Diplomarbeit Nr. 3388

Dynamische Prüfung von Spreadsheets

Manuel Lemcke

Studiengang: Softwaretechnik

Prüfer/in: Prof. Dr. Jochen Ludewig
Betreuer/in: M. Sc. Daniel Kulesz

Beginn am: 27. August 2012
Beendet am: 26. Februar 2013

CR-Nummer: H.4.1, D.2.5
Kurzfassung

Spreadsheets sind weit verbreitet. Gleichzeitig weisen sie laut verschiedenen Studien Fehlerquoten um die 90% auf. Aufgrund dessen wurde SIF (Spreadsheet Inspection Framework) geschaffen. Auf Basis dieses Rahmenwerks können Prüfzentren für Spreadsheets entwickelt werden. SIF wird im Rahmen dieser Arbeit um dynamische Prüfungen erweitert.
## Inhaltsverzeichnis

1. Einleitung  
   1.1. Motivation .............................................. 9  
   1.2. Ziel .................................................... 9  
   1.3. Gliederung .............................................. 10

2. Grundlagen  
   2.1. Statische und dynamische Prüfungen ................................. 11  
      2.1.1. Statische Prüfungen .................................. 11  
      2.1.2. Dynamische Prüfungen ................................ 11  
   2.2. Endanwender und Programmierung ..................................... 12  
      2.2.1. Gründe Spreadsheets zu testen ............................. 12  
   2.3. Das SIF Rahmenwerk ......................................... 13  
      2.3.1. Die Metapher ........................................... 13  
      2.3.2. Ablauf einer Inspektion .................................. 14

3. Konzept  
   3.1. Anforderungen .............................................. 17  
   3.2. Notation für Vorschriften ...................................... 19  
      3.2.1. Bewertungskriterien .................................... 19  
      3.2.2. Kandidaten ............................................. 20  
      3.2.3. Eigene Sprache .......................................... 22  
      3.2.4. Entscheidung ........................................... 23  
   3.3. Integration in SIF ........................................... 23  
      3.3.1. Überprüfen der Metapher ................................ 27

4. Die Implementierung  
   4.1. IST-Zustand ................................................. 30  
      4.1.1. Aufbau ............................................... 30  
      4.1.2. Allgemeines zum Hinzufügen neuer Vorschriften .................. 30  
   4.2. Erweiterung ............................................... 32  
      4.2.1. Klassenmodell .......................................... 32  
      4.2.2. Erweiterbarkeit ........................................ 34  
      4.2.3. Austauschformat ........................................ 34

5. Evaluation  
   5.1. Rahmenbedingungen .......................................... 37  
      5.1.1. Arbeitsumgebung der Probanden ............................. 37
5.1.2. Prüfling .................................................. 37
5.1.3. Aufgabenstellung .......................................... 38
5.1.4. Die Probanden ............................................ 39
5.2. Resultate ...................................................... 40
   5.2.1. Spezifizieren von Regeln ............................... 40
   5.2.2. Umfrage ................................................ 40
   5.2.3. Fehler in SIF .......................................... 41

6. Rückblick 43
   6.1. Zukünftige Arbeiten ....................................... 43

A. Anhang 45
   A.1. Aufbau der CD-ROM ...................................... 45
   A.2. Evaluation .............................................. 45

Literaturverzeichnis 51
Abbildungsverzeichnis

2.1. Verwendung von SIF durch eine Anwendung ........................................... 15
3.1. Ablauf einer Inspektion in der ersten Ausbaustufe (adaptiert aus [Zit12]) ... 24
3.2. Ablauf in der zweiten Ausbaustufe .............................................................. 26
3.3. Inspektion dynamischer Vorschriften .......................................................... 27
4.1. Aufbau von SIF. Angelehnt an [Zit12, Abbildung 8.1], jedoch detaillierter. ... 31
4.2. Aufbau einer dynamischen Vorschrift. Details der abstrakten Elternklassen werden ausgelassen. ................................................................. 32
4.3. Vererbungshierarchie der Bedingungen ...................................................... 33
4.4. Vererbungshierarchie der Prüfstände ........................................................... 34
5.1. Der Eclipse XML Editor im graphischen Modus .......................................... 38
5.2. Zur Evaluation verwendetes Spreadsheet .................................................... 39

Tabellenverzeichnis

4.1. Abbildung der Metapher auf Klassen ....................................................... 29
5.1. Median der Fragen zur Schwierigkeit ....................................................... 40

Verzeichnis der Listings

3.1. Definition der Notation in [BCP+03] ....................................................... 21
1. Einleitung

1.1. Motivation


1.2. Ziel

1. Einleitung

1.3. Gliederung

Die Arbeit ist in folgender Weise gegliedert:

**Kapitel 2 – Grundlagen**: In diesem Kapitel werden Grundlagen, auf die später zurückgegriffen wird, beschrieben und verwendete Begriffe werden definiert.

**Kapitel 3 – Konzept**: In diesem Kapitel werden geplante Abläufe und Strukturen und Entscheidungen, die nicht die konkrete Implementierung betreffen beschrieben.

**Kapitel 4 – Die Implementierung**: Die etwas technischere Beschreibung der Umsetzung des Konzepts.

**Kapitel 5 – Evaluation**: Die Beschreibung der Benutzerevaluation der zweiten Ausbaustufe von SIF.
2. Grundlagen

2.1. Statische und dynamische Prüfungen

Da es das Ziel dieser Arbeit ist, ein Rahmenwerk, das aktuell auf statische Prüfungen beschränkt ist, um dynamische Prüfungen zu erweitern, wird geklärt, was mit diesen Begriffen im Kontext von Spreadsheets gemeint ist. Dazu wird in diesem Kapitel zunächst untersucht, was die Begriffe traditionell bedeuten und anschließend, wie diese Bedeutungen auf die Spreadsheetdomäne angewandt werden können.

2.1.1. Statische Prüfungen

Statische Prüfungen stellen im Software Qualitätsmanagement ein großes Feld dar. Sie können in am Rechner durchgeführte Prüfungen (mechanisch) und ohne Hilfe des Rechners durchgeführte Prüfungen (nichtmechanisch) unterteilt werden [FLS04, LL07]. Nichtmechanische Prüfungen spielen im Software Qualitätsmanagement zwar eine große Rolle z.B. in der Form von Reviews, da es in dieser Arbeit jedoch um eine Software geht, die Prüfungen automatisch durchführen soll, wird auf diese nicht weiter eingegangen.

Am Rechner durchgeführte statische Prüfungen können genutzt werden, um Daten darüber zu generieren, ob und wie stark ein Prüfling von gewissen Normen und Vorschriften abweicht, ob eine formal definierte Spezifikation eingehalten wird oder um Metriken über den Prüfling zu erheben (vgl. [FLS04]). Übertragen auf Spreadsheets lassen sich also beispielsweise die durchschnittliche Formelkomplexität oder ob eine Vorschrift wie „keine Konstenten in Formeln“ eingehalten wird.

