UNIVERSITÄT LEIPZIG

Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE)

Realisierung eines CRF Versionen Mapping Tool

Leipzig, 2. Mai 2014

vorgelegt von:
Victor Christen
Betreuer:
Matthias Löbe

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Resultate einer Studie basieren auf der Auswertung der in jeder Phase erfassten Daten bzlg. eines Patienten oder Probanden. Diese Daten werden mittels eines CRF (Check Report Form), auch Studienbogen genannt, erfasst. Diesbezüglich enthält ein solcher Bogen eine Menge von Datenfeldern (Items), die für die spätere Auswertung relevant sind.
Zu Beginn einer Studie werden deshalb für jede Phase die relevanten Items inklusive ihrer Eigenschaften, Regeln und inhaltlichen Strukturierung ermittelt. Die Menge an CRF Bögen und die entsprechenden Daten für eine Studie können in einem klinischen Datenmanagementsystem verwaltet werden, wie z.B. *OpenClinica*[1]. Für die Durchführung einer Studie existieren verschiedene Richtlinien wie z.B. die US Food and Drug Administration (FDA) und die Good Clinical Pratice (GCP), die das Studienmanagementsystem unterstützen sollte. Bei einer Evaluierung von klinischen Studiendatenmanagementsystemen [2] wurde eine Teilmenge der Anforderung überprüft, ob die Systeme diese erfüllen. Die Resultate zeigen, dass OpenClinica alle diese Anforderungen erfüllt. Aufgrund der freien Verfügbarkeit und der effektiven Unterstützung bzgl. der Einhaltung der Anforderungen wird dieses System ebenfalls im IMISE eingesetzt. Häufig ist die Entwicklung eines CRF Bogens ein fortlaufender Prozess, da im Verlauf der Studie fehlende oder unzureichend definierte Items identifiziert werden. Diese müssen in einer aktualisierten Version ergänzt bzw. geändert werden. Sei nun folgendes Szenario gegeben: Es wurden mittels eines CRF Bogens die relevanten Daten für die Phase der Aufnahme eines Patienten erfasst. Es wurde festgestellt, dass einige Daten detaillierter erfasst werden müssen, so dass eine Anpassung des CRF Bogens erfolgen muss. Dennoch sollen die erfassten Daten weiter verwendet werden. Hierfür ist ein Mapping, eine Abbildung der alten Items auf die neuen Items, notwendig, um die alten Daten in das Schema des neuen CRF Bogens zu transformieren. Des Weiteren ist es für die Weiterentwicklung eines CRF Bogens hilfreich zu wissen, welche Items bisher geändert wurden, so dass eine gezielte Entwicklung fortgeführt werden kann. Da ein CRF Bogen potentiell eine hohe Anzahl an Datenfeldern beinhaltet und mehrere Versionen umfasst, ist eine manuelle Generierung eines Mappings für alle Versionen nicht effizient realisierbar. Eine weitere Problematik bzgl. der Änderungen von Items ist die Relevanz der Änderung. Die Änderungen von Eigenschaften eines Items, die sich auf das Layout bzgl. des CRF Bogens beschränken, verursachen keine Änderung des medizinischen Konzepts, so dass existierende Daten auf das neue Item gemappt werden können. Die Transformation der Daten eines Items, deren medizinisches Konzept sich geändert hat, ist nicht ohne weiteres möglich.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist die Entwicklung einer Applikation für die Generierung eines Mappings für eine Menge von CRF Versionen und eine entsprechende Visualisierung, so dass eine effiziente Identifikation der Änderungen und deren Ursache möglich ist. Ein Mapping repräsentiert eine Abbildung von Items, die eine effiziente Identifikation von gelöschten, hinzugefügten und geänderten Items ermöglicht.

Des Weiteren werden die geänderten Items bzgl. der Relevanz der Änderung klassifiziert, so dass eine Filterung bzgl. der Art der Änderung möglich ist. Bei der Realisierung der Applikation werden folgende Aspekte berücksichtigt:

1. Die Berechnung der Mappings für eine Menge von CRF Versionen soll effektiv und effizient sein.

2. Die GUI der Applikation soll benutzerfreundlich sein und Änderungen sollen intuitiv erkennbar sein.

3. Die Klassifikation der Änderungen soll aussagekräftig bzgl. des Schweregrads der Änderung sein.

2 Grundlagen

2.1 CRF Excel Spezifikation

Der Import eines CRF Bogens in das OpenClinica System erfolgt durch eine Excel- Datei, die die Spezifikation der enthaltenden Datenfelder mit ihren Eigenschaften, Abschnitten und potentiellen Regeln für die Items in einzelnen Tabellen enthält. Das Schema der Excel Mappe entspricht dem Klassendiagramm in Abbildung 1.
Die Klassen entsprechen den einzelnen Tabellen und die Attribute einer Klasse repräsentieren die Spalten der Tabelle. Die `CRF` Tabelle beinhaltet Informationen bzgl. des Namens des CRF, der Version, einer Beschreibung sowie Bemerkungen bzgl. der Erstellung.
Die `Sections` Tabelle definiert die layoutspezifische Struktur der Items des CRF's. Items, die das gleiche `section_label` Attribut aufweisen, werden auf der gleichen Seite aufgeführt. Jede Section wird durch das `section_label` Attribut identifiziert. Visuell wird jeder Abschnitt durch eine Überschrift getrennt, die durch den `section_title` spezifiziert wird.
Die `Groups` Tabelle ermöglicht die Gruppierung von Items. Items, die das gleiche `group_label` besitzen, erscheinen zusammen im CRF. Die Daten, die für die Auswertung notwendig sind, werden durch die Items eines CRF Bogens definiert. Die Spezifikation eines Items beinhaltet die Eigenschaften die in der Tabelle 1 aufgelistet sind.

