Design eines Mappings und Implementierung eines Converters von LimeSurvey Umfragen und Antworten in CDISC ODM

Bachelorarbeit

vorgelegt von:

Antonius Johannes Mende

Matrikelnummer: 461 328

Studiengang: Informatik 1.FB

Thema gestellt von:

Dr. Ludger Becker

Arbeit betreut durch:

Dr. Ludger Becker und Dr. Tobias Brix

Münster, 7. Juni 2021

Inhaltsverzeichnis

1 Einleitung

Das Thema dieser Arbeit ist die Konvertierung von LimeSurvey Archiven in das Operational Data Model. LimeSurvey ist ein Werkzeug, mit dem man einfach Umfragen erstellen kann, welche dann wiederum von beliebig vielen Teilnehmern beantwortet werden können. Umfragewerkzeuge lassen sich sehr vielseitig einsetzen, um die Meinungen von Kunden, Patienten oder jedem anderen Menschen einzuholen. Gerade in den heutigen Zeiten, wo immer mehr Daten gesammelt und verarbeitet werden können, ist es wichtig, Kompatibilität zwischen verschiedenen Werkzeugen und Formaten herstellen zu können. Das Institut für Medizinische Informatik an der WWU ist eine der vielen Institutionen, welche LimeSurvey nutzt, um die Meinungen und Erfahrungen von Patienten einzuholen.

1.1 Problem

Der XML-Standard lässt dem Programmierer sehr viel Freiheit, was das Design eines XML-Formates angeht. Dadurch sind verschiedene XML-Formate im Regelfall nicht kompatibel beziehungsweise untereinander austauschbar. Das macht es notwendig, XML-Dokumente zu konvertieren, um ein Dokument in einem Format mit einem Werkzeug nutzen zu können, welches nur ein anderes XML-Format unterstützt. Als Zielformat wird ODM-XML von CDISC verwendet. ODM wurde mit dem Ziel entwickelt, den Austausch und die Archivierung von Forschungsdaten und anderen damit verbundenen Daten zu ermöglichen. Durch die Unabhängigkeit des Formates von spezifischen Plattformen oder Firmen wird es durch weit mehr Werkzeuge unterstützt als ein Format wie LSA, welches von LimeSurvey selbst und zu deren eigenen Zwecken entwickelt wurde. Dies sieht man auch an dem Institut für Medizinische Informatik, welche dieses Projekt betreuen. Dort wird ODM täglich zur Verwaltung klinischer Daten genutzt, gleichzeitig ist aber auch LimeSurvey das Umfragewerkzeug der Wahl, um Daten von Patienten zu sammeln. Diese gesammelten Daten müssen dann exportiert und konvertiert werden, um sie in bestehenden Systemen einpflegen zu können.

1.2 Ziel

Ziel der Arbeit ist es, genau darzustellen, wie man aus einem LimeSurvey Archiv in ein ODM Dokument umwandeln kann und einen Konverter zu imple-

mentieren, welcher das bewerkstelligt. Es sollen so viele der Forschungsdaten wie möglich übertragen werden, zusätzliche Daten, welche z.B. die Darstellung der Daten betreffen, sollen nicht mit übertragen werden.

2 Methodik

2.1 Akronyme

EDC „*Electronic Data Capture*“ Das Sammeln und Verarbeiten von Daten

RegEx „*Regular Expression*“ Ein Ausdruck, der genutzt werden kann, um Zeichenketten auf eine bestimmte Struktur zu überprüfen

IMI „*Institut für Medizinische Informatik*“ Eines der Institute der WWU. Diese Arbeit ist in Kooperation mit ihnen entstanden

XML „*Extensible Markup Language*“

XSD „*XML Schema Definition*“ Eine Datei, welche beschreibt, wie ein XML-Dokument aufgebaut sein sollte, um einer bestimmten Definition zu entsprechen.

SAX „*Simple API for XML*“ Ein Standard, welcher beschreibt, wie man ein XML-Dokument parsen kann. Dieses wird sequentiell eingelesen und für definierte Ereignisse wird eine vorgegebene Rückruf-Funktion aufgerufen. Ein Programm kann eigene Funktionen registrieren und so das Dokument verarbeiten.

DOM „*Document Object Model*“ Bietet die Möglichkeit, die Hierarchie der XML-Knoten in Baumform darzustellen und so zu navigieren/ den Baum zu bearbeiten

CDISC „*Clinical Data Interchange Standards Consortium*“

CDISC ODM „*Operational Data Model*“ von CDISC entwickeltes XML-Format (siehe Abschnitt 2.3)

lsa Dateiendung des LimeSurvey Archives (siehe Abschnitt 2.2.4)

lsr Dateiendung der LimeSurvey Response Datei (siehe Abschnitt 2.2.4)

lss Dateiendung der LimeSurvey Struktur Datei (siehe Abschnitt 2.2.4)

2.2 LimeSurvey

LimeSurvey ist ein von der gleichnamigen deutschen Firma entwickeltes Werkzeug für Umfragen. Laut ihrer Website ist LimeSurvey ein für Einsteiger und Profis sowie für Privatpersonen als auch Institutionen gut geeignetes Werkzeug, um die Meinungen, Interessen und Entscheidungsgrundlagen einer Zielgruppe herauszufinden. Es wird seit 2006 an der Software entwickelt und sie ist sowohl als Cloud-Edition über die Firma erhältlich, sowie auch als OpenSource-Projekt.

Falls man die Cloud-Edition nutzen will, hat man fünf Optionen: *Free*, *Basic*, *Expert*, *Enterprise* und *Corporate*. Diese unterscheiden sich vor allem durch die Zahl an Antworten pro Monat, die Menge an Upload-Speicher, E-Mail-Support, White-Label-Umfragen, Alias-Domains und die Entfernung des LimeSurvey-Brandings. Innerhalb der *Corporate*-Lösung kann man noch dedizierte Server und die Nutzung von SAML beantragen. Preislich bewegt man sich in einem Rahmen von 0€-74€ für die normalen Lösungen und einem individuell absprechbaren Preis für die *Corporate*-Lösung.

2.2.1 Features

ExpressionScript

Mit Expression Script lässt sich komplexe Logik in die Umfrage einbringen. Die LimeSurvey-interne Skriptsprache lässt Bedingungen zu, unter denen bestimmte Fragen oder Antworten angezeigt werden sollen. Man kann hier sowohl Antworten auf vorherige Fragen einbinden, als auch Informationen über den Teilnehmer, welche er vorher angegeben hat beziehungsweise welche über ihn gespeichert wurden. Man kann mehrere Szenarios designen und es sind Vergleiche mit den Standard-Operatoren sowie RegEx möglich.

Timings

Es ist möglich genau festzulegen, wie viel Zeit ein Nutzer zum Beantworten einer Frage hat. Es können Warnmeldungen zu bestimmten Abschnitten innerhalb der Zeitperiode angezeigt werden, die CSS-Klasse kann angepasst werden und das Beantworten anderer Fragen kann unterbunden werden.