2.1.2. Dynamische Prüfungen

Unter dynamischen Prüfungen versteht man im Software Qualitätsmanagement solche Prüfungen, die den Prüfling ausführen. [FLS04] führt als Unterarten dynamischer Prüfungen den Test und die Leistungsmessung auf. Da ich vermute, dass die Leistung bei Spreadsheets nur eine geringe Rolle spielt konzentriere ich mich hier auf den Test. In [LL07] wird der Test folgendermaßen definiert:

**Testen** ist die - auch mehrfache - Ausführung eines Programms auf einem Rechner mit dem Ziel, Fehler zu finden.

- Der dargestellte Wert der Formelzelle (die Zelle, in der die Formel enthalten ist). Die Zelle stellt das Ergebnis je nach Formatierung dar.
- Alle Zellen, die auf die Formelzelle direkt oder indirekt verweisen.
- Diagramme, die den Zellwert verwenden.
- Bedingte Eigenschaften von Zellen wie z.B. die Hintergrundfarbe.

2.2. Endanwender und Programmierung


2.2.1. Gründe Spreadsheets zu testen

2.3. Das SIF Rahmenwerk

Rahmenwerk (Framework) Ein Rahmenwerk ist eine generische Lösung, die die Basis für sehr ähnliche Anwendungen bildet. Im Gegensatz zu einer Klassenbibliothek wird durch ein Rahmenwerk auch der Kontrollfluss der Anwendung vorgegeben (vgl. [LL07]).

Das in Java entwickelte Rahmenwerk SIF (Spreadsheet Inspection Framework) wurde im Rahmen der Diplomarbeit [Zit12] erstellt. Es unterstützt bereits drei statische Vorschriften:

- **Konstanten in Formeln** Mit dieser Vorschrift lässt sich überprüfen, ob Formeln konstante Werte enthalten. Bestimmte Werte lassen sich hiervon ausnehmen.
- **Leserichtung** Hiermit wird untersucht, ob nur Zellen, die sich rechts und unterhalb der betrachteten Zelle befinden diese referenzieren.
- **Formelkomplexität** Mit dieser Vorschrift lässt sich spezifizieren, wie viele Operationen maximel in Formeln enthalten sein dürfen und wie tief die maximale Schachtelungstiefe ist.

Der Umfang der unterstützten Prüfungen beschränkt sich zwar auf statische Prüfungen, bei der Konzeption wurde jedoch berücksichtigt, dass es um dynamische und nichtmechanische Prüfungen erweiterbar sein soll.

2.3.1. Die Metapher


**Prüfung:** Eine Prüfung von Spreadsheets hat zum Ziel die Konformität von Spreadsheets mit Richtlinien zu überprüfen. Diese Richtlinien werden im Unterschied zum Kfz-Prüfzentrum vom Nutzer festgelegt.

**Inspektions-Werkstatt:** Stellt die technischen Mittel bereit, um die Prüfung von Spreadsheets durchzuführen. Dazu gehören sowohl allgemeine Werkzeuge um die Eigenschaften von Spreadsheets zu untersuchen als auch spezielle Werkzeuge um Bestandteile von Spreadsheets zu untersuchen.
2. Grundlagen

**Vorschrift:** Eine Vorschrift definiert anpassbare Prüfkriterien, denen bestimmte Bestandteile eines Spreadsheets oder das Spreadsheet selbst genügen müssen.

**Richtlinie:** Eine Richtlinie besteht aus einer oder mehrerer Vorschriften.

**Verstoß:** Bei Abweichungen von einer Vorschrift wird ein Verstoß gemeldet. Ein solcher Verstoß kann je nach Vorschrift gegen die verstoßen wird unterschiedliche Daten enthalten.

**Inspektionsauftrag:** Ein Nutzer der Inspektions-Werkstatt kann Inspektionsaufträge aufgeben. Er gibt dazu einen Namen sowie das zu prüfende Spreadsheet an und wählt eine Richtlinie aus.


**Element-Scanner:** Element-Scanner sind Module, die Spreadsheet-Elemente erkennen und dem Spreadsheet-Inventar hinzufügen können. SIF bietet in der ersten Ausbaustufe bereits Möglichkeiten, weitere Element-Scanner einzubinden.

**Spreadsheet-Inventar:** Das Spreadsheet beinhaltet alle Spreadsheet-Elemente, die beim Einlesen des Spreadsheets oder durch Element-Scanner erstellt wurden.

**Prüfstände:** Mit Prüfständen kann die Einhaltung von Vorschriften überprüft werden. Ein Prüfstand kann entweder aus mehreren Prüfkwerzeugen zusammengestellt werden oder monolithisch sein.

**Prüfwerkzeuge:** Prüfwerkzeuge sind modulare Bestandteile von zusammengestellten Prüfständen. Sie überprüfen jeweils genau eine Klasse von Spreadsheet Elementen. In der ersten Ausbaustufe von SIF existiert zwar die Struktur um Prüfwerkzeuge einzubinden, allerdings gibt es keine konkreten Umsetzungen.

2.3.2. Ablauf einer Inspektion

Im Folgenden werden die Schritte erläutert, die nötig sind um SIF aus einer eigenen Anwendung heraus zu verwenden. Die Darstellung und Granularität sind bewusst anders gewählt als in [Zit12] um die tatsächlichen Abläufe deutlicher darzustellen.
2.3. Das SIF Rahmenwerk


**Scannen des Spreadsheets anstoßen**

In diesem Schritt wird das Spreadsheet durch alle registrierten Elementscanner untersucht. Auf diese Weise wird das Spreadsheetinventar erstellt, das dem Inspektionsauftrag zugewiesen wird.

**Richtlinie erstellen**


Abbildung 2.1: Verwendung von SIF durch eine Anwendung
2. Grundlagen

**Richtlinie registrieren**

Nach der Erstellung muss die Richtlinie noch registriert werden. Durch das Registrieren der Richtlinie weist SIF die Richtlinie dem Inspektionsauftrag zu.

**Inspektion**


**Berichtserstellung**

Die Berichtserstellung ist nicht zwingend notwendig. Es ist durchaus denkbar, dass die Anwendung die Verstöße aus dem Inspektionsauftrag ausliest und in einer eigenen Darstellung präsentiert.
3. Konzept

3.1. Anforderungen

Vorgaben

Das Rahmenwerk SIF soll im Rahmen dieser Arbeit in erster Linie um die Funktionalität erweitert werden, dynamische Vorschriften erstellen und prüfen zu können. Außerdem soll es möglich sein, Ein- und Ausgabezellen automatisch zu identifizieren und Verstöße gegen Vorschriften auszugeben.

Eigene Anforderungen

Beibehalten von Struktur und Metapher

Um die Wartung und weitere Erweiterungen zu erleichtern sollen die vorhandenen Prinzipien, Strukturen und die Metapher „Kfz-Prüfwerkstatt“ beibehalten werden. Eine Erweiterung der bestehenden Struktur um neue Teilstrukturen ist also einer Veränderung der bereits existierenden Teilstrukturen vorzuziehen.