Spaltenname	**Beschreibung**
ITEM_NAME	Identifier für ein Item
DESCRIPTION_LABEL	Beschreibung
LEFT_ITEM_TEXT	Text, der im CRF auf der linken Seite dargestellt werden soll
UNITS	Definition des Typs der Antwort
RIGHT_ITEM_TEXT	Text, der im CRF auf der rechten Seite dargestellt wird
SECTION_LABEL	Zuordnung zu einer SECTION
GROUP_LABEL	Zuordnung zu einer Gruppe
HEADER	Überschrift für das Item
SUBHEADER	Unterüberschrift für das Item

PARENT_ITEM	vorangeganges Item referenziert mittels ITEM_NAME
COLUMN_NUMBER	horizontale Anordnung des Items, Reihenfolge gem. Nr.
PAGE_NUMBER	Seite des Items bei Papierform des CRF's
QUESTION_NUMBER	Nr. der Frage, erscheint vor LEFT_ITEM_TEXT
RESPONSE_TYPE	Antworttyp basiert auf den Standard HTML Elementen - text, textarea, single-select, radio, multi-select, checkbox, calculation, group-calculation, file, instant-calculation
RESPONSE_LABEL	Label, das bei einmaliger Definition von anderen Items referenzierbar ist, welche die gleichen Antwortmgl. besitzen referenzierbar durch andere Items
RESPONSE_OPTIONS_TEXT	kommaseparierter Liste, die die mgl. Anwortwerte beinhaltet, die für den Einzutragenden interpretierbar sind
RESPONSE_VALUES_OR_CALCULATIONS	Wenn der RESPONSE_TYPE keine Berechnung ist, beinhaltet dieses Feld die Werte, die in der Datenbank gespeichert werden können. Bei einer Berechnung ist ein Ausdruck anzugeben, der Werte anderer Items des CRF's verwendet, um einen Wert zu ermitteln.
RESPONSE_LAYOUT	Ausrichtung der Antwortoptionen (Blank, Horizontal, Vertical)
DEFAULT_VALUE	Default Wert für das RESPONSE_OPTIONS_TEXT Feld
DATA_TYPE	Datentyp (ST, INT, REAL, DATE, PDATE, FILE)
WIDTH_DECIMAL	max. Anzahl der Zeichen und max. Anzahl der Dezimalstellen
VALIDATION	Validierungsausdruck
VALIDATION_ERROR_MESSAGE	Meldung bei Validierungsfehler
PHI	Signierung, ob Anzeige von geschützten Gesundheitsdaten (blank,0,1)
REQUIRED	Signierung, ob Wert vorhanden sein muss (blank,0,1)

Tabelle 1: Übersicht der Spalten der `Items` Tabelle

3 Realisierung

Die Applikation ist in Java implementiert und verwendet für das Auslesen der Excel Dateien die Apache POI [1] Bibliothek.
Der Unterabschnitt 3.1 beschreibt das zugrundeliegende Datenschema der Applikation. Im Unterabschnitt 3.2 wird der prinzipielle Ablauf der Applikation beschrieben. Abschließend wird detailliert auf die Berechnung der Unterschiede bzgl. einer Menge von CRF Versionen eingegangen.

3.1 Datenschema

Um einer potentiellen Veränderung der Excel- Dateispezifikation entgegenzuwirken, wurde ein eigenes generisches Datenschema für eine CRF Version konzipiert. Das Schema ist in Abbildung 2