2.2.2 Fragegruppen

Fragen in LimeSurvey sind in Fragegruppen unterteilt. Jede Frage ist genau einer Gruppe zugeordnet. Eine Gruppe kann beliebig viele Fragen enthalten, weiterhin hat sie einen Titel, eine Beschreibung, eine Randomisierungsgruppe und eine Relevanz-Gleichung, wo mittels ExpressionScript (siehe Abschnitt 2.2.1)

angegeben werden kann, wann die Fragen dieser Gruppe angezeigt werden sollen. Es darf beliebig viele Fragegruppen geben.

2.2.3 Fragetypen

Es gibt 36 Fragetypen in LimeSurvey, welche in fünf Kategorien unterteilt sind. Davon sind allerdings nicht alle tatsächlich Fragen, auf die der Teilnehmer antworten kann, insgesamt gibt es vier Fragetypen, welche nicht explizit Fragen sind. Die möglichen Antworten auf Freitext- und Zahlenfragen lassen sich mittels RegEx limitieren. Es gibt pro Frage einen optionalen Hilfstext. Fragen mit vordefinierten Antworten erhalten eine „Keine Antwort“-Option, wenn sie nicht verpflichtend sind. Im folgenden sollen alle Typen einmal aufgelistet und kurz beschrieben werden.

Einfachauswahl

Auf folgende Fragen kann man maximal eine Antwort geben.

5 Punkte Wahl: Hier kann auf einer Skala von 1 bis 5 ein Wert ausgewählt werden

Liste: Hier kann aus einer vordefinierten Liste eine Antwort gewählt werden. Es gibt drei unterschiedliche Darstellungsmöglichkeiten für diese Liste, entweder als Dropdown-Menü, als Bootstrap-Buttons oder mit Radio-Buttons neben den Antworten

Liste mit Kommentar: Dies ist eine Listen-Frage wie oben, allerdings kann für die Frage auch noch ein Kommentar im Freitext geschrieben werden

Image-Select-List: Hier wird zu der Liste an Antworten noch ein Bild angezeigt. Die Antworten werden mit Radio-Buttons daneben angezeigt

Matrix

Pro Frage gibt es beliebig viele Subfragen, für jede Subfrage kann eine der vordefinierten Antworten ausgewählt werden. Subfragen werden typischerweise als Zeilen dargestellt, die Antworten als Spalten

5 Punkte: Hier kann auf einer Skala von 1 bis 5 ein Wert ausgewählt werden

10 Punkte: Hier kann auf einer Skala vom 1 bis 10 ein Wert ausgewählt werden

Z/G/A: Hier kann aus den drei Möglichkeiten „Zunahme/Gleich/Abnahme“ eine gewählt werden

J/N/U: Hier kann aus den drei Möglichkeiten „Ja/Nein/Unsicher“ eine gewählt werden

Matrix (Custom): Hier kann eine eigene Liste an Antwortmöglichkeiten definiert werden

Matrix nach Spalte Identisch zu Matrix (Custom), allerdings werden Zeilen und Spalten getauscht

Dual Matrix Es gibt zwei selbst erstellbare Listen an Antwortmöglichkeiten, man kann aus beiden eine Antwort pro Subfrage wählen

Matrix (Freitext) Hier kann man zwei Listen an Subfragen angeben. Daraus wird dann eine Matrix erstellt, wo in jeder Zelle Freitext als Antwort geschrieben werden kann

Matrix (Zahlen) Identisch zu Matrix (Freitext), aber es können nur Zahlen als Antwort angegeben werden

Multiple Choice

Hier können mehrere Antworten aus einer vordefinierten Liste ausgewählt werden. Optional kann ein „Anderes“-Feld zugänglich gemacht werden, wo Teilnehmer eine eigene Antwort schreiben können

Multiple Choice Darstellung als Bootstrap-Buttons oder Radio-List ist möglich

Image Select Hier wird ein Bild mit angezeigt

Kommentar hier kann ein Kommentar pro Antwortfeld geschrieben werden

Textfragen

Browser Detect Erkennt den Webbrowser des Teilnehmers. Dies ist keine Frage, welche der Teilnehmer beantworten kann

Freitext Hier kann der Teilnehmer einen Text eintippen. Es gibt kurze, lange und riesige Freitexte

Mehrere Texte Es gibt mehrere Subfragen. Für jede kann ein kurzer Freitext angegeben werden

Input on Demand

Maskenfragen

Datum/Zeit Hier kann ein Datum und eine Uhrzeit angegeben werden. Das Format ist einstellbar, es kann auch ein minimales Datum angegeben werden

Ja/Nein Hier kann mit Ja oder Nein geantwortet werden

Gleichung Hier kann ein Text angegeben werden, wo der Nutzer Variablen einträgt, diese können in einer Rechnung verwendet werden, dessen Ergebnis dann angezeigt wird

Dateiupload Hier kann eine Datei als Antwort hochgeladen werden

Geschlecht Hier kann zwischen „Männlich“, „Weiblich“ und „Keine Antwort“ ausgewählt werden

Sprachumschaltung Hier kann der Teilnehmer die Sprache ändern. Auch das ist keine richtige Frage

Zahleneingabe Hier kann nach einer Zahl als Antwort gefragt werden.

Mehrfache Zahlen Hier gibt es mehrere Subfragen, auf die jeweils mit einer Zahl geantwortet werden muss. Die Maximal/Minimal-Werte sind limitierbar, genau so wie auch die Maximal- und Minimal-Summe. Es können genaue Gesamtergebnisse verlangt werden und nur Ganzzahlen als Antwort verlangt werden

Ranking (Advanced) Hier kann der Teilnehmer Elemente aus einer vordifinierten Liste sortieren. Es müssen nicht alle Elemente einsortiert werden

Textanzeige Hier wird ein Text angezeigt. Auch hier kann nicht geantwortet werden

2.2.4 Export

LimeSurvey Archiv

Das LimeSurvey Archiv ist eine von sieben Möglichkeiten, Daten von einer LimeSurvey Umfrage zu exportieren. Ein solches Archiv ist eine komprimierte Datei im .lsa-Format, welche mehrere extrahierbare Dateien enthält. Die Zahl und Art der Dateien ist dabei abhängig von den Einstellungen. Zwei Dateien sind immer enthalten:
Die erste Datei enthält die Umfrage-Struktur sowie Informationen über die Art und Weise, wie die Fragen darstellt werden sollen (die .lss-Datei), die zweite Datei enthält die Antworten der Teilnehmer (.lsr-Datei). Ausdrücklich erwähnt wird dabei, dass Dateien, die als Antwort auf eine Frage hochgeladen wurden, nicht Teil des Archivs sind. Weitere optionale Dateien sind eine Token-Datei, welche Daten über die

Weitere Exportmöglichkeiten

LSS Es ist möglich, nur die im LSA enthaltene LSS-Datei zu exportieren, das wird mittels dieser Option gemacht.