Austauschformat für Prüfregeln

3. Konzept

einfachen graphischen Benutzungsoberfläche zur Spezifikation von Vorschriften über eine
automatische Hintergrundüberwachung bei der Bearbeitung von Spreadsheets bis hin zur
Integration in ein Document Management System.

**Berührungsfreiheit**

Das geprüfte Spreadsheet soll durch die Prüfung nicht verändert werden, denn jede Verände-
rung am Prüfling führt unter Umständen zu Seiteneffekten, die die Resultate verfälschen.

**Überprüfbar SIF Spreadsheet-Element Eigenschaften**

Es sollen nicht nur Zellwerte sondern auch andere Eigenschaften referenzierbarer
Spreadsheet-Elemente überprüft werden können.

**Umfang der angebotenen Vorschriften**

Um die Fähigkeit der zweiten Ausbaustufe, dynamische Prüfungen durchzuführen, demons-
trieren und testen zu können sollen drei Vorschriften angeboten werden:

**Einfacher Vergleich von Zellwerten** Die wohl einfachste Vorschrift. Sie schreibt vor, dass der
u.U. durch eine Formel berechnete Zellwert einen bestimmten Wert hat.

**Übereinstimmung mit Intervall** Hierbei soll überprüft werden, ob der u.U. durch eine For-
mel berechnete Zellwert innerhalb eines Intervalls liegt. Das Intervall soll als offen,
geschlossen oder nur auf einer Seite offen angegeben werden können.

**Überprüfung der Anzahl von Spreadsheet-Elementen eines Typs** Diese Prüfung ist auf den
ersten Blick eine statische Prüfung. Allerdings kann sich durch das Eintragen von
Testeingaben die Anzahl von Spreadsheet-Elementen eines bestimmten Typs verändern.
So kann bei einem nicht gesperrten Spreadsheet das Überschreiben einer Formel dazu
führen, dass bestimmte Zellen keine Eingabe-, Ausgabe- oder Zwischenzellen sind. Ein
weiteres Szenario stellen zukünftige Element-Scanner dar, die z.B. Zellen mit Fehlern
oder einer bestimmten Formatierung erkennen. Da eine Formatierung abhängig vom
Zellwert sein kann, kann sich diese durch das Eintragen von Werten und berechnen
der Formeln ändern.

**Erweiterbarkeit um neue dynamische Vorschriften**

Wie auch in der ersten Ausbaustufe soll es möglich sein, dem Rahmenwerk neue Vorschriften
hinzuzufügen. Es soll dabei vor allem erleichtert werden, Vorschriften, die eine dynamische
Prüfung voraussetzen, hinzuzufügen.
3.2. Notation für Vorschriften

3.2.1. Bewertungskriterien

Um ein Austauschformat für dynamische Vorschriften zu erstellen wird eine Notation bzw. Sprache benötigt, die geeignet ist Spreadsheet-Bedingungen zu beschreiben und von Mensch und Maschine lesbar ist. Diese gewünschten Eigenschaften werden im Folgenden genauer formuliert. Sie sind allerdings nicht als harte Anforderungen wie in einer Softwarespezifikation zu sehen, sondern als Bewertungskriterien um den geeignetsten Kandidaten zu bestimmen.

Sprachumfang


Dokumentation

Sollte die Sprache komplex oder umfangreich sein, wird eine (gute) Dokumentation gefordert. Diese hilft sowohl dem Entwickler, den Parser zu implementieren, als auch dem Endanwender Richtlinien zu spezifizieren.

Endbenutzertauglichkeit

Auch wenn die Sprache selbst in zukünftigen Ausbaustufen wahrscheinlich durch eine graphische Benutzungsschnittstelle abgelöst wird, sollte die Sprache auch von Endbenutzern verwendet werden können. Es wäre zwar schön aber es ist nicht nötig, dass völlig unerfahrene Benutzer sie benutzen können.

Implementierungsaufwand

Da die Zeit zum Implementieren der Deserialisierung nur einen Bruchteil des gesamten Aufwands darstellen soll und somit stark begrenzt ist, muss der hierfür zu erwartende Aufwand so gering wie möglich sein. Es ist also ein großer Vorteil, falls für eine Sprache bereits Java Klassenbibliotheken oder andere Hilfsmittel existieren.
3. Konzept

Erweiterbarkeit

Da es möglich sein soll, neue Typen dynamischer Vorschriften in SIF einzubinden, muss auch das Austauschformat erweiterbar sein ohne dabei zu viel Aufwand zu erzeugen.

3.2.2. Kandidaten

Um eine Notation zu finden, wurden verschiedene existierende Sprachen zur Beschreibung von Prüfregeln verschiedener Art ermittelt und unter Berücksichtigung der im vorigen Kapitel beschriebenen Bewertungskriterien auf ihre Tauglichkeit hin untersucht. Außerdem wurde ein Zeitfenster festgelegt innerhalb dessen eine geeignete Sprache gefunden sein musste. Andernfalls sollte eine eigene Sprache erstellt werden.

Bei der Suche nach passenden Sprachen ergaben sich zunächst folgende Kandidaten:

- Eine namenlose, in [BCP+03] definierte
- Assertion Definition Language (ADL)
- Oracle CDL
- UML OCL (Object Constraint Language)

ADL und CDL


Burnett2003

Hierbei handelt es sich um eine Notation, die an der Oregon State University geschaffen wurde, um Regeln für Spreadsheets zu beschreiben. Sie wurde zunächst in [BCP+03] und später auch in [Cre05] verwendet.

3.2. Notation für Vorschriften

Listing 3.1 Definition der Notation in [BCP⁺03]

\[
( N, \{\text{and-assertions}\}), \text{ where:}
\]
- each and-assertion is a set of or-assertions,
- each or-assertion is a set of (unary-relation, value-expression) and (binary-relation, value-expression-pair) tuples,
- each unary-relation \( \in \{=, <, \leq, >, \geq\} \),
- each binary-relation \( \in \{\text{to-closed, to-open, to-openleft, to-openright}\} \),
- each value-expression is a valid formula expression in the spreadsheet language,
- each value-expression-pair is two value-expressions.


Betrachtet man die Notation im Hinblick auf die festgelegten Bewertungskriterien ergibt sich folgendes Bild:

**Sprachumfang** Die Notation ist in ihrem Umfang sehr gut auf die Problemstellung „Beschreiben einer Bedingung zur dynamischen Prüfung eines Spreadsheet-Elements“ zugeschnitten. Sie kann allerdings nur als Grundlage für das eigene Austauschformat dienen, da bestimmte Elemente fehlen um den geplanten Umfang ausdrücken zu können. Dazu gehören or-assertion Elemente für die Zählung von Spreadsheet-Elementen eines Typs sowie Möglichkeiten um Testeingaben zu definieren und festzulegen ob eine Bedingung eine Invariante oder eine Nachbedingung ist.

**Dokumentation** : Eine Dokumentation existiert nur in Form des Artikel [BCP⁺03] und der Master-Thesis [Cre05]. Allerdings ist das kein großes Manko da die Sprache weder komplex noch umfangreich ist und nur als Basis für eine eigene Sprache dienen könnte.