[1] http://poi.apache.org/

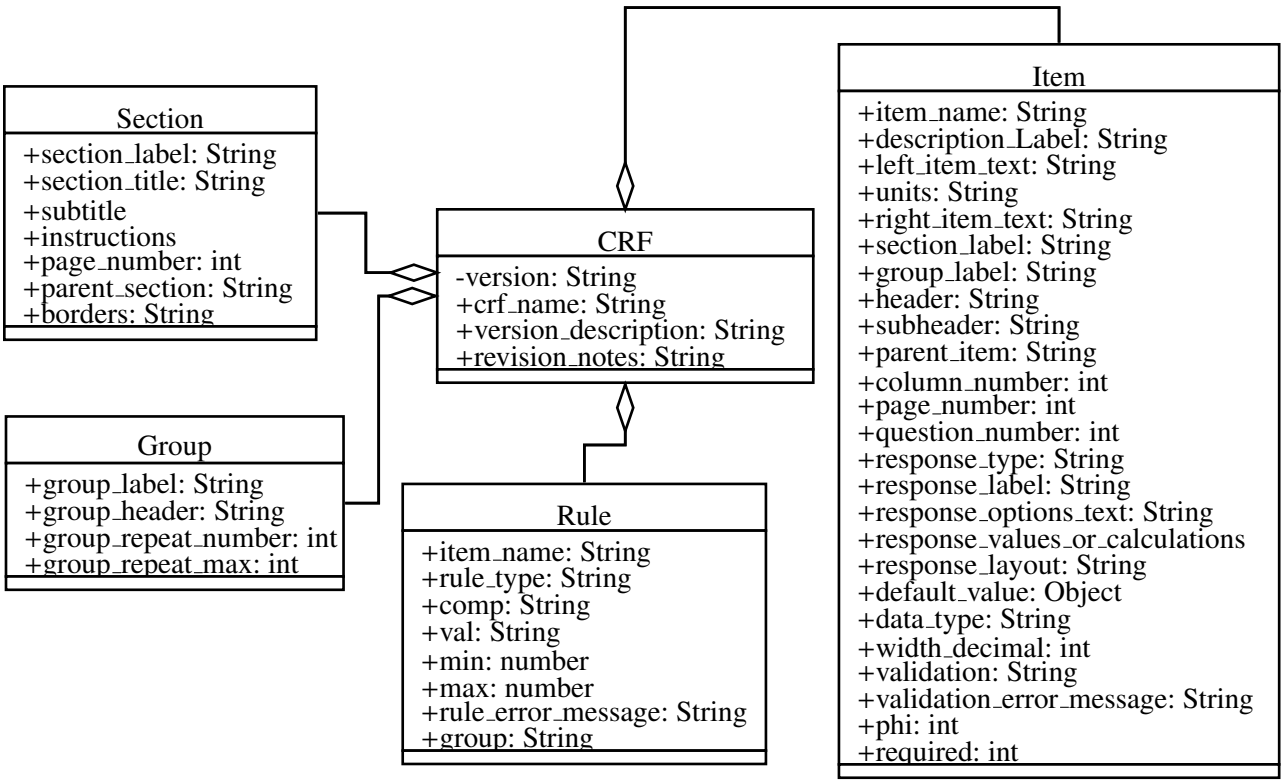

Abbildung 1: Datenschema der Excel Arbeitsmappe für einen CRF Bogen

dargestellt.
Die `CRFVersion` Klasse aggregiert die `Sections`, `Groups` und `Items` Tabelle in Form einer Hashmap mit dem jeweiligen `label` Attribut als Schlüssel und dem dazugehörigen Objekt der Klasse `Section`, `Group` bzw. `Item` als Wert. Die Klassen `Section`, `Group` und `Item` sind prinzipiell gleich aufgebaut. Als Identifier besitzen sie ein `(item|section|group)_label` Attribut und eine Hashmap `propertyMap`, die die Werte aller Spalten einer `Section`, einer `Group`, bzw. eines `Item` beinhaltet. Die Namen der Spalten fungieren als Schlüssel für die `propertyMap` und werden in der jeweiligen `(Item|Section|Group)Constants` Klasse gespeichert. Somit ist durch die lose Koppelung der Spalten zu einem Objekt mittels der `propertyMap` und der Konstanten eine Veränderung der CRF Excel- Spezifikation trivial realisierbar.

