

Konzept und prototypische Implementierung eines übergreifenden Dokumenten- und Medienmanagements

BACHELORARBEIT

für die Prüfung zum

Bachelor of Science

des Studienganges Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Sebastian Rieger

Abgabedatum 24.08.2015

Bearbeitungszeitraum

12 Wochen

Matrikelnummer

7406886

Kurs

TINF12B1

Ausbildungsfirma

Karlsruher Institut für Technologie (KIT)

Karlsruhe

Betreuer der Ausbildungsfirma

Dipl. Inform. Thorsten Schlachter

Gutachter der Studienakademie

Dipl. Inform. (BA), MBA Christian Jordan

Erklärung

Gemäß§16 (3) der „Studien- und Prüfungsordnung für den Studienbereich Technik“ vom 1.11.2007.

Ich habe die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Ort Datum

Unterschrift

Inhaltsverzeichnis

1 Abstract	7
2 Einleitung	8
3 Lastenheft	9
3.1 Zielsetzung	9
3.2 Produkteinsatz	9
3.2.1 Zusammenspiel mit anderen Systemen	9
3.3 Produktfunktionen	9
3.4 Produktdaten	10
3.5 Produktleistungen	10
4 Stand der Technik / existierende Konzepte	11
4.1 FADO und Untergruppen	11
4.2 Das Document Retrieval System	11
4.3 Naturschutz-Bildarchiv	12
4.4 Literaturarchiv der ICT-ENSURE	13
5 Analyse von Metadatenstandards	14
5.1 Fachliche Metadaten	14
5.2 Fachliche Metadaten-Standards	14
5.2.1 Darwin Core	14
5.2.2 INSPIRE	15
5.2.3 DIN EN ISO 19115	16
5.2.4 ONIX	16
5.3 Technische Metadaten	16
5.4 Technische Metadaten-Standards	16
5.4.1 Dublin Core	17
5.4.2 Exif	17
5.4.3 Andere Standards	18
6 Erstellung eines Datenkonzepts	19
6.1 FADO Metadaten	19
6.2 DRS Metadaten / Bildarchiv Metadaten	20
6.3 ICT-ENSURE Metadaten	20
6.4 Untergeordnete Datensammlungen	21

6.4.1	Gerichtbarkeit	21
6.4.2	Bibliographie Metadaten	22
6.4.3	Abstrakte Metadaten	23
6.4.4	Standard Metadaten	24
6.4.5	Abstrakte Standard Metadaten	24
6.4.6	Grundlegenden Daten	26
6.4.7	Urheber Metadaten	27
6.5	Metadatenmapping im neuen Modell	27
6.5.1	Metadatenmapping FADO	28
6.5.2	Metadatenmapping DRS	31
6.5.3	Metadatenmapping Bildarchiv	31
6.5.4	Metadatenmapping ICT-ENSURE	32
7	Technologievergleich	34
7.1	agorum core	34
7.2	Alfresco	35
7.3	Open-Xchange	37
7.4	Liferay als ECM-Tool	38
7.5	Auswertung der Möglichkeiten	39
8	Implementierung des Backends auf Basis von Alfresco	41
8.1	Installation	41
8.2	Metadatenmodell in Alfresco	41
8.3	Verändertes Datenmodell für Alfresco	42
8.3.1	Die Datentypen	43
8.3.2	Die Klasse Gerichtbarkeit	44
8.3.3	Die Klasse Standard-Aspekte	45
8.3.4	Die Klasse Bibliografische Aspekte	46
8.4	Implementierung einer Datensammlung	47
8.4.1	Das Modell in Alfresco einbinden	50
8.4.2	Übersetzungen in Alfresco	50
8.5	Modell und Übersetzungen in Alfresco anzeigen	51
8.5.1	Das Datenmodell anzeigen	51
8.5.2	Aspekte anzeigen	54
8.5.3	Datentyp umwandeln	54
8.5.4	Übersetzungen einbinden	55

8.6	Metadatenmodell Editor	56
8.7	Bulk-Import von Dateien	56
8.8	Verwendung von langen Textfeldern	56
8.9	Erfahrungen aus der Alfresco-Konfiguration	57
8.9.1	Aufspaltung der share-config-custom.xml	57
8.9.2	Reihenfolge der Java-Beans beachten	58
8.10	Fehlende Funktionalität im Backend	59
9	Implementierung des Frontends auf Basis von Liferay	60
9.1	Alfresco als IFrame im Liferay	60
9.2	Liferay-Repository als Frontend für Alfresco Dateien	60
9.2.1	Benutzerverwaltung für eine Repositoryzugriff konfigurieren	60
9.2.2	Alfresco als Repository im Liferay	61
9.2.3	Untersuchungen zum Liferay-Repository	61
9.3	Alternativen zum bestehenden Liferay-Repository	62
9.3.1	Eingriff in den Liferay Quellcode	63
9.3.2	Entwicklung eines neuen Liferay-Hooks	63
9.4	Entwicklung eines neuen Liferay-Portlets	66
9.4.1	Entwicklung von Widgets	66
9.4.2	Liferay Enterprise Edition	66
9.4.3	Verwendung von Liferay 7	67
9.5	Auswertung der Möglichkeiten	67
10	Vergleich zwischen Altsystem und dem neuen Prototyp	68
10.1	Vergleich des Frontends	68
10.2	Vergleich des Backends	68
11	Zusammenfassung	70
11.1	Umsetzung des Lastenhefts	71
12	Ausblick	72
13	Abkürzungsverzeichnis	73
14	Anhang	74
14.1	Metadaten der LUBW Fachsysteme	74
14.1.1	Metadaten des Fachsystems FADO / Urteile	74
14.1.2	Metadaten des Fachsystems FADO / Forschungsvorhaben	75