Excel/.csv Hier sind weitere Einstellungen möglich, wie das Exportieren eines Teils der Antworten oder die Wahl eines bestimmten Formates (Word, Excel, CSV, HTML, PDF).

SPSS SPSS ist ein Software-Paket, welches zur statistischen Analyse von Daten genutzt wird. Auch hier kann ausgewählt werden, welche Antworten exportiert werden sollen. Die Nutzung der Open-Source Version PSPP ist auch möglich

R R ist eine Alternative zu SPSS, hier werden allerdings alle Daten exportiert

STATA-xml Auch STATA ist eine komerzielle Lösung für Datenanalyse wie SPSS. Hierfür werden die Daten von LimeSurvey direkt in das proprietäre STATA-Format umgewandelt.

VV Durch „vertical verfication“ ist es möglich, die Antworten zu modifizieren und die modifizierte Datei dann wieder zu importieren

2.3 ODM

ODM ist einer der Standards, welcher von CDISC entwickelt wurde, um den gesamten Zyklus einer Studie in ihren verschiedensten Formen zu standardisieren. Es dient dazu, sowohl Metadaten als auch klinische Daten einer Studie zu erfassen. Auch administrative Daten sind im Standard enthalten. ODM ist unabhängig von spezifischen Firmen und Plattformen und daher gut zum Austausch zwischen verschiedensten Werkzeugen und Gruppierungen geeignet. Bereits 2006 hat ODM im internationalen Raum Anklang gefunden, in Deutschland allerdings noch nicht. Das ändert sich mittlerweile, das IMI der WWU zum Beispiel nutzt das Format bereits in mehreren Werkzeugen.

2.3.1 Aufbau

Zuerst sollte angemerkt werden, dass hier bei weitem nicht alle Elemente, die im ODM-Standard definiert sind, vorgestellt werden. Das hat sowohl Zeit- als auch Platzgründe, weiterhin werden viele Elemente für die Abbildung von LimeSurvey nicht gebraucht. Für diese Arbeit sind besonders zwei Teile des ODM Standards relevant: Die „*Study*“ und die „*ClinicalData*“. Weiterhin gibt es noch „*AdminData*“, „*ReferenceData*“ und „*Association*“. Grundlegend werden andere Elemente mittels einer OID referenziert, ein Attribut, welches jedes Element besitzt, das man referenzieren kann.

Study

Eine Studie hat globale Variablen und grundlegende Definitionen. Weiterhin gibt es die „*MetaDataVersion*“, wo die Struktur der Umfrage festgelegt wird. In der einer Version der Metadaten sind alle Elemente in Referenzen und Definitionen aufgeteilt, wobei die Referenzen immer zuerst im nächst

„höhergelegenen“ Element vorkommen.

Das Basiselement einer Studie ist das „*Protocol*“. Dies enthält Elemente des Typs „*StudyEventRef*“. Dann folgen Elemente des Typs „*StudyEventDef*“, welche wiederum Referenzen auf Elemente des Typs „*FormDef*“ beinhalten. Dieses Schema setzt sich noch mit den Elementen „*ItemGroupDef*“ und „*ItemDef*“ fort.

Als weitere Elemente gibt es noch die „*CodeList*“ und die „*Condition*“. In einer „*CodeList*“ werden Antwortmöglichkeiten festgehalten, welche auf eine Frage (Element des Typs „*ItemDef*“) gegeben werden können. Dabei gibt es zwei Arten von CodeLists, die simple und die komplexe. Die simple Liste besteht aus „*EnumeratedItems*“, wo die Antwortmöglichkeit in dem Attribut „*CodedValue*“ angegeben ist. In den klinischen Daten wird als Antwort dann dieser codierte Wert stehen. Die komplexe Liste hat dieses Attribut ebenfalls, in Elementen des Typs „*CodeListItem*“. Es ist auch als Antwort angegeben, aber die tatsächliche Antwortmöglichkeit ist der Text von „*CodeListItem/Decode/TranslatedText*“.

„*ItemDef*“ enthält ein Element „*Question*“, in welchem eine Frage festgehalten wird. Mit „*ConditionDef*“ kann eine Bedingung festgelegt werden. Referenziert eine Frage ein Element dieses Typs und evaluiert der Ausdruck zu „true“, wird die Frage nicht angezeigt.

ClinicalData

In der „*ClinicalData*“ gibt es für jeden Teilnehmer der Studie ein Element des Typs „*SubjectData*“. Ab hier ist dann die Stuktur der Studie abgebildet, wobei die „*ItemData*“ das unterste Element ist und die Antwort auf eine Frage beinhaltet.

2.4 IMI Syntaxerweiterung für ODM

2.5 dom4j

Dom4j ist eine API, welche die bestehenden Funktionen der *javax*-Bibliothek abstrahiert und so wesentlich simplere Wege bietet, diese zu nutzen. Unter anderem ist es möglich, Elemente in einem XML-Dokument mittels XPath-Ausdrücken zu finden und DOM/SAX-Parser zu nutzen, um ein Dokument zu verarbeiten. Auch ein *XMLWriter* existiert, welcher unter anderem Standardelemente von XML automatisch an ein Dokument anhängt und das Format des Dokuments mittels eines einzeiligen Befehls so anpassen kann, dass es leicht für einen Menschen lesbar ist.

3 Ergebnisse

3.1 Analyse des LSS- und LSR-Formates

In der Online-Dokumetation von LimeSurvey gibt es zwar eine Sektion zum Thema Export und Export-Formate, dieses Kapitel war aber bis vor kurzem leer. Auch jetzt enthält das Kapitel nur eine oberflächliche Beschreibung der Formate, die genaue Struktur wird nicht erklärt. Da diese allerdings benötigt wird, um eine Konvertierung vornehmen zu können, muss zunächst ermittelt werden, wie die .lss- und .lsr-Dateien genau aufgebaut sind.

Dies wurde bewerkstelligt, indem zunächst ein simples Dokument erstellt wurde, welches eine Fragegruppe und drei Freitextfragen enthält. Dafür wurde eine Instanz der LimeSurvey Community Edition benötigt, diese aufzusetzen war dank existierenden Docker-Compose-Dateien relativ simpel. Nachdem die Struktur der Umfrage selbst, sowie der Fragegruppen so ermittelt wurde, wurden komplexere Dateien erstellt. Diese enthielten zunächst Fragen mit festen Antwortmöglichkeiten, also vor allem Maskenfragen und Multiple Choice Fragen. Nachdem so auch die Struktur der Antworten deutlich geworden war, wurden zuletzt die Matrixfragen eingebaut und die Struktur der Subfragen ermittelt. Das Ergebis dieser Analyse wird im Folgenden dargestellt (Es werden nicht alle existierenden Elemente angesprochen, sondern nur die für diese Arbeit relevanten, eine vollständige Auflistung kann im gefunden werden):

3.1.1 Grundlegende Struktur

Das Root-Element in LSS ist „*document*“. Hier sind zuerst grundlegende Informationen wie die Datenbank-Version und den Typ des Dokuments enthalten. Jedes der größeren Hauptelemente in „*document*“ hat die gleiche Struktur. Es gibt zwei Elemente, „*fields*“ und „*rows*“. In „*rows*“ gibt es „*row*“ Elemente, welche die Informationen selbst in weiteren Elementen enthalten. In „*fields*“ gibt es „*fieldname*“ Elemente, wobei es für jedes mögliche Element in „*rows/row*“ ein „*filedname*“ Element mit dem Namen als Text gibt.