**Implementierungsaufwand** : Es existieren keine öffentlich zugänglichen Parser-Implementierungen, sodass das Parsen selbst implementiert werden müsste.

**Erweiterbarkeit** : Die Erweiterbarkeit der Notation hängt der Art der Parser-Implementierung ab. Falls es möglich ist einen Schema- oder Grammatikbasierten Parser oder Parser-Generator zu erstellen muss im Fall einer Erweiterung nur das Schema bzw. die Grammatik angepasst werden. Falls das nicht der Fall ist müsste beispielsweise im Falle eines Expertensystems, bei dem jedes Sprach-Element an ein eigenes Parse-Modul weitergereicht wird das Parse-Modul des Oberelements angepasst und ein Modul für das neue Element hinzugefügt werden.

Die Notation kann nur als Grundlage für eine eigenes Austauschformat dienen, da beispielsweise das Zählen von Elementen eines Typs nicht spezifiziert ist. Außerdem

**UML OCL**

3. Konzept


Endbenutzertauglichkeit: Da die OCL eine prädikatenlogische Sprache und ähnlich komplex ist wie eine Programmiersprache halte ich sie nicht für geeignet von Endbenutzern verwendet zu werden.


3.2.3. Eigene Sprache

Da innerhalb des festgelegten Zeitfensters keine Sprache gefunden wurde, von der erwartet werden konnte, dass sie allen Bewertungskriterien in ausreichendem Maße genügt, musste damit begonnen werden eine eigene Beschreibungssprache zu erstellen. Zum Erstellen eigener Sprachen gibt es eine Vielzahl möglicher Technologien und Werkzeuge. Ich habe mich mit den folgenden drei Möglichkeiten auseinandergesetzt:

JavaCC steht für „Java Compiler Compiler“ und kann Parser für eine große Zahl von Grammatiken\(^1\) erstellen. In Kombination mit anderen Werkzeugen, wie JJTree kann man so aus einer Eingabe, die der Grammatik folgt, einen Baum der Sprachelemente erstellen. Im Fall einer Notation für Vorschriften, würde man diesen Baum durchlaufen und währenddessen das eigene Datenmodell aufbauen.

XML steht für extensible Markup Language (erweiterbare Auszeichnungssprache) und ist eine hierarchische Auszeichnungssprache, die in den letzten Jahren zu großer Verbreitung gefunden hat. Dadurch existiert eine Vielzahl an Werkzeugen und in vielen

\(^1\)LL(k) Grammatiken. Für weitere Informationen zum Thema Compiler und Grammatiken siehe [ALSU07]
hoher Programmiersprachen sind Klassenbibliotheken enthalten, die den Umgang mit XML Dateien erleichtern. Um in Java XML zu serialisieren gibt es prinzipiell drei Möglichkeiten:

- Die automatische Erstellung eines Objektbaums z.B. mit DOM (Data Object Model). Der Objektbaum besteht hier im Unterschied zur XML Bindung allerdings aus Objekten der eingesetzten API. Die Anwendung, die die API einsetzt muss den Objektbaum also durchlaufen und transformieren. Der Vorteil gegenüber der XML Bindung ist, dass mehr Anwendungswissen eingesetzt werden kann. Dadurch können z.B. Schwierigkeiten mit Interfaces umgangen werden.
- Schrittweises Lesen des Eingabestroms z.B. mit SAX (Simple API for XML). Diese Methode benötigt bei guter Implementierung weniger Arbeitsspeicher als das Erstellen eines Objektbaums und ist dabei genauso universell einsetzbar. Allerdings besteht eine größere Gefahr, dass bei der Implementierung Fehler gemacht werden.

3.2.4. Entscheidung


3.3. Integration in SIF

3. Konzept

<table>
<thead>
<tr>
<th>SIF weitere Ausbaustufen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspektionsauftrag erstellen</td>
</tr>
<tr>
<td>Spreadsheet-Inventar visualisieren</td>
</tr>
<tr>
<td>Ergebnisse visualisieren und untersuchen</td>
</tr>
<tr>
<td>Dynamische Prüfung</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SIF erste Ausbaustufe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inspektionsauftrag erstellen</td>
</tr>
<tr>
<td>Spreadsheet-Inventar erstellen</td>
</tr>
<tr>
<td>Inspektionsauftrag konfigurieren</td>
</tr>
<tr>
<td>Inspektion durchführen</td>
</tr>
<tr>
<td>Ergebnisse analysieren</td>
</tr>
<tr>
<td>Bericht erstellen</td>
</tr>
</tbody>
</table>

Abbildung 3.1.: Ablauf einer Inspektion in der ersten Ausbaustufe (adaptiert aus [Zit12])

Stattdessen soll die dynamische Prüfung im Rahmen der normalen Prüfung für alle Spreadsheet-Elemente durchgeführt werden, sodass auch Spreadsheet-Elemente, die keine Verstöße bei der statischen Prüfung gezeigt haben geprüft werden.


Da es viele verschiedene Arten von Vorschriften geben kann, die eine Ausführung des Spreadsheets zu ihrer Überprüfung benötigen werden austauschbare Unterelemente für den neuen Prüfstand benötigt. Prinzipiell böte sich hier also ein aus Prüfwerkzeugen zusammengesetzter Prüfstand (siehe Kapitel 2.3.1) an. Da jedoch zusammengesetzte Prüfstände und
Prüfwerkzeuge jeweils auf eine Klasse von Spreadsheet-Elementen beschränkt sind, wird stattdessen ein **monolithischer Prüfstand** verwendet.

### Auswertung von Formeln

Um die Auswertung der Formeln nicht selbst implementieren zu müssen soll die bereits eingesetzte externe Klassenbibliothek dazu eingesetzt werden. Die einzige aktuell verwendete Klassenbibliothek Apache POI unterstützt die Auswertung von Formeln bereits. Falls ein neues Format unterstützt werden soll und die entsprechende Klassenbibliothek keine Formelauswertung anbietet, führt das allerdings dazu, dass die dynamische Prüfung mit diesem Format nicht funktioniert. Das heißt in diesem Fall muss dennoch eine eigene Formelauswertung implementiert werden.

### Ablauf


**Richtlinie erstellen** Im ersten Schritt wird die Richtlinie erstellt. Entweder programmatisch oder indem eine entsprechende XML Datei deserialisiert wird. Die Anwendung übergibt die so erhaltene Richtlinie zusammen mit dem Pfad des Spreadsheets an SIF. Diesen Pfad kann sie entweder der Richtlinie entnommen oder selbst bestimmt haben z.B. durch einen Auswahldialog.