14.1.3 Metadaten des Fachsystems FADO / Berichte	76
14.1.4 Metadaten des DRS	77
14.1.5 Metadaten des Bildarchivs	78
14.2 Metadaten der ICT-ENSURE	79
14.3 Exif-Metadaten von Alfresco	80
14.4 Bilder des Datenmodells	81
14.5 Darstellung der Metadaten in Alfresco	83
14.6 Darstellung einer Bulk-Import XML-Datei	84
14.7 Repositorydarstellung in Liferay	85
14.8 Die Methode createCmisSession()	85

1 Abstract

In this paper a comprehensive metadata model for document management systems of the *Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg* (LUBW) is created. The LUBW runs different Web portals, which have grown independently for historical reasons and therefore use different metadata models. These portals will be studied and analyzed how the metadata models can be unified.

The aim of this paper is to clarify if each of the respective metadata models can be represented by a combination of reusable and standardized metadata components. For the implementation of a prototype system and as a proof of concept, different *Enterprise Content Management* (ECM) systems need to be analyzed and a suitable system will be selected to build the prototype upon.

It is described how the unified metadata model can be implemented using Alfresco, one of the most popular ECM systems.

After the development of the backend, different options for building a user frontend based on a Liferay portal server, in particular, an implementation based on *Content Management Interoperability Services* (CMIS).

2 Einleitung

Behörden und Firmen haben heute immer größere Datenbestände zu verwalten, welche schon in elektronischen Systemen vorzufinden sind. Da diese Systeme historisch bedingt gewachsen sind, entstanden im Laufe der Zeit immer mehr Insellösungen, welche ein übergreifendes Verwalten und Nutzen von Daten erschwert oder gar ganz verhindert.

Das Ziel dieser Arbeit ist daher, ein Konzept sowie eine prototypische Implementierung zu erstellen, welche ein übergreifendes und vereinheitlichtes Arbeiten mit den Datenbeständen ermöglicht. Heutige *Dokumentenmanagementsysteme* (DMS) decken eine Vielzahl von Anwendungsgebieten ab, welche an die jeweiligen Problemstellungen wie Rechtsfragen oder fachliche Anforderungen angepasst sind oder angepasst werden können. Ein genaues Konzept mit Anforderungen soll am Beispiel der *Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg* (LUBW) sowie der *Gewerbeaufsicht Baden-Württemberg* (GAA) entstehen. [Göt14]

Es sollen durch das Konzept auch bestehende Systeme weiter unterstützt werden, wofür hier eine Anbindmöglichkeit geschaffen werden soll. Über eine serviceorientierte Anbindung sollen somit die verschiedensten Frontends wie Desktopanwendung, Web, Mobil usw. das neue Konzept nutzen können. Welches *Enterprise Content Management* (ECM)-Tool für die Erstellung des Projekts am besten geeignet ist, muss analysiert werden.

Für die Arbeit soll hierbei kein komplett neues DMS mit Nutzer- und Dokumentenverwaltung entwickelt werden. Vielmehr soll evaluiert werden, welche Systeme es bereits auf dem Markt gibt, und wie sich diese für die anstehende Aufgabe eignen, vor allem im Bezug auf die hierarchische Gliederung der Metadaten, wie sie im Projekt vorgesehen ist.

Zusätzlich zu den Systemen der LUBW soll das Literaturverzeichnis der *Literature Information System in the Field of ICT for Environmental Sustainability* (ICT-ENSURE) untersucht werden. Hierfür soll das Datenmodell herangezogen werden, um es wie die Systeme der LUBW zu analysieren.

Für die ICT-ENSURE soll jedoch keine Implementierung erfolgen.

Als sogenannter *Proof of Concept* (PoC) soll im Verlauf dieser Arbeit ein bestehendes Fachsystem auf der Basis einer erarbeiteten Schnittstelle nachgebaut werden. Diese Frontend soll das derzeitige *Fachdokumente Online* (FADO)-System der LUBW nachbilden.

3 Lastenheft

An das Projekt, welches im Rahmen dieser Arbeit bearbeitet werden soll, gibt es viele Anforderungen. Um diese Anforderungen möglichst strukturiert und übersichtlich darzustellen, wurde die Form eines Lastenhefts gewählt.