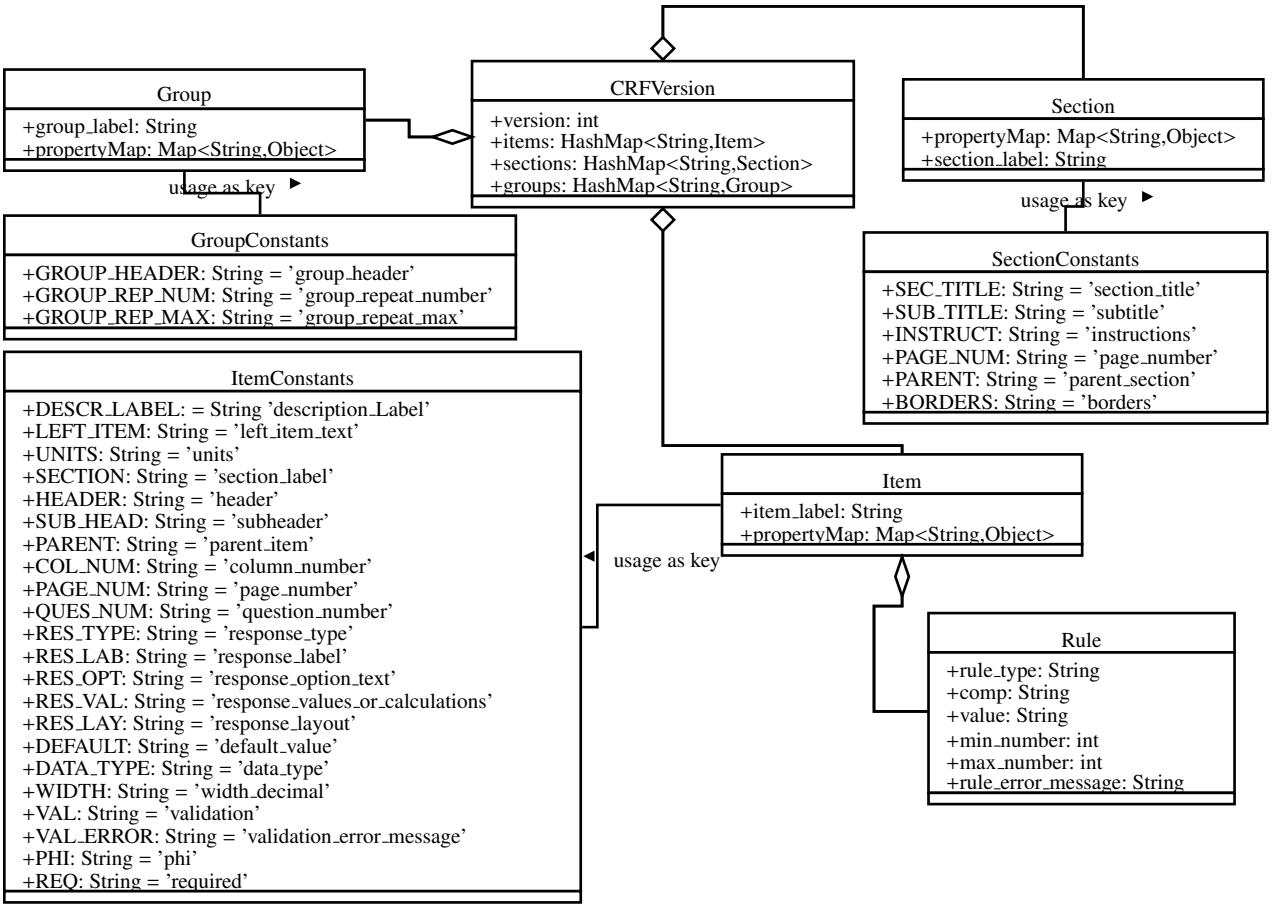

Abbildung 2: Klassendiagramm für eine CRFVersion

3.2 Allgemeiner Ablauf

Der allgemeine Ablauf der Applikationslogik ist in Abbildung 3 dargestellt. Zu Beginn spezifiziert der Anwender eine Menge von CRF Versionen. Die Dateien werden mittels der POI Bibliothek ausgelesen und zu Objekten des eigenen Datenschemas transformiert. Anschließend werden iterativ die Änderungen der CRF Versionen ermittelt und die Änderungen nach ihrer Relevanz klassifiziert. Bei jeder Iteration werden zwei aufeinander folgend Versionen miteinander verglichen. Des Weiteren werden in jeder Iteration die Änderungen der Eigenschaften der Items bzgl. der Intensität ihres Ausmaßes klassifiziert. Die Änderungen werden in Form eines Baumes visualisiert, indem die aktuelle Version als Bezugspunkt für alle Änderungen der CRF Versionen dient.

3.3 Diff-Berechnung

Die Menge der Änderungsoperationen wird als `Diff` bezeichnet. Die Ermittlung des Diff für eine Menge von CRF Versionen basiert auf dem fortlaufenden Vergleich von zwei CRF Versionen. Im Folgenden wird das Verfahren für die Ermittlung näher beschrieben.

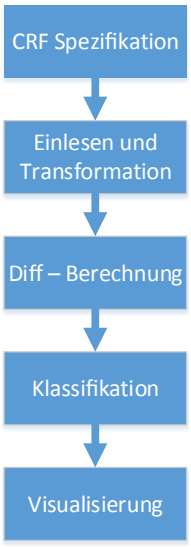

Abbildung 3: Workflow der Applikation

Die Eingabe der Methode sind zwei CRF Versionen. Das Ergebnis ist eine Menge von gleichen, gelöschten, hinzugefügten und geänderten Items. Die geänderten Items werden als Mapping von altem Item zu neuem Item repräsentiert. Des Weiteren wird für jedes Mapping die Menge der geänderten Eigenschaften sowie der alte und der neue Wert gespeichert. Das Vorgehen ist im Algorithmus 1 dargestellt. Das Prinzip ist ähnlich wie bei der Berechnung des Diffs für zwei Ontologieversionen wie in [3]. Zunächst werden alle gleichen Entitäten ermittelt und danach alle alten und neuen Entitäten. Jedes Item der alten Version wird mit jedem Item der neuen Version verglichen, indem die `item_labels` und die`propertyMap` Objekte verglichen werden (Zeile 3). Wenn diese gleich sind, sind die Items gleich und ein Repräsentant wird zu der Menge *eqItemSet*, die die Menge der gleichen Items repräsentiert, hinzugefügt (Zeile 4). Nachdem alle gleichen Items ermittelt wurden, werden die gelöschten Items bzw. die hinzugefügten Items ermittelt. Ein Item ist gelöscht bzw. hinzugefügt, wenn es ausschließlich in der alten Version bzw. in der neuen Version existiert. Die Ermittlung erfolgt durch die Iteration über alle Items der alten Version (resp. neuen Version) und alle Items der Menge *eqItemSet*, wenn das alte Item (resp. neue Item) keine Korrespondenz zu einem Item der Menge *eqItemSet* aufweist (Zeile 7 resp. Zeile 11), wird es zu der Menge der gelöschten Items *delItemSet* (Zeile 8) bzw. zu der Menge der neuen Items *addItemSet* (Zeile 12) hinzugefügt. Die beiden Mengen enthalten jedoch ebenfalls modifizierte Items, da die Ermittlung der Korrespondenz auf der Gleichheit des Item Labels basiert. Da geänderte Items sich durch einen anderen Suffix beim `item_label` unterscheiden, befinden sich in den Mengen *delItemSet* und *addItemSet* eventuell geänderte Items. Um die Menge der geänderten Items zu ermitteln, wird mithilfe eines Fuzzy- Matchers die Ähnlichkeit für jedes Item der *delItemSet* Menge und jedem Item der *addItemSet* Menge bestimmt (Zeile 13-23). Der Fuzzy- Matcher verwendet für die Berechnung der Ähnlichkeit eine Kombination der Editierdistanz [4] für die `item_labels` und die `description_labels`. Als geändertes Item wird das Paar angesehen, dessen Ähnlichkeit größer als ein Threshold $\delta(0.6)$ ist und das die höchste Ähnlichkeit aufweist. Die Intention der Ver-

wendung des Fuzzy- Matchers ist die Berücksichtigung einer mgl. minimalen Änderung des Item-Labels. Es wird dabei folgendes Szenario betrachtet. Ein Item wird modifiziert, jedoch enthält das Label einen Rechtschreibfehler. Durch die Behebung des Rechtschreibfehlers ist die Modifikation nicht durch die ausschließliche Betrachtung des Suffixes erkennbar. Nach der Identifikation der modifizierten Items wird für jedes Paar die Menge der geänderten Eigenschaften, die die Spalten repräsentieren, ermittelt.
Diese Methode wird iterativ für die nächsten beiden Versionen fortgeführt, wobei die aktuelle neue Version, die alte Version ist und die darauffolgende die neue. Die Berechnung des Diff's ist abgeschlossen, wenn alle Versionen verarbeitet sind.