3.1.2 Fragegruppen

Im Element „*groups*“ werden Metadaten über Fragegruppen gesammelt, wie zum Beispiel die Gruppen-ID. Diese sind aber auch in „*group_l10ns*“ enthalten, darüber hinaus sind auch noch weitere Inhalte wie Gruppenname und Sprache

in diesem Element. Daher werden später keine Informationen aus „*groups*“ später verwendet. Ein „*row*“ Element steht hier für eine Fragegruppe.

3.1.3 Fragen

Das Element „*questions*“ enthält Metadaten über Fragen, wie „*qid*“, „*gid*“, „*title*“, „*type*“. Ein „*row*“ Element steht hier für eine Frage, pro richtiger Frage in der Umfrage gibt es ein Element in „*questions*“. „*subquestions*“ hat Subfragen von Arrays, Matrizen et cetera. Diese sind via „*parent_qid*“ and ein Element aus „*questions*“ gekoppelt. Auch in „*subquestions*“ sind nur Metadaten enthalten, jede Subfrage hat, wie die richtigen Fragen auch, eine Fragen-ID. In „*question_l10ns*“ gibt es nun die tatsächliche Frage in „*question*“, weiterhin gibt es mit „*help*“ einen Hilfstext für die Frage, „*language*“ gibt die Sprache des Fragetextes an. Für jedes Element aus „*question*“ und „*subquestion*“ gibt es hier ein Element pro Sprache. Referenziert werden diese mit der „*qid*“.

„*question_attributes*“ enthält Informationen über eine Frage wie ein Präf/Suffix zu der Antwort, RegEx-Validations-Ausdrücke für die Antwort, Timings und Informationen zur Darstellung einer Frage (Textfeldbreite, Default-Antworten).

3.1.4 Antwortmöglichkeiten

Es gibt eine Reihe an Fragen, für die es eine Menge an vordefinierten Antworten gibt. Diese sind entweder schon durch die Frage festgelegt, wie bei dem Fragetyp „*5 Punkte Wahl*“ oder dem Typ „*Geschlecht*“, oder der Umfrage-Ersteller kann sie selber angeben. Sind die Möglichkeiten schon durch den Typ festgelegt, werden die Antwortmöglichkeiten implizit ermittelt und nie konkret im Dokument niedergeschrieben. Hat der Umfrage-Ersteller die Möglichkeiten selber festgelegt, werden diese in „*answers*“ und „*answer_l10ns*“ gespeichert. In „*answers*“ gibt es dabei wieder Metadaten wie „*qid*“, „*aid*“, und einen „*code*“. In „*answer_l10ns*“ hingegen gibt es den Antworttext in „*answer*“, eine „*aid*“ zur Verknüpfung mit den Metadaten und die Sprache in „*language*“. Für jedes Element aus „*answers*“ gibt es pro Sprache ein Element in „*answer_l10ns*“.

3.1.5 Umfrage-Metadaten

In „*surveys*“ findet man Metadaten über die Umfrage, allerdings sind keine davon für die Konvertierung relevant. Beispiele wären Daten darüber, ob Willkommenstexte angezeigt werden sollen oder ob IP-Adressen der Teilnehmer gespeichert werden sollen. „*surveyls_languagesettings*“ enthält relevante Informationen wie die „*surveyls_id*“, den Titel in „*surveyls_title*“ und eine Beschreibung der Umfrage in „*surveyls_description*“. „*themes*“ und „*themes_inherited*“ enthalten Informationen über die visuelle Darstellung der Umfrage in LimeSurvey.

3.1.6 LSR-Aufbau

Auch für die Antworten wurde die gleiche Strategie wie für die Umfragestruktur verwendet. Die Ergebnisse sind wie folgt:

Das Hauptelement ist wieder „*Document*“, zuerst gibt es wieder einige Metadaten wie der Typ des Dokuments, die Datenbank-Version und die Sprache. Dann gibt es ein „*responses*“ Element, welches dieselbe grundlegende Struktur wie die Elemente in der LSS-Datei besitzt. Für jeden Teilnehmer der Umfrage gibt es ein Element vom Typ „*row*“. Dies enthält eine ID, das Absendedatum, die Sprache und einen Seed. Dann kommen die Antworten, wobei es für jede Frage bzw. Subfrage ein Element mit folgendem Namensschema gibt: „$\_\{Survey\text{-}ID\}X\{GID\}X\{QID\}\{SQID\}?\{ext\}?$“, wobei „*ext*“ folgendes sein kann („other“|„comment“). Man kann bereits am Format sehen, dass es auch für die „other“ und Kommentar-Felder hier eine separate Antwort gibt.

3.1.7 Erstellung eines XML-Schemas

Da die Struktur der Umfrage nun bekannt ist, wird ein XML-Schema für das LSS-Format erstellt. Das sorgt einerseits für eine wesentlich bessere Übersicht über den Aufbau, das kann für zukünftige Arbeit nützlich sein. Andererseits lässt sich dieses Schema später verwenden, um zu überprüfen, ob es sich bei der Eingabedatei um eine valide LSS-Datei handelt.

3.2 Mapping

Da der Aufbau beider Formate nun bekannt ist, muss überlegt werden, wie man das LSS-Format auf ODM abbilden kann. Im Folgenden werden die verschiedenen Teile von LSS nacheinander abgehandelt und es wird erklärt, wie mit überschüssigen Elementen in ODM ungegangen wird.

3.2.1 Dummy Elemente in ODM

ODM besitzt mehr Möglichkeiten, mögliche Abläufe einer Studie oder Umfrage darzustellen, da es als generisches Format natülich mehr Einsatzzwecke abdecken soll, als das LSS-Format. Dementsprechend gibt es mehrere Wege, LSS auf ODM abzubilden. Durch die hier getroffenen Entscheidungen werden die Elemente „*StudyEvent*“ und „*GlobalVariables*“ überflüssig, da es sich dabei um für eine Studie relevante Felder handelt, es soll aber nur eine Umfrage innerhalb einer Studie erstellt werden.