3.1 Zielsetzung

Ziel ist es, ein einheitliches ECM-System zu entwerfen, welches möglichst alle Fachdokumente der LUBW, der ICT-ENSURE und der GAA beinhaltet. Hierbei sollen alle Systeme über eine Schnittstelle mit dem DMS kommunizieren.

3.2 Produkteinsatz

Bei dem zu entwickelnden System, handelt es sich um einen Prototyp für ein DMS, welches gegebenenfalls von der LUBW, der ICT-ENSURE und der GAA eingesetzt werden kann. Das zu entwickelnde Backend soll alle Dokumente, welche schon heute auf den Websites zu finden sind, zusammenführen und diese in einer geeigneten Form ablegen.

3.2.1 Zusammenspiel mit anderen Systemen

Das Backend soll die heutigen Insellösungen für Dokumentenverwaltung der LUBW und der GAA vereinen. Es sollen sowohl die bestehenden als auch zukünftige Systeme unterstützt werden. Die jeweils benötigten Daten, sollen hierfür über eine oder gegebenenfalls mehrere Schnittstellen erreichbar sein.

Für die ICT-ENSURE soll ein eigenes System entwickelt werden, welches jedoch auf der gleichen Metadatenbasis aufbauen soll wie das der LUBW.

3.3 Produktfunktionen

Schon bestehende Dokumente sollen aus den alten Systemen übernommen werden können ohne dass ihre Metadaten verloren gehen oder geändert werden.
/LF10/ Dies bedeutet, dass das neue System den schon vorhandenen Datenbestand abbilden muss.

Die Metadaten der einzelnen Dokumente sollen fachlich und technisch gegliedert werden. Hierbei sollen gegebenenfalls Gruppierungen erstellt werden. Zum Beispiel sollen Longitude und Latitude zur Metadaten-Gruppe „Geodata“ zusammengefasst werden.
/LF20/

Die Dokumente sollen ebenfalls nach den einzelnen Anwendungen und Fachbereichen gruppiert werden. Hier sollen die Anwendungen auch speziell ihre Dokumente nach Gruppenzugehörigkeit abfragen können. Dies ist wichtig, da im FADO nur die entsprechenden FADO-Dokumente angezeigt werden sollen.
/LF30/

Eine Suche über die Dokumente und ihre Metadaten soll grundsätzlich möglich sein. Hier muss der Nutzer aber auch in der Lage sein, die Suche weiter einschränken oder nur nach bestimmten Metadaten suchen zu können.
/LF40/

/LF50/ Die alten Systeme sollen durch das neue Backend keinerlei Einschränkungen im Funktionsumfang unterworfen sein. Dies muss gegebenenfalls durch verschiedene Arten von Schnittstellen realisiert werden.

/LF60/ Für den Frontend-Prototyp, soll die Kernfunktionalität einer Schnittstelle funktionsfähig sein, anhand dessen der Prototyp evaluiert werden kann.

3.4 Produktdaten

/LD10/ Die entsprechenden Metadaten zu Dokumenten sollen durch das Backend verwaltet werden. Hierbei sollen nicht nur die manuell hinzugefügten Metadaten beachtet werden, sondern auch die Metadaten, welche das Dateiformat liefert und solche die das Dokument enthält.

/LD20/ Metadaten sollen nach Möglichkeit so zusammengefasst werden, dass eine Oberklassenbildung und Vererbung von Metadaten möglich ist. So soll zum Beispiel ein Standort alle Adressdaten zusammenfassen. Ein Standort wiederum kann in mehreren Dokumenten verwendet werden, wie zum Beispiel in Bildern oder in einem Forschungsbericht. Wiederum könnte der Standort aber auch Bestandteil der Unterklasse Gericht sein und somit den Standort des Gerichts abbilden.

/LD30/ Metadaten sollen ähnlich wie objektorientierte Klassen in Programmiersprachen agieren. So wird ein Autor „Max Mustermann“ nur einmal angelegt. Er kann jedoch in mehreren Dokumenten auftauchen. Sucht ein Nutzer nach „Max Mustermann“, so werden ihm alle Einträge zu diesem angezeigt. Verglichen werden kann diese Art von Abbildung auch mit einer relationalen Abbildung wie sie in Datenbanken zu finden ist.

/LD40/ Für die Gliederung der Metadaten sollen auch schon bestehende Standards betrachtet und gegebenenfalls umgesetzt werden. Im Verlauf der Arbeit muss evaluiert werden, welche Standards es für Metadaten gibt und wie sie im Projekt umgesetzt werden müssen und können. [Wik15c] [Wik13b] [Wik15e] [Wik13a]

3.5 Produktleistungen

/LL10/ Das Backend soll für Dokumente, welche in verschiedenen Versionen vorliegen, eine Art Versionsverwaltung bieten, sodass auch ältere Versionen zugänglich sind. Dies ist zum Beispiel bei Gesetzestexten wichtig, da hier immer die zum Vorfall aktuelle Version betrachtet werden muss.