Algorithm 1: Diff Berechnung für zwei CRF Versionen

input : $oldVersion$, $newVersion$
output: $eqItemSet$, $addItemSet$, $delItemSet$, $modItemMap$, $propertyMap$

```
1  for oldItem ∈ oldVersion do
2      for newItem ∈ newVersion do
3          if oldItem.equals(newItem) then
4              eqItemSet ← eqItemSet ∪newItem
5  for oldItem ∈ oldVersion do
6      for equalItem ∈ eqItemSet do
7          if oldItem.item_label ≠ equalItem.item_label then
8              delItemSet ← delItemSet ∪ oldItem
9  for newItem ∈ newVersion do
10     for equalItem ∈ eqItemSet do
11         if newItem.item_label ≠ equalItem.item_label then
12             addItemSet ← addItemSet ∪ newItem
13 for addItem ∈ addItemSet do
14     topMapping ← ∅
15     for delItem ∈ delItemSet do
16         similarity = fuzzyMatch(addItem, delItem)
17         if similarity ≥ δ then
18             topMapping ← add(similarity, (delItem, addItem))
19         if topMapping ≠ ∅ then
20             (delItem,addItem) = bestMapping(topMapping)
21             modItemMap ← add(delItem,addItem)
22             addItemSet = addItemSet \ { addItem }
23             delItemSet = delItemSet \ { delItem }
24 for (oldItem, newItem) ∈ modItemMap do
25     oldPropertySet = oldItem.properties
26     newPropertySet = newItem.properties
27     modProperties = diff(oldPropertySet,newPropertySet)
28     propertyMap ← add(modProperties)
```

3.4 Klassifikation

Um die Auswirkungen der geänderten Datenfelder abzuschätzen, ist eine Einordnung dieser nach spezifizierten Kategorien erforderlich. Die Kategorien sind vordefiniert und orientieren sich an der möglichen Ausprägung der Semantik eines Items. Es existieren folgende Klassen:

- *semantic*
  Diese Kategorie umfasst Änderungen, die durch die Modifikation der Beschreibung, der Header oder der linken bzw. rechten Textlabels hervorgerufen werden. Es wird angenommen, dass eine solche Änderung eine Veränderung des semantischen Konzepts als Ursache hat.

- *semantic specialization*
  Diese Kategorie ist eine Erweiterung der *semantic* Kategorie. Wenn der Inhalt durch einen Suffix erweitert wurde, wird angenommen, dass es sich um eine Spezialisierung handelt.

- *response range*
  Diese Kategorie repräsentiert alle Änderungen, die einen Einfluss auf die Spezifikation der möglichen Antworten des Datenfelds haben.
  Die Eigenschaften, die davon betroffen sind:
  DATA_TYPE, DEFAULT_VALUE, RESPONSE_LABEL, RESPONSE_OPTIONS_TEXT, RESPONSE_TYPE,REQUIRED, RESPONSE_VALUES_OR_CALCULATIONS und UNITS.

- *item interrelation*
  Eine Änderungen, die die Beziehung bzgl. eines anderen Items ändert, wird zu dieser Kategorie zugeordnet.
  Die Eigenschaften, die davon betroffen sind:
  GROUP_LABEL und PARENT_ITEM.

- *validation*
  Diese Kategorie umfasst alle spezifizierenden Eigenschaften, die die Validierung der Daten festlegen oder Sicherheitseinstellungen bzgl. der Sichtbarkeit der Daten spezifizieren.
  Die Eigenschaften, die davon betroffen sind:
  VALIDATION, VALIDATION_ERROR_MESSAGE und PHI.

- *small editing*
  Die Änderungen, die keinen Einfluss auf die Eigenschaft haben, sondern lediglich eine Korrektur oder eine andere Codierung realisieren, werden dieser Kategorie zugeordnet.

- *layout*
  Diese Kategorie beinhaltet alle Änderungen, die sich auf die Anordnung und die Strukturierung der Datenfelder beziehen.

4 Installation und Konfiguration

Die Applikation ist frei verfügbar und kostenlos unter https://github.com/vicolinho/crfMapping herunterladbar. Die Applikation erfordert die Java Runtime Enviroment 1.7. Für die Weiterentwicklung der Applikation ist ein eigenes Java Projekt zu erstellen mit dem 'src' Ordner als Quellcodeordner. Des Weiteren sind alle Bibliotheken des 'lib' Ordner in den Klassenpfad einzutragen. Das Programm lässt sich als Kommandozeilenprogramm ausführen mithilfe der 'config.properties' Datei. Diese Variante ermittelt für zwei CRF Versionen, die in der Konfigurationsdatei spezifiziert sind, die hinzugefügten, gelöschten und modifizierten Items. Die Applikation wird mithilfe von `ant runDiffCalculator` gestartet.
Um die Applikation mit einer GUI zu starten ist der Aufruf `ant DiffCRFTool` erforderlich oder die Jar- Datei 'CRFDiffTool' auszuführen. Im Allgemeinen ist folgender Ablauf vorgesehen:

1. Spezifikation der CRF Version im Menü 'File/Open'
2. Auf der linken Seite können alle CRF Versionen in einer Baumstruktur betrachtet werden. Auf der rechten Seite können die Änderungen berechnet und angezeigt werden. Des Weiteren ist die Anzeige der Änderungen filterbar bzgl. des Versionenintervalls und der Änderungskategorie.