**Inspektion durchführen** Die Inspektion wird von einem neuen Prüfstand durchgeführt. Da Invarianten immer gelten sollen, werden diese zu Beginn und zu den Zeitpunkten, an denen sich das Spreadsheet ändert überprüft (siehe Abbildung 3.3). Nachdem die Invarianten ganz zu Beginn überprüft wurden wird das Spreadsheet-Modell (der externen Klassenbibliothek) kopiert, sodass das Original erhalten bleibt. Anschließend werden die Testeingaben eingefügt. Nachdem die Invarianten erneut geprüft wurden, werden die Formeln mit Hilfe der externen Klassenbibliothek ausgewertet. Anschließend werden zuerst die Invarianten und danach die
3. Konzept

Abbildung 3.2.: Ablauf in der zweiten Ausbaustufe

26
3.3. Integration in SIF

Für alle dynamischen Regeln

1. Formeln evaluieren
2. Invarianten prüfen
3. Spreadsheet Modell kopieren
4. Testeingaben realisieren
5. Invarianten prüfen
6. Nachbedingungen prüfen
7. Verstoßliste ergänzen
8. Ende

Abbildung 3.3.: Inspektion dynamischer Vorschriften


3.3.1. Überprüfen der Metapher

Um zu beurteilen wie gut die dynamische Prüfung in die Metapher passt untersuche ich, inwieweit ein Fahrzeug in einem Kfz-Prüfzentrum ausgeführt wird. In einer solchen Einrichtung werden nicht nur Sichttests durchgeführt, sondern auch allerlei Tests, bei denen jeweils mindestens das untersuchte Teilsystem ausgeführt wird. So wird beispielsweise bei einem Bremsanlagentest eine Wirkungsprüfung auf einem Bremsprüfstand durchgeführt [Klu90]. Auf einem solchen Bremsprüfstand werden die Räder durch eine externe Rolle oder durch ein Rollband angetrieben [Rot95]. Um die die Bremswirkung zu messen wird die Bremsanlage betätigt. Das gesamte Fahrzeug wird für Bremstests nur in Bewegung gesetzt, falls es zu groß für den Bremsstand ist. In diesem Fall wird die Bremsleistung mit einem schreibenden Bremsmeßgerät aufgezeichnet [Klu90]. Laut § 29 Anlage IIIa der StVZO gehört zur Hauptuntersuchung von PKW, die in Deutschland regelmäßig durchgeführt werden muss, eine Messung des Fahrgeräusches. Außerdem beginnt die Hauptuntersuchung seit dem 1. Juli 2012 mit einer Probefahrt von mindestens 8 km/h um elektronische Systeme zu aktivieren. Man also davon sprechen, dass das Fahrzeug ausgeführt wird und nicht nur statische sondern auch dynamische Prüfungen durchgeführt werden. Das Fahrzeug wird außerdem in einer Umgebung ausgeführt, die nicht dem täglichen Gebrauch entspricht, was mit der Ausführung des Spreadsheets durch eine Klassenbibliothek statt der eigentlichen Spreadsheet-Umgebung verglichen werden kann [Tue12].
4. Die Implementierung


<table>
<thead>
<tr>
<th>Begriff in der Metapher</th>
<th>Klassenname</th>
<th>Bemerkung</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Empfang)</td>
<td>FrontDesk</td>
<td>Der Empfang ist in der Metapher nicht aufgeführt. Allerdings ist er für die Implementierung eine wichtige Komponente, da er die Schnittstelle zur aufrufenden Anwendung darstellt.</td>
</tr>
<tr>
<td>Inspektionsauftrag</td>
<td>InspectionRequest</td>
<td></td>
</tr>
<tr>
<td>Prüffeld</td>
<td>TestBay</td>
<td></td>
</tr>
<tr>
<td>Prüfstand</td>
<td>AbstractTestFacility</td>
<td>Je nach Implementierung als Unterklassse von MonolithicTestFacility oder CompositeTestFacility.</td>
</tr>
<tr>
<td>Richtlinie</td>
<td>Policy</td>
<td></td>
</tr>
<tr>
<td>Verstoß</td>
<td>IViolation</td>
<td></td>
</tr>
<tr>
<td>(Vorschriftsmanager)</td>
<td>PolicyManager</td>
<td>Dieser ist, wie der Empfang, nicht in der Metapher aufgeführt aber eine zentrale Komponente. Er bestimmt, welche Vorschrift mit welcher Prüfwerkstatt geprüft wird.</td>
</tr>
</tbody>
</table>

Tabelle 4.1: Abbildung der Metapher auf Klassen
4. Die Implementierung

4.1. IST-Zustand

4.1.1. Aufbau

Das SIF Design besteht aus vier Paketen, die im Folgenden beschrieben sind. Abbildung 4.1 stellt die Pakete und die wichtigsten in ihnen enthaltenen Klassen dar. Damit die Abbildung nicht zu unübersichtlich wird werden lediglich die Beziehungen der Klassen über Paketgrenzen hinweg dargestellt.

FrontOffice

Das Paket FrontOffice enthält die Klasse FrontDesk, die die Schnittstelle nach außen darstellt und öffentliche Methoden anbietet um z.B. Inspektionsaufträge anzufragen oder einen Bericht zu erstellen. In FrontOffice enthalten sind außerdem die Klassen PolicyManager und InspectionManager, an die Anfragen von FrontDesk delegiert werden und die die Pakete IO und TechnicalDepartment verwenden um diese Anfragen zu bearbeiten.

TechnicalDepartment

Im Paket TechnicalDepartment sind die Prüflogik und die Element-Scanner Implementierungen untergebracht. Die Klasse TechnicalManager delegiert die meisten Aufrufe an TestBayManager oder ScanningManager.

IO

Mit den in diesem Paket enthaltenen Klassen können Spreadsheets ausgelesen und in das eigene Datenmodell transformiert werden. Schnittstelle nach zu anderen Paketen ist die Klasse DataFacade.

4.1.2. Allgemeines zum Hinzufügen neuer Vorschriften


Das kann allerdings nur innerhalb von SIF selbst stattfinden, denn es existieren keine öffentlichen Methoden um Klassen beim PolicyManager zu registrieren. Außerdem existiert keine Methode, um die aktuell verwendete Instanz von PolicyManager zu ermitteln. Es ist
Abbildung 4.1: Aufbau von SIF. Angelehnt an [Zit12, Abbildung 8.1], jedoch detaillierter.

allerdings ohne größeren Aufwand möglich FrontOffice um solche Methoden zu erweitern, sodass auch eine auf Basis des Rahmenwerks entwickelte Anwendung eigene Vorschriften hinzufügen könnte.
4. Die Implementierung

Abbildung 4.2.: Aufbau einer dynamischen Vorschrift. Details der abstrakten Elternklassen werden ausgelassen.

4.2. Erweiterung

4.2.1. Klassenmodell

In Kapitel 3.3 wurde spezifiziert, dass eine dynamische Vorschrift Testeingaben, Nachbedingungen und Invarianten haben soll. Nachbedingungen und Invarianten haben die gleichen Eigenschaften, lediglich der Prüfzeitpunkt unterscheidet sich. In Kapitel 3.3 wurde festgestellt, dass diese gleichen Eigenschaften ein Ziel, ein SOLL-Wert und eine Eigenschaftsangabe sind. Daher werden sie durch die selbe abstrakte Klasse implementiert: `AbstractCondition`. Je nachdem, ob es sich um einen Vergleich, ein Intervall oder ein Zählen von Elementen handelt handelt wird eine andere Spezialisierung verwendet (siehe 4.2). Das Ziel einer Bedingung wird durch zwei Attribute implementiert, nämlich `target` und `elementType`. Target wird verwendet, wenn ein Ziel über eine Adresse oder einen Namen direkt angesprochen werden kann. `ElementType` hingegen wird eingesetzt, wenn alle Spreadsheet-Elemente eines Typs, also auch Spezial-Elemente, die mit einem Element-Scanner erstellt wurden (siehe Kapitel 2.3.1) überprüft werden sollen.