/LL20/ Die Dokumente sollen über verschiedene Schnittstellen wie zum Beispiel eine *Representational State Transfer* (REST)-Schnittstelle abgerufen werden können. Der Einsatz von *Application Programming Interface* (API)s ist je nach ECM-Tool gegebenenfalls auch möglich, jedoch ist eine solche Umsetzung nicht Teil der Arbeit. [Wik15i]

/LL30/ Eine Suche über Metadaten muss realisiert werden. Hierbei soll dem Nutzer zum einen eine grobe und zum anderen eine verfeinerte eingeschränkte Suche nach bestimmten Metadaten geboten werden.

4 Stand der Technik / existierende Konzepte

Im folgenden Kapitel werden die einzelnen Fachsysteme der LUBW und deren Datenbestände genauer analysiert. Zusätzlich wird ein Blick auf das Literaturarchiv der ICT-ENSURE gerichtet, um ein möglichst übergreifendes Metadatenmodell erstellen zu können.

Welche Metadaten die einzelnen Dokumentenbestände der Fachsysteme genau enthalten, ist im Anhang 14.1 genau aufgelistet. Hierbei wird unterschieden, ob es sich um Metadaten oder Relationen handelt. Dafür wurden alle Metadaten aus den Systemen extrahiert.

Die Metadaten der in diesem Kapitel untersuchten Systeme werden anschließend im Kapitel 6 zu einem übergreifenden Metadatenmodell vereint, welches die spätere Grundlage für die Implementierung in einem ECM-Tool sein soll.

4.1 FADO und Untergruppen

Das FADO-System der LUBW erlaubt es den Nutzern nach verschiedenen Texten aus unterschiedlichen Themenrichtungen zu suchen und diese herunterzuladen. Hierbei stehen die Dokumente im PDF-Format bereit.

Die einzelnen Veröffentlichungen können nach verschiedenen Kriterien durchsucht werden, wobei der Bestand an Dokumenten von der LUBW fortlaufend erweitert wird. [LUB]

Im FADO-System sind die Dokumente in die sechs Kategorien **Altlasten**, **Boden**, **Natur und Landschaft**, **Umweltbeobachtung**, **Umweltforschung** und **Umweltinformationssysteme** gegliedert. Hierbei ist zu beachten, dass ein Dokument nicht nur einem Themengebiet zugeordnet werden kann, sondern durchaus unter mehreren Gebieten zu finden ist. Solche Dokumente sind im FADO jedoch nur einmal abgespeichert und werden über Relationen den verschiedenen Kategorien zugeordnet.

Die einzelnen Kategorien sind wiederum mit Untergruppen versehen, welche die Zugehörigkeit der Dokumente konkretisieren.

Da im Verlauf der Arbeit nicht alle Gruppen in ein neues System überführt werden können, wird sich auf die Dokumentklassen **Berichte**, **Urteile** und **Forschungsvorhaben** beschränkt, welche in der Kategorie **Natur und Landschaft**, in der Kategorie **Boden** beziehungsweise in der Kategorie **Umweltforschung** zu finden sind.

Forschungsvorhaben wiederum sind Berichte, welche sich in eine Skizze, einen Zwischen- und einen Abschlussbericht aufteilen. Hierbei gehören die drei Teile zu jeweils einem Forschungsvorhaben. [LUB]

4.2 Das Document Retrieval System

Das *Document Retrieval System* (DRS) der LUBW ist eine Dokumentenverwaltung mit einer Art Suchmaschine, die es ermöglicht verschiedene Rechtsvorschriften, Regelungen, Fachberichte und Erlässe zu suchen. Abfragen im DRS können auf drei Arten erfolgen. Es gibt die „Standardsuche“, welche es erlaubt, nach Inhalt oder Metadaten der Dokumente zu suchen. Die „Titelsuche“ erlaubt es, wie der Name schon sagt, nach Titeln oder gegebenenfalls nach Normtiteln der Dokumente zu schauen. Als dritte Möglichkeit bietet das DRS eine „Gezielte Suche“ an, die es ermöglicht, Suchkriterien genau anzugeben und diese auch einzuschränken. [DRS08]

Zu beachten ist, dass das DRS eine eigenständiges Plattform ist, welche einen eigenen Dokumentenbestand vorhält.

In Abbildung 1 ist die Suchmaske der „Gezielten Suche“ vom DRS zu sehen. Es werden in der Maske alle Möglichkeiten aufgeführt, nach denen gesucht werden kann, was alle vorhandenen Metadaten mit einschließt. Die meisten Metadaten sind durch Auswahllisten beschränkt und andere können vom Nutzer frei mit Begriffen gefüllt werden.

Zu sehen ist in den Feldern **Fassung** und **Änderung**, dass eine Art Verwaltung von älteren Versionen vorgenommen wird. Diese Versionierung ist notwendig, da alte Gesetzestexte für in der Vergangenheit liegende Fälle aufbewahrt werden müssen.