Da diese Elemente dennoch zwingend in ODM vorkommen müssen, werden die globalen Variablen leer gelassen und es wird ein „*StudyEvent*“ erstellt, welches unabhängig von der Eingabe-Datei immer die gleichen Dummy-Werte hat. Weiterhin wird die „*OID*“ des „*Study*“ Elements auch auf einen Dummy-Wert gesetzt, da wir hier ebenfalls kein Äquivalent in LSS haben. Die „*OID*“ der

„*MetaDataVersion*“ wird auf „MetaData“ + Study-ID gesetzt, da die Version der Metadaten die gesamte Umfrage enthält, es ergibt also Sinn, hier keinen Dummy-Wert einzusetzen sondern eine Referenz auf die Studie mit einzubringen.

3.2.2 Umfrage-Eigenschaften

Innerhalb unseres Dummy-„*StudyEvent's*“ gibt es nun ein „*Form*“ Element, dieses soll die Umfrage repräsentieren. Das „*Repeating*“ Attribut wird auf „No“ gesetzt, da die Umfrage nur einmal pro Teilnehmer ausgefüllt werden soll, füllt jemand sie mehrmals aus, gerade wenn es keine vorher festgelegten Nutzer gibt, gilt das als zwei verschiedene Teilnehmer Als „*OID*“ des „*Form's*“ wird die Study-ID gewählt, als „*Name*“ wird der Titel der Umfrage gesetzt. Die „*surveyls_description*“ wird zur „*Description*“ des „*Form's*“.

3.2.3 Fragegruppen

Fragegruppen in LimeSurvey sind fast äquivalent zu „*ItemGroup's*“ in ODM. Entsprechend wird für jede Fragegruppe eine „*ItemGroupDef*“ erstellt, die „*gid*“ wird zur „*OID*“ in ODM, der „*title*“ wird zum „*Name*“. Das Attribut „*Repeating*“ wird wieder auf „No“ gesetzt. Entsprechende Referenzen werden zur „*FormDef*“ hinzugefügt.

3.2.4 Fragen

Grob gesprochen wird die Frage aus „*question*“ in LimeSurveys „*question_l10ns*“ Element bei ODM in „*Question/TranslatedText*“ der „*ItemDef*“ eingetragen, die Sprache aus „*language*“ wird im Attribut „*xml:lang*“ des „*TranslatedText*“ eingetragen. Aus „*qid*“ wird „*OID*“ und aus „*help*“ wird elDescription. Aus „*questions*“ wird „*title*“ genutzt, um es in „*Name*“ einzusetzen und „*mandatory*“ wird auf „*Mandatory*“ der „*ItemRef*“ abgebildet.

In ODM gibt es allerdings keine Subfragen, dementsprechend muss eine Frage in LimeSurvey potentiell in mehrere Fragen, eine pro Subfrage, umgewandelt werden, damit diese dann in ODM eingefügt werden können. Dementsprechend muss jeder Fragetyp aus LimeSurvey einzeln behandelt werden, wobei die Behandlung eines Fragetyps teils in der Behandlung eines anderen Fragetyps enthalten ist. Auch sind die Behandlungen für mehrere Fragetypen identisch. Zum Beispiel werden die Fragetexte aus „*question*“ der Hauptfrage und „*question*“ der Subfrage konkateniert, um so eine neue Frage zu formulieren. Für die Festlegung der „*OID*“ von Subfragen wird die „*qid*“ der Hauptfrage mit dem „*title*“ der Subfrage konkateniert. Gründe dafür werden in Abschnitt 3.3.6 dargelegt. Im folgenden werden alle Behandlungen erklärt, mit einer Angabe, für welche Fragetypen diese genutzt werden.

Einfachauswahl

Hier werden die Fragen 1:1 abgebildet, die Antwortmöglichkeiten werden in einer CodeList gespeichert. Für den Typ „*Liste mit Kommentar*“ wird eine weitere Frage hinzugefügt, deren „*OID*“ die „*qid*“ der Frage ist, allerdings wird noch „comment“ angehangen. An die Frage wird ebenfalls das gleiche angehangen. Bei dem Bild der „*Image-Select-List*“ besteht das Problem, dass dieses nur mittels Link zum Bild auf dem LimeSurvey-Server eingebettet ist. Beim Übertragen der Fragen ist somit auch nur dieser Link enthalten.

Matrix

Für die Matrizen mit Zahlen- oder Freitextantworten wird eine Frage pro Zelle der Matrix hinzugefügt. Für alle weiteren Typen dieser Kategorie bis auf die „*Dual Matrix*“ wird eine Frage pro Subfrage hinzugefügt, die gleiche Liste an Antwortmöglichkeiten wird für jede Subfrage verwendet. Die „*Dual Matrix*“ wird behandelt, als würde die gleiche Frage zwei Mal mit unterschiedlichen Antwortmöglichkeiten dargestellt.

Multiple Choice

Hier wird eine Frage pro Subfrage hinzugefügt, Das Bild beim Typ „*Image Select*“ wird wie schon bei Abschnitt 3.2.4 nur auf Umwegen übernommen.

Textfragen

Hier wird der Fragetext wie oben erklärt übertragen, als „*DataType*“-Attribut in „*ItemDef*“ wird „string“ gesetzt. Das funktioniert für alle drei Freitextfragetypen, kurz, lang und riesig so. Im Falle von „*Mehrere Texte*“ wird wieder eine Frage pro Subfrage genutzt. „*Input on Demand*“ und „*Browser Detect*“ werden nicht gemappt. Für Details siehe Abschnitt 4.1.

Maskenfragen

Für den Typ „*Datum/Zeit*“ wird der „*DataType*“ auf „datetime“ gesetzt. Für „*Zahleneingabe*“ wird der „*DataType*“ auf „float“ gesetzt. Für „*Mehrfache Zahlen*“ wird wieder eine Frage pro Subfrage erstellt Es gibt mehrere Typen, bei denen es feste Antwortmöglichkeiten gibt und daher eine CodeList implizit aus dem Typ erstellt wird: „*Ja/Nein*“, „*Geschlecht*“. „*Ranking*“, „*Textanzeige*“, „*Dateiupload*“ und „*Sprachumschaltung*“ werden nicht gemappt. Für weitere Details siehe Abschnitt 4.1.

3.2.5 Antwortmöglichkeiten

Grundsätzlich wird jedes Mal, wenn es eine vordefinierte Liste an Antwortmöglichkeiten gibt, eine CodeList genutzt, um diese Liste in ODM darzustellen. Für die Fra-

gen, wo man mit 1 bis 5 oder 1 bis 10 antworten kann, wird eine simple Liste genutzt, für alle weiteren Fragen eine komplexe Liste. Das liegt unter anderem daran, dass LimeSurvey fast alle Antworten mit einem Buchstaben darstellt und die tatsächlichen Antworten aus mindestens einem Wort bestehen. So werden komplexere Umwandlungen vermieden und die existierende Struktur wird übernommen. Für selbstdefinierte Antwortmöglichkeiten ist ohnehin eine komplexe Liste notwendig, da der Ersteller der Umfrage beliebige Kombinationen für Code und Antwort definieren kann.