Außerdem werden an verschiedenen Stellen Unterklassen der bestehenden Klassen gebildet.
4.2. Erweiterung

Abbildung 4.3.: Vererbungshierarchie der Bedingungen.


DynamicInspectionRequest erweitert die Implementierung des Inspektionsauftrags um eine Feld zum Halten des Spreadsheet-Modells der externen Bibliothek.


Für die Überprüfung der Bedingungen wurden die abstrakte Klasse AbstractConditionChecker sowie die in Abbildung 4.3 dargestellten Implementierungen geschaffen. Diese Klassen implementieren ein modifiziertes Schablonenmuster (Template Method Pattern), d.h. der konkrete Ablauf wird von AbstractConditionChecker vorgegeben und lediglich die Implementierung von Methoden, die Teilschritte wie die Überprüfung der Bedingung durchführen wird in den konkreten Klassen implementiert. Die Modifizierung des Entwurfsmusters besteht darin, dass die „Hauptmethode“ in der abstrakten Klasse nicht final ist um die Erweiterbarkeit nicht unnötig einzuschränken.
4. Die Implementierung

Abbildung 4.4.: Vererbungshierarchie der Prüfstände

4.2.2. Erweiterbarkeit

**Austauschbare Komponenten**  Um die Austauschbarkeit von Komponenten wie den Klassen zum Einfügen der Testeingaben oder Kopieren zu gewährleisten, wurde vor allem in Paket technicalDepartment das Strategie Entwurfsmusters eingesetzt und insgesamt soweit möglich gegen Schnittstellen programmiert und nicht gegen konkrete Implementierungen, wie es im Sinne guter objektorientierter Entwicklung ohnehin getan werden sollte.

**Hinzufügen neuer Bedingungen**  Um es zu ermöglichen, neue Bedingungen hinzuzufügen wurde in FrontDesk die Methode register(AbstractConditionl.class, AbstractConditionChecker.class) implementiert. So kann eine Anwendung neue Bedingungen hinzufügen ohne den Code von SIF verändern zu müssen.

4.2.3. Austauschformat

Die Serialisierung wird in das Paket IO integriert. Um das Auslesen des externen Datenmodells und die Transformation in das eigene Datenmodell zu trennen, wird die Klasse DataFacade so angepasst, dass sie diese Tätigkeiten als getrennte Methoden anbietet. Zum Serialisieren der DynamicPolicy Objekte sind die Getter der Felder, die im Austauschformat vorhanden sein sollen mit Annotationen versehen, die JAXB mitteilen, ob das Feld als Unterelement oder als Attribut seiner Klasse zu serialisieren ist. In diesen Annotationen ist
4.2. Erweiterung

außerdem angegeben, welchen Namen das Element oder Attribut tragen soll. Für Klassen, die nicht von Haus aus mit JAXB serialisierbar sind, werden Implementierungen der JAXB Klasse XMLJavaAdapter erstellt, um JAXB so mitzuteilen wie sie zu serialisieren sind.

Umsetzung

Um eine Richtlinie abzubilden, sollen folgende XML Elemente eingeführt werden:

dynamicPolicy ist das Wurzelement. Es stellt eine Richtlinie dar. Ihm können optional als Attribute ein Name, ein Autor und eine Beschreibung hinzugefügt werden. Diese Attribute tragen die Namen name, author und description. Es besitzt folgende Unterelemente:

- rules: In dieses Unterelement können beliebig viele Vorschriften (rule Elemente) eingefügt werden.


- spreadsheetFilePath: Der Dateipfad zu dem Spreadsheet, auf das sich die Richtlinie bezieht.

rule stellt eine Vorschrift dar. Optional können ein Autor (author), ein Name (name) und eine Gewichtung (severityWeight) in Form von Attributen angegeben werden. Invarianten, Testeingaben und Nachbedingungen können in den Unterelementen invariants testInputs und postconditions eingefügt werden.

testInput stellt eine Testeingabe dar. Für eine Testeingabe müssen das Ziel, Zelltyp und der Wert angegeben werden. Hierfür existieren die Unterelemente target, type und value.

compare ist eine Bedingung und stellt einen Wertevergleich dar. Das Element kann also in invariants oder postconditions eingefügt werden. Das Element besitzt folgende Unterelemente:

- target: Hiermit kann das Ziel entweder in A1 Notation oder per Name angegeben werden.


- value: Der SOLL-Wert.

- relation: Die Art des Vergleichs. Mögliche Werte dieses Unterelements sind equal, greaterThan, lessThan, lessOrEqual und greaterEqual.
4. Die Implementierung


count ist die dritte Bedingung und stellt die Zählung der Elemente einer Klasse dar. Bei ihr kann das Ziel nur mit elementType angegeben werden. Der durch value angegebene Wert entspricht der erwarteten Anzahl an Elementen, der angegebenen Klasse.
5. Evaluation

Um zu untersuchen, wie gut die erstellte zweite Ausbaustufe von SIF geeignet ist, um dynamische Prüfungen zu spezifizieren und durchzuführen, wurde eine Benutzeroevaluation durchgeführt. Dazu wurde zusätzlich zu der im vorigen Kapitel beschriebenen Implementierung eine Kommandozeilen-Schnittstelle erstellt, die es erlaubt, SIF wie in 3.2 dargestellt auszuführen. Im Kontext der Umfrage wurde statt dem Begriff „Vorschrift“ der Begriff „Regel“ gebraucht, da der Sinn des Elements rule so deutlicher wurde und Endanwender keinen Bezug zur hauptsächlich für die Entwicklung relevanten Metapher haben.

5.1. Rahmenbedingungen

5.1.1. Arbeitsumgebung der Probanden

Als Werkzeug um die Spezifikation im XML Format zu erstellen wurde der XML Editor der Entwicklungsumgebung Eclipse eingesetzt. Dieser Editor bietet Hilfsmittel wie Autokomplierung und kann sowohl im Quelltext-Modus als auch im graphischen Modus verwendet werden. Als Spreadsheets-Umgebung wurde LibreOffice 3.6.3.2 unter Kubuntu 12.04 eingesetzt.