5 Funktionsweise

Die Applikation besitzt drei basale Funktionalitäten, die Spezifikation der CRF Versionen, die Betrachtung der einzelnen CRF Versionen und die Betrachtung der Unterschiede der CRF Versionen im Bezug auf die aktuelle Version.
Die Spezifikation der CRF Version erfolgt im Menü File/Open. Für die Betrachtung der Unterschiede müssen mindestens zwei Versionen eines CRF's selektiert werden.
Der Anwender ist in der Lage sich die einzelnen CRF Versionen einzeln zu betrachten (siehe Abb. 4), indem er eine Version in der ComboBox selektiert und hinzufügt. Die selektierte Version wird als Baum, deren Blätter die Items repräsentieren, auf der linken Seite der Applikation dargestellt. Bei der Selektion eines Items können die Eigenschaften im unteren Teil der Anwendung betrachtet werden.
Die Änderungen werden auf der rechten Seite des Programms dargestellt(siehe Abb. 5). Die Items der aktuellen Version werden in einem Baum dargestellt. Die geänderten Items werden farblich hervor gehoben. Des Weiteren werden alle Änderungen, die ein Item betreffen chronologisch in einer Tabelle als Blattknoten des betreffenden Items dargestellt. Die Tabelle enthält die Versionsnummer von der vorherigen Version, den Namen des alten Items und die Art der Änderung. Bei der Selektion eines Eintrages der Tabelle, werden die geänderten Eigenschaften im unteren Teil in einer Tabelle präsentiert. Diese Tabelle enthält den Namen der geänderten Eigenschaft, den alten und neuen Wert und die Kategorie der Änderung. Der Anwender ist in der Lage die geänderten Items nach der Kategorie der Änderung zu filtern, indem er im Menü *category selection* die jeweiligen Kategorien selektiert. Des Weiteren ist das Intervall der zu betrachtenden Versionen mittels der Combo- Boxen spezifizierbar, so dass der Anwender nicht gezwungen ist die ganze Historie

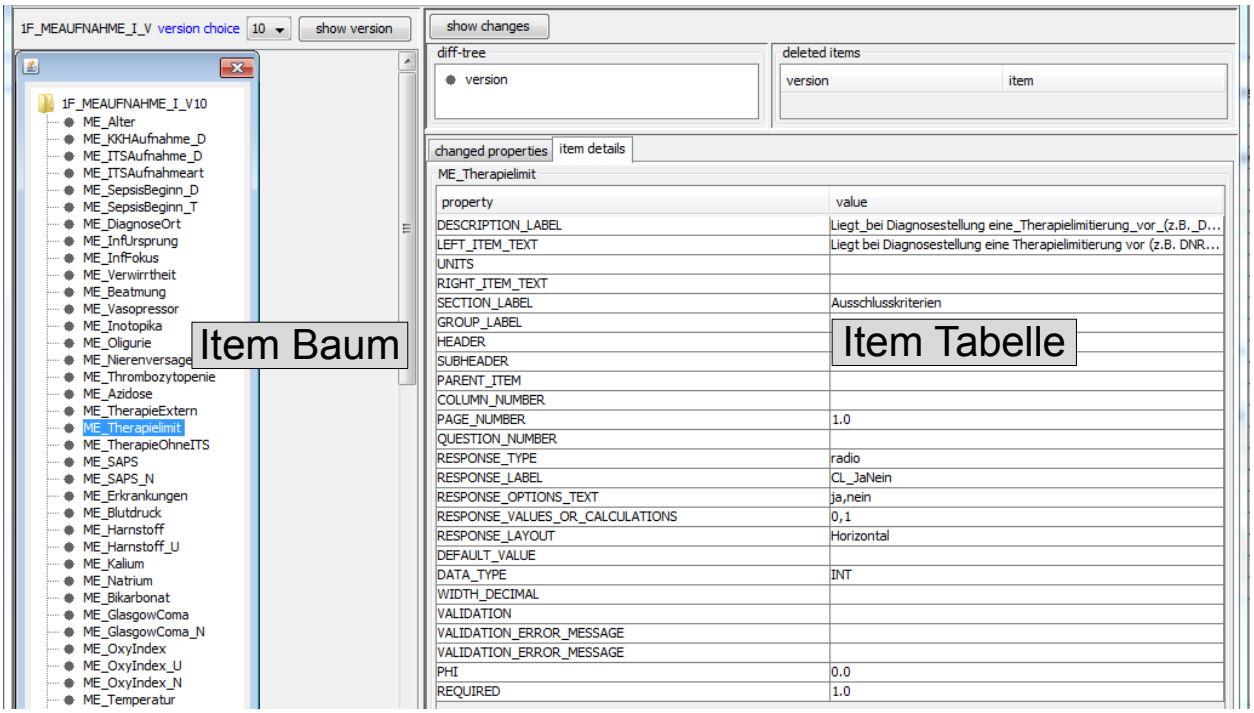

Abbildung 4: Darstellung der Items einer CRF Version in einem Baum inklusive der Ansicht der Eigenschaften eines Items in einer Tabelle

zu analysieren. Die gelöschten Items werden in einer separaten Tabelle chronologisch nach ihrem Löschzeitpunkt präsentiert, da diese in der aktuellen Version nicht vorhanden sind.