Volltextsuche:

- Suchmethode: **Suchmethode**

Gültigkeit: **hat Gültigkeitsvermerk** **Heute, 02.06.2015**

Titelsuche:

- Suchmethode: **einige Begriffe suchen**

Aktenzeichen: (abkürzen mit *)

Kurz-Titel: **+ alle +** Kurz- oder Norm-Titel ausgewählter Dokumente

Dokumentart: **+ alle +**

Herausgeber: **+ alle +**

Erscheinungsort: **+ alle +**

Handbuch: **+ alle +**

Kapitel: **+ alle +**

Fundstelle: **+ alle +** **Jahr(e):** **Seite(n):**

Fassung: **+ alle +** **1.** **2015** **Zeitpunkt:** **ab**

Änderung: **+ alle +** **1.** **2015** **Zeitpunkt:** **ab**

Abfragelogik: **UND (einschränkend)**

Sortierung: **automatisch optimal**

Ergebnisse: **Kurzinformationen** **10 je Seite**

Abfrage: **ausführen** **zurücksetzen**

Abbildung 1: Suchmaske des DRS

4.3 Naturschutz-Bildarchiv

Im Naturschutz-Bildarchiv der LUBW finden sich viele Bilder zu verschiedenen Themengebieten, so zum Beispiel auch zu den Gebieten **Biototyp**, **Lebensraumtyp**, **Naturschutzgebiet** und einige mehr. [Nat]

Diese Themenkomplexe können wiederum nach Stichworten durchsucht werden, wie zum Beispiel „Aurorafalter“, was im Themengebiet „Pflanzen- und Tierart“ Bilder von entsprechenden „Faltern“ liefert. Auch das Bildarchiv ist ein eigenständiges System, welches seinen eigenen Dokumentenbestand besitzt. In diesem Fall handelt es sich im Wesentlichen um Fotos.

4.4 Literaturarchiv der ICT-ENSURE

Im Literaturarchiv der ICT-ENSURE sind verschiedene Dokumente zu verschiedenen Themengebieten zu finden. Die ICT-ENSURE ist ein EU-Projekt, welches es sich zur Aufgabe gemacht hat die Zusammenarbeit von Forschung und Wissenschaft in Europa zu stärken.

ICT-ENSURE enthält, wie schon erwähnt, viele verschiedene Dokumente aus dem Bereich Forschung und erlaubt eine Volltextsuche innerhalb dieser Dokumente. Zusätzlich sind die einzelnen Dokumente nach verschiedenen Fachrichtungen beziehungsweise Konferenzen gegliedert. [SLG10]

Im speziellen sind auf der Homepage Dokumente zu verschiedenen wissenschaftlichen Tagungen zu finden. Alle Dokumente der ICT-ENSURE sind Bücher und so erfolgt auch ihre Speicherung. Zu jeder Konferenz gibt es ein Buch, welches in Kapitel unterteilt ist. Diese Kapitel wiederum sind in Unterkapitel geteilt, welche den einzelnen Vorträgen entsprechen. Jedes aufgelistete Buch entspricht einer Konferenz, die abgehalten wurde.

Eine genaue Auflistung der Metadaten ist im Anhang 14.2 zu finden.

5 Analyse von Metadatenstandards

Nach dem nun die Portale DRS, Naturschutz-Bildarchiv und FADO, der LUBW und das Literaturarchiv der ICT-ENSURE im Kapitel 4 vorgestellt und die Verwaltungsstrukturen aufgezeigt wurden, betrachtet dieses Kapitel nun Standards für Metadaten. Hierbei wird unterschieden, ob es sich um fachliche oder technische Metadaten handelt.

5.1 Fachliche Metadaten

Fachliche Metadaten sind Daten, welche den Inhalt einer Datei oder eines Dokuments genauer beschreiben und dem Anwender dabei helfen für ihn relevante Dateien zu identifizieren. Solche Metadaten sind immer fachspezifisch und unabhängig von den technischen Eigenschaften einer Datei. [Lei] [Fac] [DHW14] [Bla07]

Die Dokumente der Fachsysteme der LUBW, sowie der ICT-ENSURE enthalten alle fachlichen Metadaten. Eine genaue Aufstellung aller fachlichen Metadaten der in dieser Arbeit untersuchten Dokumente ist im Anhang 14.1 zu finden.

5.2 Fachliche Metadaten-Standards

Für die Erstellung eines Datenkonzepts wie es im Kapitel 6 geschieht, ist es aber nicht nur notwendig die vorhandenen Metadaten zu betrachten. Auch Standards für fachliche Metadaten sollen in diesem Umfeld untersucht werden.

Da fachliche Metadaten meist anwendungs- beziehungsweise dokumentbezogen sind, gibt es nicht sehr viele Standards, welche sich im Umfeld der LUBW beziehungsweise der ICT-ENSURE einsetzen lassen.

5.2.1 Darwin Core

Darwin Core beschreibt eine Zusammenfassung von Metadaten, welche für biologische Zwecke eingesetzt werden können. So ist es zum Beispiel möglich, Angaben zum Organismus oder zum Lebensraum zu machen.