5.1.2. Prüfling

5. Evaluation

Abbildung 5.1.: Der Eclipse XML Editor im graphischen Modus

5.1.3. Aufgabenstellung

Um zu untersuchen, wie gut die Probanden in der Lage sind das XML Austauschformat zu verwenden um Vorschriften zu spezifizieren, wurden ihnen eine Beschreibung der Sprachelemente auf zwei Seiten ausgehändigt und es wurden zwei Aufgaben gestellt. Die erste Aufgabe bestand darin, eine Regel zu spezifizieren, die vorschreibt, dass „Restschuld Gesamt“ nie den Wert 10.000 übersteigt. Hiermit sollte untersucht werden, wie gut die Probanden den Unterschied zwischen Invarianten und Nachbedingungen mit Testeingaben erfassen können und ob sie in der Lage sind, einfache Regeln zu spezifizieren. Für die zweite Aufgabe wurde die Berechnung des Rückzahlungsbetrags bei vorzeitiger Rückzahlung so
5.1. Rahmenbedingungen

Abbildung 5.2.: Zur Evaluation verwendetes Spreadsheet

modifiziert, dass bei einer Restschuld von weniger als 10.000 ein falsches Ergebnis berechnet wird. In der zweiten Aufgabe wurde erklärt, dass das zweite Spreadsheet verändert wurde und das erste Spreadsheet als Test-Orakel verwendet werden kann. Anschließend wurde die Aufgabe gestellt, einen Testfall zu spezifizieren, der einen Verstoß erzeugt.

5.1.4. Die Probanden

Vor der eigentlich Evaluation wurde eine Pilot-Evaluation mit drei Probanden durchgeführt. Dabei viel auf, dass das modifizierte Spreadsheet auf den Rechnern der Probanden in der falschen Version kopiert wurde, wodurch es etwas seltener Fehler produzierte und so die Bearbeitung der zweiten Aufgabe erschwerte.

Leider fanden sich für die eigentliche Evaluation weniger Probanden als erwartet. Dadurch konnte die Evaluation nur mit fünf weiteren Probanden durchgeführt werden. Außerdem bestanden die Probanden ausschließlich aus Informatik und Softwaretechnik Studenten sowie Mitarbeitern des Instituts für Softwaretechnik. Die Evaluation lässt also keine Schlüsse
5. Evaluation

<table>
<thead>
<tr>
<th>Frage</th>
<th>Ergebnis im Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verständlichkeit der Aufgabenstellung</td>
<td>Leicht</td>
</tr>
<tr>
<td>Verständlichkeit der Sprach-Beschreibung</td>
<td>Moderat</td>
</tr>
<tr>
<td>Schwierigkeit: Erstellen von Regeln</td>
<td>Leicht bis moderat</td>
</tr>
</tbody>
</table>

Tabelle 5.1.: Median der Fragen zur Schwierigkeit

darauf zu, wie erfolgreich ein Endanwender ohne Erfahrung auf diesem Gebiet Richtlinien spezifizieren kann.

5.2. Resultate

5.2.1. Spezifizieren von Regeln

Aufgabe 1  Alle Probanden konnten Aufgabe 1 lösen und eine Invariante erstellen, die vorschreibt, dass „Restschuld Gesamt“, also der Wert aus Zelle B34 nicht größer als 10.000 werden darf.

Aufgabe 2  Nur einem Probanden gelang es, Aufgabe 2 zu lösen, also eine Vorschrift zu erstellen, die einen richtig positiven Verstoß meldete. Ein anderer Proband erstellte eine Regel, die einen Verstoß meldete, der durch den schon im Original vorhandenen Fehler ausgelöst wurde. Zwei weitere Probanden erstellten zwar Regeln, die eigentlich zu einem Verstoß geführt hätten aber durch Syntaxfehler, die z.T. aus fehlenden Informationen in der Sprachbeschreibung herrührten, nicht durchgeführt werden konnten.

5.2.2. Umfrage


Zusammenfassung  Zusammenfassend lässt sich sagen, dass die Evaluation gezeigt hat, dass die Kombination aus XML Austauschformat und SIF nicht geeignet war um innerhalb kurzer Zeit komplexe Regeln zu spezifizieren. Da es sich bei den Probanden um Personen mit technischem Hintergrund handelte ist anzunehmen, dass Personen mit weniger technischem Hintergrund noch größere Schwierigkeiten haben werden.
5.2.3. Fehler in SIF

Durch die Evaluation fielen mehrere Probleme in SIF auf. Einer davon war schon in der ersten Ausbaustufe vorhanden. Er bewirkt, dass Werte mit mehreren Nachkommastellen falsch ausgelesen werden und wird eventuell durch die Klassenbibliothek Apache POI verursacht. Ein anderer Fehler kam in der zweiten Ausbaustufe hinzu. Es handelt sich um die Tatsache, dass sich als TestInput alle Werte angeben lassen, also auch Werte, die durch eine Gültigkeitsprüfung im Spreadsheet nicht zugelassen sind. Es wurde zwar nicht spezifiziert, dass das nicht möglich sein soll und kann auch in einer Spreadsheet-Umgebung durch den Endanwender mit „Copy and Paste“ erfolgen, stellt aber dennoch keine normale Nutzung des Spreadsheets dar und sollte somit nicht möglich sein. Außerdem viel auf, dass die meisten Benutzer nicht erwarteten, dass Prozentwerte als Wert zwischen 0 und 1 angegeben werden sollten und eine Eingabe von einem Wert zwischen 0 und 100 gefolgt von einem Prozentzeichen zu Fehlern in der Formelberechnung von Apache POI führt. Im Fall von Zellen, die als Prozentwerte formatiert sind sollte also eine Konversion in SIF implementiert werden.
6. Rückblick

Rückblickend lässt sich sagen, dass das Austauschformat zu viel Zeit in Anspruch genommen hat. Zwar war die Zeit, die für die Suche nach einer geeigneten Sprache aufgewendet wurde begrenzt, aber die anschließende Konzipierung und Implementierung des Austauschformats haben mehr Aufwand verursacht, als gedacht. Diese Zeit wäre vermutlich besser investiert gewesen, wenn stattdessen ein genaueres und weitere Bestandteile umfassendes Konzept erstellt worden wäre. Die Evaluation hätte entweder in einem größeren Rahmen stattfinden oder ganz aufgegeben werden müssen, da die daraus gewonnenen Erkenntnisse nicht besonders umfangreich und vor allem nicht repräsentativ sind.

6.1. Zukünftige Arbeiten

Die möglichen zukünftigen Arbeiten auf Basis von SIF sind umfangreich. Der meiner Meinung nach nächste Schritt sollte die Erstellung eines graphischen Editors für Richtlinien sein, da dieser den Benutzer bei der Erstellung entlastet und die Schnittstelle
A. Anhang

A.1. Aufbau der CD-ROM

Die beiliegende CD-ROM ist folgendermaßen aufgebaut:

Ausarbeitung Der Ordner ausarbeitung enthält dieses Dokument im PDF-Format.

Evaluation Der Ordner evaluation enthält die Evaluations Ordner in dem Zustand wie sie nach der Bearbeitung durch die Probanden jeweils waren.


UML Der Ordner UML enthält die UML Diagramme aus dem Bericht im UMLet Format.

A.2. Evaluation
Evaluation: Dynamische Spreadsheetprüfung mit SIF

Spreadsheets werden in Unternehmen und Verwaltung in hohem Maß eingesetzt, sind jedoch sehr häufig fehlerhaft. Mit der XML-basierten Sprache SpRuDeL lassen sich Regeln für Spreadsheets formulieren, die sich anschließend mit SIF (Spreadsheet Inspection Framework) überprüfen lassen.