3.2.6 Antworten

Als „*StudyOID*“ und „*MetaDataVersionOID*“ der „*ClinicalData*“ werden die entsprechenden OIDs genutzt. Jede „*row*“ aus dem LSR Dokument wird auf ein „*SubjectData*“ Element abgebildet. Als „*SubjectKey*“ wird die „*id*“ genutzt. Für „*StudyEventOID*“ und „*FormOID*“ werden die bereits erstellten Werte eingetragen. Als „*ItemGroupOID*“ wird die „*gid*“ gesetzt. Als „*ItemOID*“ des „*ItemData*“ Elements wird ein Teil des Elementnamens genutzt, nämlich „{qid}{ext}“, in „*Value*“ wird der Text des Elements eingetragen. Da die OIDs der Fragen vorher bereits so gewählt wurden, dass sie mit dieser Stuktur übereinstimmen, haben wir bereits funktionierende Referenzen, ohne weitere Umwandlungen vornehmen zu müssen.

3.2.7 Themes und Frageattribute

Die Themes werden nicht in ODM übernommen. Weitere Informationen gibt es in Abschnitt 4.2. Auch von den Frageattributen werden fast keine übernommen, da die meisten der visuellen Darstellung dienen.

3.3 Implementierung

3.3.1 Java

- Bietet sehr mächtige Werkzeuge zur Bearbeitung von XML - Kann z.B. mit mehreren Attributen in einem Element umgehen - Unterstützt XSD-1.1 - Wird beim IMI bereits viel genutzt

3.3.2 Eingabe

- Archiv - muss entpackt werden

3.3.3 Properties

3.3.4 XSD-Validierung

- Nutze Xerces zusammen mit der erstellten XSD um eine Eingabedatei zu prüfen - Prüfung ist nicht bindend - Auch invalide Datei wird weiter verarbeitet - Grund: Starke Versionsabhängigkeit - Invalide lss-Datei kann eventuell trotzdem erfolgreich umgewandelt werden - Zur Information für Anwender: Potentiell problematisch/inkompatibel mit Konverter

3.3.5 Parsing der LimeSurvey Struktur

Zuerst wird in der Main-Methode eine Instanz der Klasse *LssParser* mit der entsprechenden LSS-Datei erstellt, dann wird die Methode *parseDocument* aufgerufen. In *LssParser* gibt es eine Instanz der Datenklasse *Survey*. In dieser werden zunächst alle gesammelten Informationen abgelegt.

Hier werden zunächst die drei in Abschnitt 3.1.5 angesrochenen Elemente mit Metadaten der Studie in die *survey* übernommen. Dann werden die Fragegruppen übernommen, hierbei werden alle für das Mapping relevante Informationen gespeichert.

Parsing der Fragen

Als nächstes werden die Fragen in *survey* übernommen. Dabei werden diese bereits gemäß des Mappings umgewandelt. Das wird gemacht, um die Gesamtstruktur so früh wie möglich zu simplifizieren und nicht für jede Frage eine gesonderte Klasse erstellen zu müssen. Da sich viele Fragen im Aufbau stark ähneln, ist dies ein simpler und schneller Weg, dieses Ziel zu erreichen. Am Ende dieser Verarbeitung soll es noch vier Fragetypen geben:

T Eine Frage, auf die mit einem Freitext geantwortet werden kann

N Bei dieser Frage muss mit einer Zahl geantwortet werden

A Bei dieser Frage muss aus einer vordifinierten Liste an Antwortmöglichkeiten gewählt werden

D Diese Frage hat ein Datum und eine Uhrzeit als Antwort

Zuerst wird eine Liste aller „*row*“ Elemente aus „*questions*“ erstellt, durch welche im Anschluss iteriert wird. In jeder Iteration wird nun zuerst eine Instanz der Klasse *Question* erstellt, wo alle relevanten Daten über die Frage gespeichert werden. Eine Liste dieser ist in zu sehen. Nicht in jedem Fall wird diese Frage auch zur Umfrage hinzugefügt, wenn es sich zum Beispiel um eine Matrix-Frage handelt, wird diese Instanz nur zum Erstellen neuer Fragen genutzt. Dann werden potentiell existierende Bedingungen hinzugefügt.

Anschließend gibt es ein *switch*-Statement, welches einen passenden Weg zum Parsen der Frage abhängig vom Typ ausführt.

Für Textfragen wird hier nur der Typ zu „T“ geändert. Bei Datum/Zeit-Fragen wird die „*qid*“ zu einer Liste hinzugefügt, welche später in Abschnitt 3.3.6 gebraucht wird. Eine numerische Frage braucht keine weitere Bearbeitung.

Dabei wird die Frage-ID wie folgt aufgebaut: „{Parent_QID} + {Title(y-Axis)}_{Title(x-Axis)}“.

Parsing der Anzeige-Bedingungen

Da die Bedingungen innerhalb des LSS-Formates ohnehin als Elemente gespeichert werden und nicht als ExpressionScript, ergibt es mehr Sinn, diese nicht wieder in ExpressionScript umzuwandeln, sondern direkt in die Syntax des IMI. Die in LimeSurvey mit ExpressionScript formulierten Bedingungen geben an, wann eine Frage angezeigt werden soll. Die in ODM definierten Bedingungen müssen zu „True“ evaluieren, wenn eine Frage nicht angezeigt werden soll. Somit muss die Bedingung zuerst umgedreht werden, indem ein Paar an Klammern um den Gesamtausdruck gesetzt wird und ein *NOT* vor den Anfang geschrieben wird. Dann werden alle Bedingungen rausgesucht, welche zur „*qid*“ gehören. Diese sollen nun durch ein logisches „Und“ verknüpft werden.

Die Frage aus „*cfieldname*“ wird mittels des regulären Ausdrucks „^\\d+X(\\d+)X(.+?)$“ so verarbeitet, dass wir die darin enthaltene „*gid*“ und „*qid*“ der Frage enthalten. Mit diesen, der Dummy-ID für das *StudyEvent* und der Umfragen-ID wird dann ein Pfad gemäß der IMI-Syntax erstellt.

Handelt es sich nicht um einen Regulären Ausdruck, der als Operator genutzt wird, werden folgende drei Teile in dieser Reihenfolge aneinander gehangen: „{*PATH*} {*OPERATOR*} {*VALUE*}“. Wobei das Element „*method*“ den Operator enthält und „*value*“ den Wert. *PATH* ist der vorher erstellte Pfad.

Handelt es sich um einen regulären Ausdruck, wird das führende Leerzeichen entfernt und die Bedingung in folgender Form aufgeschrieben: „*MATCH({REGEX}, {PATH})*“.