Hierfür verwendet Darwin Core bis zu 172 Tags, welche jedoch nicht zwingend verwendet werden müssen. Zusätzlich enthält der Standard auch die Tags von Dublin Core (siehe Abschnitt 5.4.1) um das Dokument grundlegend zu beschreiben. [Wie15] [Wik15b]

Im Listing 1¹ ist einmal ein Beispiel für den Darwin Core-Standard mit XML dargestellt. Es ist zu sehen, dass ein „SimpleDarwinRecord“ verwendet wird, welcher nicht alle 172 Tags beinhaltet.

Es ist auch zu erkennen, dass die Dublin Core-Tags inbegriffen sind (beginnend mit „dcterms“). Die eigentlichen Tags des Darwin-Standards beginnen mit „dwc“.

¹<http://rs.tdwg.org/dwc/terms/guides/xml/index.htm>

```

1  <?xml version="1.0"?>
2  <dwr:SimpleDarwinRecordSet
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://rs.tdwg.org/dwc/xsd/simpledarwincore/ http://rs.
5          dwg.org/dwc/xsd/tdwg_dwc_simple.xsd"
6      xmlns:dcterms="http://purl.org/dc/terms/"
7      xmlns:dwc="http://rs.tdwg.org/dwc/terms/"
8      xmlns:dwr="http://rs.tdwg.org/dwc/xsd/simpledarwincore/">
9      <dwr:SimpleDarwinRecord>
10         <dcterms:type>PhysicalObject</dcterms:type>
11         <dcterms:modified>2009-02-12T12:43:31</dcterms:modified>
12         <dcterms:rightsHolder>Museum of Vertebrate Zoology</dcterms:rightsHolder
13             >
14         <dcterms:rights>Creative Commons License</dcterms:rights>
15         <dwc:institutionCode>MVZ</dwc:institutionCode>
16         <dwc:collectionCode>Mammals</dwc:collectionCode>
17         <dwc:occurrenceID>urn:catalog: MVZ:Mammals:14523</dwc:occurrenceID>
18         <dwc:basisOfRecord>PreservedSpecimen</dwc:basisOfRecord>
19         <dwc:country>Argentina</dwc:country>
20         <dwc:countryCode>AR</dwc:countryCode>
21         <dwc:stateProvince>Neuquen</dwc:stateProvince>
22         <dwc:locality>25 km al NNE de Bariloche por Ruta 40 (=237)</dwc:locality
23             >
24     </dwr:SimpleDarwinRecord>
25 </dwr:SimpleDarwinRecordSet>
```

Listing 1: Darwin Core-Beispiel in XML

5.2.2 INSPIRE

Infrastructure for Spatial Information in the European Community (INSPIRE) ist ein Standard für Metadaten, welcher von der *Europäische Union* (EU) nach der Richtlinie „2007/2/EG“ vom 14. März 2007 erlassen wurde. INSPIRE enthält Metadaten, welche für Geo-Referenzen benutzt werden müssen, denn nach der eben genannten EU-Richtlinie müssen alle vom Land veröffentlichten digitalen Dateien mit Geo-Referenzen versehen werden.

INSPIRE stellt 25 Meta-Tags zur Verfügung, mit dessen Hilfe die EU-Richtlinie zur Bekanntmachung von Geo-Referenzen eingehalten wird. Außerdem enthält der INSPIRE-Standard die notwendigen Tags der DIN EN ISO 19115 (siehe Abschnitt 5.2.3), wodurch dieser gleichzeitig nach dieser Norm ISO-konform wird. [INS08]

Von den 25 Meta-Tags müssen 12 Tags zwingend angegeben werden, um die Richtlinie zu erfüllen. Die Meta-Tags von INSPIRE sind zum Teil untergliedert, was bedeutet, dass mehr Information in diesen Standard enthalten sein können.

Aus Übersichtlichkeitsgründen wird an dieser Stelle kein Beispiel Listing erfolgen, da das XML-Format von INSPIRE sehr ausführlich und groß ist. Es wird jedoch an dieser Stelle auf den Editor für INSPIRE-Metadaten der Europäischen Kommission verwiesen, mit dessen Hilfe schnell und einfach XML-Dokumente mit INSPIRE-Metadaten erzeugt werden können². [INS15] [Wik13a]

²<http://inspire-geoportal.ec.europa.eu/editor/>

5.2.3 DIN EN ISO 19115

Die DIN EN ISO 19115 ist eine Norm, welche 2005 spezifiziert wurde. Sie enthält Metadaten für die Bescheinigung von Geo-Information. Mit über 400 möglichen Tags ist sie eine der detailliertesten Beschreibungsstandards für Geo-Daten.

Von den über 400 Tags sind für eine Verwendung der Norm nur ca. 22 Tags erforderlich. Alle anderen Tags sind optional. [Leh08] [Wik13b]

Wird der INSPIRE-Standard verwendet (siehe Abschnitt 5.2.2), so ist automatisch auch die ISO 19115 erfüllt, da diese Bestandteil von INSPIRE ist. [INS08]

5.2.4 ONIX

Der *Online Information Exchange* (ONIX)-Standard ist international bekannt und für den elektronischen Austausch von Buchinformationen geschaffen worden.