Im Folgenden wird die Funktion der Änderungspräsentation anhand der CRF Versionen für die Medusa Studie in der Aufnahmephase veranschaulicht. Diese Phase umfasst 5 Versionen. Zu Beginn werden alle hinzugefügten und gelöschten Items ermittelt inklusive der Version der Änderung. Abschließend werden alle Items ermittelt, deren Spezifikation sich geändert haben.

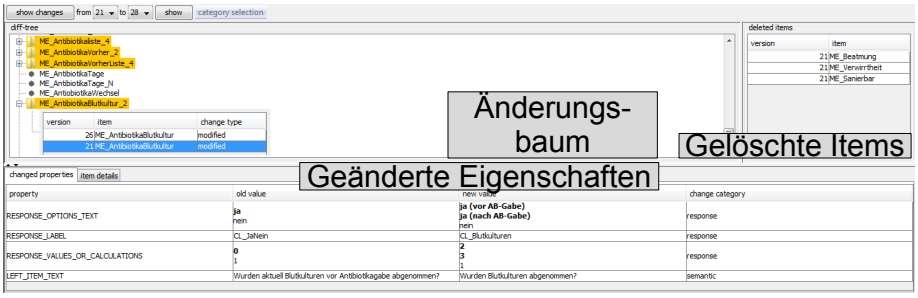

Abbildung 5: Darstellung der Änderung der geänderten Items und der geänderten Eigenschaften

5.1 Hinzugefügte und Gelöschte Items

Diese Art der Items ist bei der Auswertung nicht problematisch, da die Daten bei hinzugefügten Items (resp. gelöschten Item) in die Auswertung einfließen bzw. nicht berücksichtigt werden. Die Menge der hinzugefügten Items ist in der Tabelle 2 dargestellt.

Tabelle 2: Menge der hinzugefügten Items inklusive der Version, in der das Item hinzugefügt wurde

Version	Item
21	ME_Groesse
21	ME_Groesse_N
21	ME_Gewicht_N
21	ME_Vigilanzstoerung
21	ME_Oxygenierungsstoerung
21	ME_Hypotonie
21	ME_Basendefizit
21	ME_Basendefizit_N
21	ME_FokusErfolg

Die Tabelle 3 veranschaulicht die gelöschten Items inklusive der Version, in der sie gültig waren. Die Daten dieser Items werden bei der Auswertung der Studie voraussichtlich nicht berücksichtigt.

Tabelle 3: Menge der gelöschten Items inklusive der Version, in der ein Item gültig war

Version	Item
10	ME_Tachypnoe
10	ME_AntibiotikaVorher_N
10	ME_Tachypnoe_N
21	ME_Sanierbar
21	ME_Verwirrtheit
21	ME_Beatmung

5.2 Geänderte Items

Im Folgenden werden alle geänderten Items identifiziert und die Änderungen für eine Teilmenge dieser Items dargestellt. Die Präsentation der Änderungen ist itemweise, da der Biometriker die Daten einer Menge von gemappten Items betrachten muss und somit eine versionsbezogene Visualisierung der Änderungen nicht geeignet ist.
Bei der Identifikation der geänderten Items ist auffällig, dass viele Änderungen bzgl. zwei Versionen lediglich eine Änderung des *Item Labels* beinhalten, wie z.B. ME_InfUrsprung_2 von Version 26 auf Version 28 ist keine Veränderung identifiziert wurden, außer beim Label. Die Ursache ist vermutlich die nicht strikte Einhaltung der Konvention, dass bei einer inhaltlichen Veränderung des Items der entsprechende Suffix für die Erweiterung des Labels verwendet wird. Die geänderten Items über alle Versionen sind in der Tabelle 4 dargestellt.

Da die Menge der geänderten Items relativ hoch ist, wird eine Teilmenge der modifizierten Items betrachtet. Für jede Kategorie wird ein Repräsentant aufgeführt.

Tabelle 4: Menge der geänderten Items inklusive der Versionen, in der die alten Items gültig waren. Die Reihenfolge der Versionen ist chronologisch bzgl. des Auftretens der Änderungen

Version	Item
10	ME_Alter
21, 26	ME_InfUrsprung_2
20	ME_Therapielimit
10	ME_SAPS
10, 21	ME_Erkrankungen
10, 21	ME_Blutdruck_2
10, 21	ME_Harnstoff
21	ME_Harnstoff
10, 21	ME_Kalium_2
10, 21	ME_Natrium_2
10, 21	ME_Bikarbonat_2
21	ME_GlasgowComa
10, 21	ME_OxyIndex_2
10, 21	ME_Kreislauf_3
10	ME_Diurese
10	ME_Kreatinin
10	ME_Bilirubin
21, 26	ME_Thrombozyten_U_2
10	ME_Leukozyten
21, 26	ME_Leukozyten_U_2
10	ME_CRP
21, 26	ME_CRP_U_2
10	ME_Procalcitonin
10	ME_Laktat
21	ME_Laktat_U_2
10	ME_PhWert
10, 21, 26	ME_Antibiotikaliste_4
10, 21	ME_AntibiotikaVorher_2
21, 26	ME_AntibiotikaVorherListe_4
10	ME_AntibitikaTage
21, 26	ME_AntibitikaBlutkultur_2

- *ME_Harnstoff* (semantic)
  Die Änderung umfasst eine inhaltliche Spezifikation der Genauigkeit bzgl. der Angabe der Daten. Dies wird durch die Änderung des 'RIGHT ITEM TEXT' Feldes realisiert.