Soll eine Antwort leer bleiben, wird „NULL“ verwendet. Bei beidem handelt es sich nicht um einen Teil der IMI-Syntax, weiteres dazu in Abschnitt 4.7.

Abschließend wird der Bedingung eine OID gegeben, in der Form „{*qid*}{*ex*}“, wobei „*ex*“ in der Properties-Datei festgelegt werden kann. Diese wird auch in der Frage eingetragen.

3.3.6 Parsing der LimeSurvey Antworten

Die Klasse *LsrParser*, zusammen mit den Klassen *Response* und *Answer* wurden erstellt, um diesen Zweck zu erfüllen. Zuerst wird in *createDocument* ein neuer SAXReader erstellt, welcher die LSR-Datei in eine Instanz der Klasse Document einliest. Selbige wird dann in *parseAnswers* weiterverwendet.

Alle für das Parsing relevanten Felder stehen in „*document/responses/rows*“. Dort gibt es ein „*Row*“ Element pro Beantwortung des Fragebogens. Entsprechend wird über eine Liste aller „*row*“ Elemente iteriert. Dabei wird für jede „*row*“ eine Instanz der Klasse *Response* erstellt. Als *id* wird der Text des Elements „*id*“ genommen. Dann wird bereits etwas Vorbereitung für die spätere Erstellung der ODM-Datei betrieben, es gibt in *Response* nämlich eine Map namens *answers*, welche vom Typ $< Integer, ArrayList < Answer >>$ ist. Dort wird ein Key/Value Paar für jede Fragegruppe erstellt, wobei der Integer die *gid* ist. Als nächstes wird über alle Kind-Elemente von „*row*“ iteriert und es wird nach Antworten auf Fragen gesucht. Diese sind an der Struktur des Element-Namens zu erkennen, sie besitzen fast den gleichen Aufbau wie der Inhalt des Elements „*cfieldname*“ aus Abschnitt 3.3.5, allerdings gibt es hier zusätzlich noch einen führenden Unterstrich. Wir nutzen also den regulären Ausdruck „^_\\d+X(\\d+)X(.+?)$“, um pasende Elemente zu finden und mittels der Capture Groups die *gid* und *qid* zu erhalten.

Dann wird die Antwort aus dem Elementtext gezogen. Dann wird geprüft, ob die Antwort leer ist, in dem Fall wird sie ignoriert. Ist sie nicht leer, wird geschaut, ob es sich um eine „*Datum/Zeit*“ Frage handelt, dann wird das Leerzeichen durch ein „T“ ersetzt, damit das Format dem Datentyp „xsd:dateTime“ entspricht. Dann wird eine neue Instanz der Klasse *Answer* erstellt, dabei im Konstruktor *gid* und *qid* aus dem regulären Ausdruck und die Antwort aus dem Elementtext gesetzt und schließlich wird die Antwort mittels der Methode *addToAnswers* zu *answers* hinzugefügt. Diese Methode sortiert die Antwort dabei in die passende ArrayList ein, sodass am Ende alle Antworten zu Fragen aus der gleichen Fragegruppe in einer Liste sind.

Dann wird die *Response* zur Liste *responses* hinzugefügt. Zuletzt werden die Antworten in das ODM Dokument übertragen, falls es bereits 1000 in Responses gibt und dann wird die Liste geleert. Dadurch soll die Menge an gebrauchtem RAM verringert werden.

3.3.7 Ausgabe als ODM-Datei

Mittels der Klasse *ODMWriter* soll eine neue ODM-Datei erstellt werden. Der Konstruktor nimmt dabei eine *Survey* entgegen. Die Methode *createODMFile* dient vor allem als Caller für andere Methoden, welche die eigentlichen Funktionen ausführen.

Zuerst erstellt *createODMRoot* das Wurzelelement „*ODM*“ mit allen benötigten Attributen. Als nächstes wird *addStudyData* aufgerufen. Hier werden die Elemente „*Study*“, sowie die globalen Variablen mittels der Dummy-Werte aus der Properties-Datei erstellt. Weiterhin werden die Elemente „*MetaDataVersion*“„*Protocol*“, „*StudyEventDef*“ und „*Form*“ erstellt. Die *meta_data_oid* besteht dabei aus dem Prefix in der Properties-Datei und der Studien-ID. „*OID*“ und „*Name*“ des „*StudyEvent's*“ stammen aus der Properties-Datei, das Element „*Form*“ wird gemäß des Mappings befüllt.

Nun werden die Fragegruppen mittels *addQuestionGroups* hinzugefügt. Wie in Abschnitt 3.2.3 beschrieben wird aus jeder *QuestionGroup* ein Element „*ItemGroupDef*“. In der „*Form*“ wird die entsprechende Referenz auf die Fragegruppe eingefügt. Gibt es eine Beschreibung, wird sie in ein neues Element „*Description*“ in der „*ItemGroupDef*“ eingefügt. In die Map *question_groups* der Klasse *LsrParser* wird dann ein Eintrag mit der *gid* und der Referenz auf das neu erstellte XML-Element eingefügt, sodass hier später einfach Referenzen auf Fragen eingefügt werden können.

Danach werden alle Fragen eingefügt.

4 Diskussion

4.1 Weglassen von Fragetypen

Auch wenn mit diesem Konverter eine vollständige Umsetzung der LimeSurvey Archiv-Daten angestrebt wird, so wurden doch einige Fragetypen bewusst nicht umgesetzt. Im Folgenden soll erläutert werden, welche Fragetypen nicht konvertiert wurden, wie diese Fragetypen hätten umgesetzt werden können und warum die Entscheidung getroffen wurde, dies nicht zu tun.

4.1.1 Datei-Upload

Der Fragetyp „*Datei-Upload*“ kann genutzt werden, um den Nutzer auf eine Frage mit einer Datei antworten zu lassen. In ODM hätte man diese Datei einbinden können, indem man sie in „*hexBinary*“ umwandelt, ein Datentyp in ODM, welcher Stream-Daten in einem hexacodierten Binärformat sammelt. Trotzdem wurde dieser Fragentyp im Konverter nicht umgesetzt. Das liegt zum einen daran, dass die hochgeladenen Dateien nicht Teil des Archives sind (siehe Abschnitt 2.2.4) und andererseits daran, dass die Praktikabilität dieses Vorgehens eher fragwürdig ist. Unter anderem wird eine Rekonstruktion zur Originaldatei schwer, da es zum Beispiel keine Informationen über den ursprünglichen Dateityp gibt, auch wird die XMl-Datei nur noch sehr unangenehm von Hand lesbar, wenn man diese Stream-Daten in Antworten einbinden würde. Auch wird der Fragetyp nicht häufig genutzt, was die Umsetzung noch unattraktiver macht.