ONIX kann gemäß der Lizenz frei verwendet werden und ist zur Beschreibung von traditionellen Büchern und E-Books vorgesehen. Der Standard enthält sehr viele Tags, mit dessen Hilfe ein Buch beschrieben werden kann. So können zum Beispiel Angaben zum Verleger und zur ISBN gemacht werden. [ONI09a] [Wik15g]

Nachteilig ist, dass das Format sehr viel auf Abkürzungen setzt, was eine Menschenlesbarkeit erschwert oder gar verhindert.

Ein kleines Beispiel zum ONIX-Format ist im Listing 2 zu finden. Hier wird kurz beschrieben das es sich um einen Download handelt, welcher im Epub-Format vorhanden und lesbar ist. Ohne die Erklärungen am Ende der Zeile wären die Informationen nicht ohne weitere Hilfe lesbar.

```
1 <ProductForm>ED</ProductForm>                               Digital download
2 <ProductFormDetail>E101</ProductFormDetail>                 Epub format
3 <ProductContentType>10</ProductContentType>                 Readable text
```

Listing 2: ONIX-Beispiel in XML [ONI09b]

5.3 Technische Metadaten

Technische Metadaten sind Daten, die den internen Dateiaufbau und dessen technische Aspekte beschreiben. Sie betreffen den Anwender nur sekundär und sind eher wichtig für die richtige Speicherung und für die Auswahl von Viewern oder Parsern. Technische Metadaten sind unabhängig von dem fachlichen Inhalt eines Dokuments und beschreiben zum Beispiel dessen Datenformat, das Erstellungsdatum oder die Nutzerrechte genauer. [Lei] [Fac] [DHW14] [Bla07]

In den Systemen der LUBW und der ICT-ENSURE werden keine technischen Metadaten angezeigt, weshalb diese an der Stelle nicht betrachtet werden. Technische Metadaten sind vor allem bei physikalischen Daten wichtig. Da die Systeme nur intern mit diesen Daten arbeiten werden sie dem Nutzer nicht angezeigt. In einem ECM-System spielen sie jedoch eine wichtige Rolle und werden im späteren Verlauf der Arbeit genauer betrachtet (siehe Kapitel 6)

5.4 Technische Metadaten-Standards

Wie schon bei den fachlichen Metadaten (siehe Abschnitt 5.2) gibt es auch für technische Metadaten Standards, welche versuchen, die Beschreibung von Dokumenten zu vereinheitlichen. In

den folgenden Abschnitten werden einige Standards für technische Metadaten vorgestellt und genauer beschrieben.

5.4.1 Dublin Core

Dublin Core ist ein Standard, welcher ein Dokument grundlegend beschreibt. Von der *Dublin Core Metadata Initiative* (DCMI), welche den Standard beschlossen hat, werden 15 Kernfelder zur Verwendung, die so genannten „core elements“, empfohlen. Es gibt jedoch weitere Felder, welche zusätzliche Informationen enthalten können. [Wik15c] [Dub15] [Bla07]

Der Standard sollte heute Bestandteil jeder Website sein, da viele Suchmaschinen wie zum Beispiel Google nach Dublin Core-Metadaten suchen und Ergebnisse zum Beispiel anhand dieser Filtern. Ist also eine *search engine optimization* (SEO) angedacht beziehungsweise wie in FADO umgesetzt, sollten diese Metadaten vorhanden sein. [Fer13]

Ein Beispiel für Dublin Core ist im Listing 3³ zu finden. Hier wird der Standard im HTML-Format verwendet. Es wird zum einen auf das offizielle Schema verlinkt, zum anderen werden die einzelnen Metadaten dargestellt, welche in den Meta-Tags zu finden sind, und mit dem Namen „DC.“ beginnen, um zu verdeutlichen, dass es sich hier um Dublin Core handelt.

```
1 <head profile="http://dublincore.org/documents/dcq-html/">
2   <title>Dublin Core</title>
3   <link rel="schema.DC" href="http://purl.org/dc/elements/1.1/" />
4   <link rel="schema.DCTERMS" href="http://purl.org/dc/terms/" />
5   <meta name="DC.format" scheme="DCTERMS.IMT" content="text/html" />
6   <meta name="DC.type" scheme="DCTERMS.DCMIType" content="Text" />
7   <meta name="DC.publisher" content="Jimmy Wales" />
8   <meta name="DC.subject" content="Dublin Core Metadaten-Elemente,
9     Anwendungen" />
10  <meta name="DC.creator" content="Bjoern G. Kulms" />
11  <meta name="DCTERMS.license" scheme="DCTERMS.URI" content="http://www.gnu.org
12    /copyleft/fdl.html" />
13  <meta name="DCTERMS.rightsHolder" content="Wikimedia Foundation Inc." />
14  <meta name="DCTERMS.modified" scheme="DCTERMS.W3CDTF" content="2006-03-08" />
15 </head>
```

