieles mehr. Ein Fragebogen lässt sich dann komfortabel in der eigenen Sprache spezifizieren.

```
Catalog SurveyOnTechnicalDevicesUse {

    Page firstPage {
        + computerUseQuestion -> computerUseQAL
        + chooseUsedDevices -> chooseDevicesQAL
        + specifyOtherDevices -> otherDevicesSpecificationQAL
    }

    QuestionAnswerLink computerUseQAL {
        Question computerUseQuestion {
            text : "Do you use computer devices?"
        }
        Answer computerUseAnswer {
            + yesNoSelectionAssignment -> SingleSelectionGroup yesNoSelection {
                type : SELECT
                SelectionElement yesSelectionElement {
                    text : "yes"
                }
                SelectionElement noSelectionElement {
                    text : "no"
                }
            }
        }
    }
```

Abbildung 2: Auszug aus der Formular DSL

Einen Auszug aus einer solchen Spezifikation zeigt Abbildung 2. Zu sehen ist darin ein umschließendes Container Element (*Catalog*). Innerhalb dieses Elements werden die wichtigsten Fragebogenbestandteile definiert. Diese sind als Bestandteil dieses Prototypen:

- **Answer**: Containerelement für jede Art von Antwort
- **Dependency**: Definiert Abhängigkeiten im Zielfragebogen. Die Anzeige von Elementen kann abhängig sein von beliebig komplexen Antwortmustern anderer Elemente.
- **FileDataProvider**: Parametrisiert Texte im Fragebogen (wird zur Übersetzungszeit aufgelöst)
- **MultiSelectionGroup**: Antwortelement, dass die Auswahl von n Optionen aus m Optionen zulässt (wobei n <= m)
- **Page**: Definiert eine Seite bzw. ein Gruppenelement
- **Polarity**: Antwortelement, dass die Auswahl im Sinne einer Abstufung interpretiert (z.B. Schulnoten)
- **Question**: Containerelement für die Frage
- **QuestionAnswerGroup**: Element zur Gruppierung von Frage/Antwort Verknüpfungen
- **QuestionAnswerLink**: Verknüpft Fragen mit Antworten
- **SingleSelectionGroup**: Antwortelement, dass die Auswahl von einer Option aus m Optionen zulässt
- **Textfield**: Antwortelement mit textuellem Inhalt
- **Variable**: Variable die in Abhängigkeiten (*Dependencies*) als Restriktion verwendet werden kann

In Abbildung 2 erkennt man demnach die Beschreibung einer Seite, der jeweils unter einem Bezeichner drei Seitenelemente zugewiesen sind. Eines dieser zugewiesenen Elemente identifiziert man darunter als Frage-Antwort Verknüpfung. Die Frage ist selbst erklärend. Die Antwort beinhaltet als Antwortelement eine einfache Auswahl aus mehreren Optionen. Es erschließt sich, dass die Antwort als eine Ja/Nein Antwort beschrieben ist. Die Grafik beinhaltet ein Beispiel in kompakter Schreibweise. Es ist alternativ möglich die Elemente unabhängig voneinander zu definieren (auch auf mehrere Dateien verteilt) und sie dann in den Containern zu referenzieren. Dazu muss die Namensgebung der Elemente eindeutig sein. Was im Fall des Prototypen gegeben ist.

Die Trennung von Entwicklung und Design wird durch eine *Template Sprache* realisiert. Darin ist zu jedem Element der Fragebogen DSL, das visuelle Relevanz hat, ein *Style Template* definiert.

```
Question{
    <p style="font-weight:bold" %Name%>%Text%</p>
}Question
```

Abbildung 3: Beispiel für ein *Style Template* zum Frage Element der Formular DSL

Ein Beispiel zeigt Abbildung 3. Zu dem *Question* Element der Fragebogen DSL wird innerhalb der geschweiften Klammern festgelegt, wie das Element übersetzt wird. Die *Template Sprache* erlaubt darin sämtliche Elemente die auch in auf *html* Seiten vorkommen können. Im Prinzip ist die *Template Sprache* nichts anderes als eine Webseite die man beliebig designen kann. Lediglich die Elemente der Formular DSL müssen enthalten sein. Sind diese nicht enthalten schlägt die Übersetzung fehl.

Im Rahmen des Prototypen ist ein exemplarischer Übersetzer entstanden. Dieser kann eine beliebige Fragebogenbeschreibung der Formular DSL in eine *javascript* und *jquery* basierte Webseite übersetzten. Beim Übersetzungsvorgang werden sämtliche Elemente der Sprache berücksichtigt. Beispielsweise werden Abhängigkeiten (*Dependencies*) in der Zieltechnologie ad hoc aufgelöst.