4.1.2 Browser-Detection, Language-Switch

- Es werden Infos gesammelt - Darstellung durch Frage + Antwort
  - Ruft die Irreführende Darstellung hervor, der Nutzer habe geantwortet (Was nicht der Fall ist) - Die Information ist nicht Teil der Meinung/Ansicht des Teilnehmers, es sind eher Metainformationen über den Antwortenden

4.1.3 Text-Display

- Es werden keine Informationen gesammelt - Es ist nicht automatisiert feststellbar, wozu Infos des Textes gehören - Nicht z.B. als Description irgendwo zuordnebar

4.2 Visuelle Darstellung der Fragen

- LimeSurvey speichert viele Infos über die Darstellung - Themes - CSS-Klassen - Attribute für visuelle Elemente - Werden nicht übernommen in ODM - Ist kritisch, da die Darstellung von Fragen direkten Einfluss auf die Antworten hat - So gehen evtl. Infos verloren, die ein gewisses Antwortverhalten erklären könnten - Infos sind Teil des .lsa-Archives

- ODM bietet keinen Weg, visuelle Darstellungen zu speichern - Jede Lösung ist nicht im Standard definiert und daher potentiell für andere nicht verwendbar - Es handelt sich weder um Fragen noch um Antworten, auch wenn die Informationen nicht egal sind, so sind sie doch auch nicht sehr relevant - Oft ist am wichtigsten, wie die Frage formuliert ist, die Formulierung wird hier übernommen

4.2.1 Timings

4.3 Formatierung

- Nicht alle Fragen können in ihrer Ursprungsform dargestellt werden - ODM bietet wenig Möglichkeiten, anzugeben, wie Fragen dargestellt werden sollen - LimeSurvey hat wesentlich mehr Optionen

4.3.1 Arrays

- Auseinanderziehen in Einzelne Single-Choice-Fragen ist nicht optimal - Originale Struktur geht verloren - Bestimmtes Antwortverhalten tritt nur bei Arrays auf - Antworten in Mustern

- Darstellung der Antwortmöglichkeiten bleibt - Änderungen sind nicht sehr groß - Antwortverhalten wie oben ist nicht sehr häufig und bei Analyse größerer Datenmengen wohl ohnehin nicht feststellbar

4.3.2 Multiple Choice Fragen

- Auseinanderziehen in einzelne Ja/-Nein Fragen ist nicht optimal - Man macht sich mehr Gedanken über einzelne Antwortmöglichkeiten - Man überliest potentiell weniger Antworten - Man ist weniger geneigt, mehrere Dinge zu antworten - Die IMI bietet eine eigene Syntax für die Darstellung von MC-Fragen

- ODM hat keine Optionen für MC-Fragen - Ist eine bessere Alternative dazu, die gleiche Frage fünf Mal zu stellen

4.4 Versionsabhängigkeit

4.5 XSD Defintion

4.5.1 Element-Inhalte

Auffällig war, dass in der LSS-Datei nur CDATA-Werte als Text verwendet werden. So ist niemals klar, welcher Datentyp genau nun in ein bestimmtes Feld gehört. Das Schema erzwingt hier teils genauere Datentypen, wenn der Inhalt offensichtlich ist, allerdings ist dies nicht immer möglich. Trotzdem wird so nicht nur ein Maß an struktureller Korrektheit sondern auch an inhaltlicher Korrektheit erzwungen. Prinzipiell sollten diese Datentypen einer aus LimeSurvey exportierten Datei nie im Wege stehen, daher ist die Einführung dieser kein Problem.

4.5.2 Design

Auch das Design der hardgecodeten Elemente ist nicht optimal, da so in der Zukunft zum LSS-Format hinzugefügte Elemente als invalide erkannt werden. Man könnte sicherlich einen dynamischeren Weg kreieren, indem man die Liste an möglichen Elementen in dem „*fields*“-Element und Features von XSD 1.1 nutzt. Allerdings widerspricht das der Art und Weise, wie XSD verwendet werden sollte, das festhalten der genauen Elementnamen ist dort vorgesehen. Auch ist es kein großes Problem, da der Konverter diese Felder dann sowieso nicht kennt, so wird man ebenfalls vor potentiellen Inkompatibilitäten gewarnt.

4.6 Implementierung

4.6.1 Java

- Nicht die schnellste Sprache - Bietet viele praktische Funktionen anderer Sprachen nicht - Optionale Parameter

- Betriebssystemunabhängig - Einfache Mittel zur Programmierung - Logging (log4j) - Ordentlichen Code (Lombok) - Keine Getter/Setter, kaum Konstruktoren im Code - Immer noch schneller als andere Sprachen (Python)

4.6.2 Switch-Statement

- Eher C-Stil, für Objektorientierung nicht optimal

- Bietet sich an - mehrere Fragetypen haben gleiche Verarbeitung - Verarbeitung mancher Fragetypen ist Teil einer anderen Verarbeitung - =¿ Switch Statement kann all diese Fälle gut verarbeiten

4.6.3 JAXB

4.7 Erweiterung der IMI-Syntax

In der IMI-Syntax wird keine Möglichkeit vorgestellt, reguläre Ausdrücke zu evaluieren. Daher wird in dieser Arbeit vorgeschlagen, eine Funktion „*MATCH(REGEX, PATH)*“ einzuführen, welche reguläre Ausdrücke mit beginnendem und endendem Schrägstrich und potentiell Flags hinter dem endenden Schrägstrich entgegennimmt und prüft, ob die Antwort dem Muster entspricht. Mittels einer Custom-Funktion ließe sich so eine Syntax in *expr-eval* und *EvalEx* integrieren.

Auch wird keine Möglichkeit geboten, die Abwesenheit einer Antwort zu überprüfen. Dafür wird ein Vergleich mit „NULL“ vorgeschlagen. Das wird allerdings nur von *EvalEx* unterstützt, nicht durch *expr-eval*.

4.8 Verwandte Arbeiten

- OpenClinica zu ODM - Kann nur Klinische Daten einlesen, keine Metadaten (Angeblich) - Betrachtet verschiedene Eigenschaften, die für eine Konformität zum ODM-Standard eingehalten werden müssen - Zeigt ein Mapping (Hier werden auch Metadaten gemappt) - Nutzt eine Vendor-Extension, ebenfalls im Mapping enthalten - Redet sowohl über Import als auch Export

5 Fazit

Dieses Kapitel bildet die abschließende Zusammenfassung der Arbeit. Dazu können die folgende Punkte behandelt werden:

- Reflexion: wurden die Ziele der Arbeit erreicht?
- mögliche Erweiterungen und Verbesserungen („future work“)

Eidesstattliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit über *„Titel“* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind.

Vorname Nachname, Münster, 7. Juni 2021

Ich erkläre mich mit einem Abgleich der Arbeit mit anderen Texten zwecks Auffindung von Übereinstimmungen sowie mit einer zu diesem Zweck vorzunehmenden Speicherung der Arbeit in eine Datenbank einverstanden.

Vorname Nachname, Münster, 7. Juni 2021