- *ME_GlasgowComa* (semantic specialization)
  Um eine Fehlinterpretation beim Ausfüllen des CRF zu vermeiden bzgl. des Items *ME_GlasgowComa*, wird das Feld 'LEFT ITEM TEXT' für dieses Item erweitert, von 'Glasgow Coma Scale' auf 'Glasgow Coma Scale (ohne Sedierung)'.

- *ME_InfUrsprung_2* (response)
  Die Antwortmöglichkeiten werden von der Version 10 zu der Version 21 spezifiziert, indem die Antwortmöglichkeit 'nosokomial (Normalstation/Pflegeheim)' zu zwei Antworten aufgetrennt wird. Diesbezüglich werden ebenfalls die Wertlisten erweitert, die im Studienma-

nagementsystem verwendet werden. Die Änderungen von Version 21 zu Version 26 umfasst lediglich die Änderung des Item Labels.

- *ME_SAPS*(validation)
  Bei diesem Item wird die Validierungsfuktion verändert von 'func: lte(163)' zu 'func:range(0,163)', so dass Werte unter 0 ausgeschlossen werden. Dementsprechend ändert sich das 'VALIDATION_ERROR_MESSAGE' Feld.

- *ME_Erkrankungen*(editing)
  Dieses Item enthält eine Änderung der Codierung bzgl. der Anführungszeichen für den Wert des SUB HEADER Feldes.

- *ME_Natrium_2* (layout)
  Dieses Item ist exemplarisch für die Änderung des Layouts. Bei allen betrachteten CRF Versionen ist eine Änderung bzgl. des Layouts eine Veränderung der Seite von '3' auf '4' feststellbar.

6 Diskussion

Im Folgenden werden die erzielten Ergebnisse mit den gesetzten Zielen verglichen sowie weitere Erweiterungsmöglichkeiten präsentiert.

1. Die Aufgabe des Tools ist die Unterstützung des Biometrikers beim Auswertungsprozess. Diesbezüglich muss er wissen, welche Items aufeinander gemappt werden. Somit muss das Tool effizient sein, so dass der zeitliche Arbeitsprozess des Biometrikers nicht unnötig erhöht wird. Die Laufzeit der Berechnung des Diffs ist sehr gering, so beträgt die benötigte Zeit für die fünf CRF Versionen der Medusa Studie für die Aufnahmephase lediglich 78 ms.

2. Bei der Realisierung der Applikation wurde versucht eine hohe Benutzerfreundlichkeit zu realisieren, indem sich auf die wesentlichsten Funktionen beschränkt wurde. Alle möglichen Änderungen sind sofort erkennbar. Gelöschte Items befinden sich auf der rechten Seite der Applikation. Hinzugefügte Items befinden sich im Versionenbaum und sind durch ein 'Plus' Symbol erkennbar. Des Weiteren sind geänderte Items durch die farbliche Markierung sofort erkennbar. Mithilfe der Filterfunktionen bzgl. der angezeigten Versionen und der Kategorien ist eine einfache Identifizierung der Änderungen möglich.

3. Die Klassifizierung der Änderungen bietet eine grobe Einstufung des Schweregrads. Jedoch sind keine detaillierten Aussagen ableitbar, da zum größten Teil die Klassifizierung abhängig ist von der geänderten Spalte der Excel- Tabelle und nicht von der semantischen Änderung. Diesbezüglich wäre für zukünftige Arbeiten eine semantische Klassifizierung zu realisieren.

Das Tool ist ausschließlich auf die Visualisierung der Änderungen ausgelegt. In zukünftigen Arbeiten wäre eine Exportfunktion bzgl. der Änderungen in eine Excel- Mappe sinnvoll, so dass ein Austausch der Änderungsinformationen in kompakter Form statt finden kann, ohne das Tool ausführen zu müssen.

7 Zusammenfassung

Diese Arbeit befasste sich mit der Entwicklung eines Tools, welches ein Mapping für eine Menge von CRF Versionen generiert. Ein Mapping repräsentiert eine Abbildung von alten Datenelementen auf die entsprechenden neuen Datenelemente der CRF Versionen.
Generell werden CRFs verwendet, um auswertungsrelevante Daten für jede Phase einer klinischen Studie zu erfassen. Jedoch ist ein CRF Bogen nicht statisch, da stetig Verbesserungen oder bestehende Fehler realisiert bzw. korrigiert werden. Für die Auswertung einer Studie, die mehrere CRF Versionen umfasst, ist es notwendig die geänderten Items zu identifizieren, um die Daten der gemappten Items auswerten zu können. Mithilfe des Tools ist eine effektive und effiziente Identifikation aller Änderungen bzgl. der betrachteten CRF Versionen möglich. Da für das Mapping der Daten nicht alle Änderungen als problematisch angesehen werden müssen, ist eine Klassifikation hilfreich für die Filterung der problematischen Items. Das Tool unterstützt eine triviale Klassifizierung abhängig vom geänderten Attribut des Items.

Literatur

[1] Akaza Research. OpenClinica. https://community.openclinica.com/. Accessed: 2014-04-29.

[2] Hugo Leroux, Simon McBride, and Simon Gibson. On selecting a clinical trial management system for large scale, multi-centre, multi-modal clinical research study. *Stud Health Technol Inform*, 168:89–95, 2011.

[3] Michael Hartung, Anika Groß, and Erhard Rahm. Conto–diff: generation of complex evolution mappings for life science ontologies. *Journal of Biomedical Informatics*, 46(1):15–32, 2013.

[4] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.