Abbildung 4: Abhängigkeit aufgelöst in der generierten Webseite

Dies ist in Abbildung 4 visualisiert. In der DSL ist eine Abhängigkeit der zweiten Frage von Frage eins definiert. Deshalb erscheint Frage zwei nur, wenn in Frage eins eine entsprechende Auswahl getroffen ist. Dieses Prinzip funktioniert auch, wenn die Elemente unterschiedlichen Seiten zugewiesen sind.

Zu Demonstrationszwecken deutet der Prototyp auch die Möglichkeiten der benutzerdefinierten (logischen) Validierung an. So erzeugt der Editor einen Fehler, wenn in einer Abhängigkeit ein Element von der Antwort in einem anderen Element abhängig ist, welches erst nach dem ersteren Element in der Fragen Reihenfolge erscheint. Dass durch die geschickte Einschränkung des *Scope* ein höherer Schreibkomfort entsteht, demonstriert der Prototyp ebenfalls. So wird bei der Referenzierung von Elementen in den Abhängigkeiten der *Scope* auf Elemente beschränkt die nach dem Kopf des Referenzpfads „sichtbar“sind.

Im zurückliegenden Abschnitt wurden die wichtigsten Bestandteile des Systems und deren Umsetzung vorgestellt. Im nächsten Abschnitt folgt eine Bewertung des DSL basierten Ansatzes.

5. Bewertung und Ausblick

Im Rahmen dieser Arbeit wurde untersucht inwieweit sich eine domänenspezifische Sprache zur Definition von Formularen eignet. Dabei ist ein Prototyp entstanden der zeigt wie ein DSL basierter Formulargenerator aufgebaut sein kann.
Die Umsetzung genügt den Anforderungen an den Prototypen. Die Trennung von Design, Übersetzung und Formularmodellierung hat sich als sinnvoll erwiesen. Eine Verbesserungsmöglichkeit liegt in der Schaffung einer Möglichkeit zum Design einzelner Elementinstanzen. In der aktuellen Realisierung ist das Designen nur auf Elementebene möglich.

Wie bereits erwähnt gibt es kein Patentrezept für eine erfolgreiche Sprache (vgl. [4]). Aus diesem Grund gibt es auch so viele verschiedene Programmiersprachen (auch gleicher Paradigmen). Jeder Entwickler hat eben eigene Vorlieben und diese zu bedienen ist die Schwierigkeit im Zusammenhang mit der Entwicklung von Sprachen.

Tests anhand des Prototypen haben die These bekräftigt, dass sich der DSL basierte Ansatz für die Generierung von Formularen eignet. Softwareentwickler finden sich intuitiv in der Sprache zurecht. *Xtext* erzeugt einen *Eclipse* Editor zur DSL. Das bedeutet, dass Entwickler die im Umgang mit Eclipse geübt sind deren Features sofort auch auf die neue Sprache anzuwenden wissen. Durch die Kombination von *code completion* , sinnvollem *scoping* und aussagekräftigen reservierten Wörtern kann ein Fragebogen definiert werden ohne jemals zuvor von der Sprache gehört zu haben.

Eine Fragebogen Deklaration ist nicht nur von Entwicklern intuitiv lesbar. Hier hat die Technologie Vorteile gegenüber anderen Technologien zur Modellierung. Während man bei *XML* und *UML* doch wenigsten einige Zeit braucht um die Systematik zu verstehen, kann eine DSL so aufgebaut sein, dass sie einem Großteil der Menschen ohne Einarbeitung zugänglich ist.
Für normale Anwender mag eine grafische Oberfläche zur Erstellung von Fragebogen besser geeignet sein. Doch kann auch auf der Grundlage der DSL ein grafisches Interface geschaffen werden. In diesem Fall besteht der Vorteil der DSL gegenüber anderen Ansätzen in der Beschaffenheit der Metaebene.

Da die Erfahrungen mit dem Prototypen überwiegend positiv ausfallen, wird dieser in Zukunft weiter ausgebaut. Primär werden dabei weitere Übersetzer entstehen. Beispielsweise in eine *Android App* oder in eine *iOS App*. Außerdem wird der Sprache ein Element hinzugefügt, dass die Definition von Datenkonsumenten erlaubt. Diese sind für die Generierung von auswertenden Services vorgesehen.

Literatur

[1] `http://www.heise.de/developer/artikel/Was-sind-DSLs-228030.html` Zugriff: 23.01.2014 13:08 Uhr

[2] `http://www.heise.de/developer/artikel/Neues-Buch-zu-domaenenspezifischen-Sprachen-1791131.html` Zugriff: 23.01.2014 13:31 Uhr

[3] `http://www.eclipse.org/Xtext/documentation.html#DSL` Zugriff: 23.01.2014 13:45 Uhr

[4] Michael Wiedeking „Sprachschöpfer“in „iX Developer Programmieren heute“, Ausgabe 1/2013 S. 12-17

[5] `http://www.eclipse.org/Xtext/documentation.html#Overview` Zugriff: 24.01.2014 12:40 Uhr