In dieser Evaluation soll die Qualität eines Spreadsheets gesichert werden, mit dessen Hilfe sich Erlasse (Vergünstigungen) auf das BAFöG Darlehen berechnen lassen. Diese Erlasse werden gewährt, wenn ein Studium besonders schnell oder mit besonders guten Leistungen abgeschlossen wird bzw. wenn das Darlehen vorzeitig zurückgezahl wird.

Aufgabenstellung


1. Erstellen Sie eine Regel, die sicherstellt, dass Restschuld Gesamt immer kleiner oder gleich 10.000€ ist.

Exkurs: Referenzierung von Zellen

Zellen können über eine Adresse oder einen Namen referenziert werden.

Eine Addressnotation ist die A1 Notation, die hier verwendet wird. Bei dieser Notation wird die Spalte als Buchstabe (mit A beginnend) angegeben und die Zeile als Zahl (mit 1 beginnend) z.B. „B5“.

In Spreadsheet-Umgebungen wie Excel oder LibreOffice können Zellen zusätzlich Namen zugewiesen werden. Diese können über das in Abbildung 1 dargestellte Textfeld betrachtet und verändert werden.
Die Beschreibungssprache SpRuDeL (Spreadsheet Rule Definition Language)

Der Aufbau eines SpRuDeL Dokuments ist folgendermaßen (* = 0...∞, ? = 0...1):

```
<dynamicPolicy>
  <rules>
    <rule name="[string]"> * 
      <invariants>
        (<compare /> | <compare /> | <compare /> ) ?
      </invariants>
      <testInputs> ?
      </testInputs> ?
      <postconditions> ?
      </postconditions> ?
    </rule>
  </rules>
</dynamicPolicy>
```

Die Testeingaben einer Regel lassen sich durch beliebig viele testInput Elemente innerhalb von testInputs spezifizieren. TestInputs hat folgende Unterelemente:

- **target**: Gibt die Zelle als A1-Adresse (z.B. „B5“) bzw. Zellname an, in die geschrieben werden soll. Der Name einer Zelle wird im in Abbildung 1 dargestellten Textfeld angezeigt.
- **type**: Hier muss der Typ des Zelleninhalts der geschrieben werden soll ausgewählt werden
- **value**: Beinhaltet den Wert, der geschrieben werden soll. Hier kann eine Zahl mit einem Punkt als Trennzeichen, ein String oder true bzw. false eingetragen werden.

```
<testInput>
  <target>[A1Address] or [Name] </target>
  <type>NUMERIC, TEXT, BOOLEAN, BLANK, ERROR</type>
  <value>[double], [string], true/false</value>
</testInput>
```


Solche Bedingungen besitzen jeweils mehrere der folgenden Unterelemente:

- **target bzw. elementType**: Das Ziel einer Bedingung kann entweder wie bei einem TestInput durch target oder durch elementType angegeben werden. Mit elementType lassen sich alle Spreadsheet-Elemente eines bestimmten Typs überprüfen. Dazu gehören unter anderem:
  - **InputCell**: Eingabezellen, also Zellen auf die verwiesen wird aber die nicht selbst auf andere Zellen verweisen
  - **OutputCell**: Ausgabezellen, also Zellen die auf andere Zellen verweisen aber auf die nicht von anderen Zellen aus verwiesen wird.
  - **IntermediateCell**: Zellen mit ein- und ausgehenden Verweisen.
• **value:**
Spezifiziert den erwarteten Wert der Invariante oder Nachbedingung. Intervalle besitzen zusätzlich noch eine Obergrenze, die mit dem Element *value2* angegeben wird.

• **relation:**
Legt die Art des Vergleichs oder des Intervalls fest.

Es gibt drei Arten von Bedingungen: compare, interval und count:

• **compare:** Beschreibt einen Wertevergleich also z.B. „Zelle B5 muss größer 0 sein“.

```xml
<compare>
  (<target>[A1Address] or [Name])
  | (elementType[sif class name])
  | <property>[sif class property]:<property>
  | <value>[double]?
  | <relation>[equal, greaterThan, greaterOrEqual, lessThan, lessOrEqual]
</compare>
```

• **interval:** Hiermit lässt sich überprüfen, ob sich ein Wert innerhalb des angegebenen Intervalls befindet. Open bedeutet hierbei, dass die Intervallgrenzen zum Intervall gehören während closed bedeutet, dass die Intervallgrenzen nicht zum Intervall gehören. Mit openLeft bzw. openRight kann die linke (untere) bzw. rechte (obere) Intervallgrenze als offen definiert werden.

```xml
<interval>
  (<target>[A1Address] or [Name])
  | (elementType[sif class name])
  | <property>[sif class property]:<property>
  | <value>[double]?
  | <value2>[double]
  | <relation>[open, closed, openLeft, openRight]
</interval>
```

• **count:** Überprüft die Anzahl der Vorkommnisse eines bestimmten Spreadsheet Elements (Eingabezellen, Referenzen, ...). Mögliche Werte für elementType siehe oben.

```xml
<count>
  | (elementType[sif class name])
  | <value>[int]
</count>
```
Dynamische Spreadsheetprüfung

<table>
<thead>
<tr>
<th>Frage</th>
<th>Antwortmöglichkeiten</th>
</tr>
</thead>
<tbody>
<tr>
<td>28. Wie verständlich fanden Sie die Aufgabenstellung?</td>
<td>Sehr leicht, Leicht, Moderat, Schwer, Sehr schwer</td>
</tr>
<tr>
<td>29. Wie verständlich fanden Sie die Regel-Beschreibung?</td>
<td>Sehr leicht, Leicht, Moderat, Schwer, Sehr schwer</td>
</tr>
<tr>
<td>30. Wie schwer ist es Ihnen gefallen, die Regeln zu erstellen?</td>
<td>Sehr leicht, Leicht, Moderat, Schwer, Sehr schwer</td>
</tr>
<tr>
<td>31. Haben Sie einen graphischen XML Editor verwendet um die Aufgabe zu lösen?</td>
<td>Ja, Nein, ich habe den Quelltext bearbeitet</td>
</tr>
<tr>
<td>32. Erfahrung mit XML: Bearbeiten Sie gelegentlich XML Dokumente mit einem Editor?</td>
<td>Ja, Nein</td>
</tr>
<tr>
<td>33. Wann haben Sie das erste mal ein XML Dokument bearbeitet?</td>
<td>Vor weniger als einem Jahr, Vor weniger als drei Jahren, Vor mehr als drei Jahren, Noch nie</td>
</tr>
<tr>
<td>34. Sonstige Anmerkungen</td>
<td></td>
</tr>
</tbody>
</table>
Literaturverzeichnis


http://elib.uni-stuttgart.de/opus/volltexte/2012/7056. (Zitiert auf den Seiten 7, 13, 14, 15, 24, 29 und 31)

Alle URLs wurden zuletzt am 25.02.2013 geprüft.
Erklärung


__________________________________________

Ort, Datum, Unterschrift