Listing 3: Dublin Core-Beispiel in HTML

5.4.2 Exif

Das Exif-Metadatenformat ist ein Standard für die Beschreibung von Bildern. In der heutigen Zeit verfügen nahezu alle digitalen Kameras über dieses Format und speichern bei der Aufnahme eines Bildes verschiedene Parameter im *Exchangeable Image File Format* (Exif) ab. [Wik15e]

Mit über 50 Tags kann ein Bild mit Hilfe von Exif sehr detailliert beschrieben werden. Es gibt unter anderem Tags für die Belichtungszeit, die Blende und den Farbraum. Aber auch Koordinaten, an dem das Bild aufgenommen wurde, können vorhanden sein. [Wik15e] [Exi13]

Eine genaue Auflistung aller Tags und deren Inhalt ist in der Spezifikation „Exchangeable Image File Format for digital still cameras“⁴ zum Format zu finden.

³http://de.wikipedia.org/wiki/Dublin_Core

⁴http://www.jeita.or.jp/japanese/standard/book/CP-3451C_E/#page=1

5.4.3 Andere Standards

Das *Resource Description Framework* (RDF) ist im eigentlichen Sinn keine Beschreibung von standardisierten Metadaten. Vielmehr ist es ein übergeordneter Standard, der beschreibt, wie Metadaten beschrieben werden können. [Wik15j]

Hierfür wird eine Satzstruktur verwendet, welche jeweils aus Subjekt, Prädikat und Objekt besteht. [Bla07]

Das ANSI/NISO Z39.87-2006 ist ein amerikanisches nationales Format, welches Bilddateien beschreibt. Es ist ähnlich aufgebaut wie Exif und enthält grundlegend die selben Möglichkeiten wie das im Abschnitt 5.4.2 vorgestellte Exif. [NIS06]

Da diese Formate im weiteren Verlauf der Arbeit keine Rolle mehr spielen, wird auf sie nicht weiter eingegangen.

6 Erstellung eines Datenkonzepts

Im nun folgenden Kapitel werden die zuvor in den Kapiteln 4 und 5 beschriebenen Metadaten der einzelnen Systeme zu einem umfassenden Datenkonzept zusammengefasst. Dieses Konzept bildet die Grundlage zur Speicherung in einem DMS.

Zuerst wird das Metadatenmodell möglichst weit „normalisiert“ und aufgebrochen. Dies ist wichtig um Gemeinsamkeiten bei dem Metadaten zu erkennen und diese zu extrahieren. Im späteren Verlauf muss dieses hoch modulare Modell wieder vereinfacht und an ein ECM-Tool angepasst werden (siehe Abschnitt 8.3).

Für die grafische Darstellung des Metadatenmodells wurde eine Darstellung nach UML gewählt, wobei eine Klasse eine Datensammlung darstellt.

Es wird explizit darauf hingewiesen, dass die Darstellung kein Klassendiagramm nach UML im eigentlichen Sinn ist und somit von der vorgeschriebenen Darstellungsform abgewichen wurde.

Attribute, welche sich direkt in der Sammlung befinden, sind ohne besondere Kennzeichnung einfach dargestellt. Andere Attribute, welche wiederum auf eine Datensammlung verweisen, sind mit dem jeweiligen Verweistyp gekennzeichnet. Zusätzlich wird mit Hilfe der Hintergrundfarbe sichtbar gemacht, in welchem Package sich die Datensammlung befindet.

Verweise sind zusätzlich über Pfeile gekennzeichnet, an welchen die Kardinalität des Verweises zu finden ist.

Um in der Arbeit eine Übersicht zu geben, werden die Packages einzeln beschrieben. Eine gesamte Übersicht des Diagramms ist auf dem der Arbeit beiliegenden Datenträger zu finden.

6.1 FADO Metadaten

In Abbildung 2 ist der erste Ausschnitt des Datenmodells zu sehen, welcher das Package **FADO Metadaten** zeigt.

Die Metadaten sind in vier Datensammlungen zusammengefasst, wobei die wichtigste **FADO Metadaten** ist. In dieser Sammlung sind dokumentübergreifende Metadaten zu finden, welche von allen FADO-Dokumenten verwendet werden. Zum Teil können hier alte Attribute wie **Unsichtbar** oder **Ausblenden** wiedergefunden werden. Aber manche Namen der Attribute haben sich auch geändert, weshalb eine Tabelle mit dem Mapping zwischen alten und neuen Namen im Abschnitt 6.5 zu finden ist.

Die drei Datensammlungen **FADO Urteil**, **Forschungsvorhaben** und **Bibliographische Angaben** sind die jeweiligen Hauptdatensammlungen der betrachteten Dokumente **Urteile**, **Forschungsvorhaben** und **Berichte**.

Attribute, welche farblich hinterlegt sind, stellen Verweise zu anderen Datensammlungen dar. Im vollständigen Diagramm werden diese Verweise zusätzlich durch Pfeile realisiert, welche die entsprechenden Kardinalitäten anzeigen.

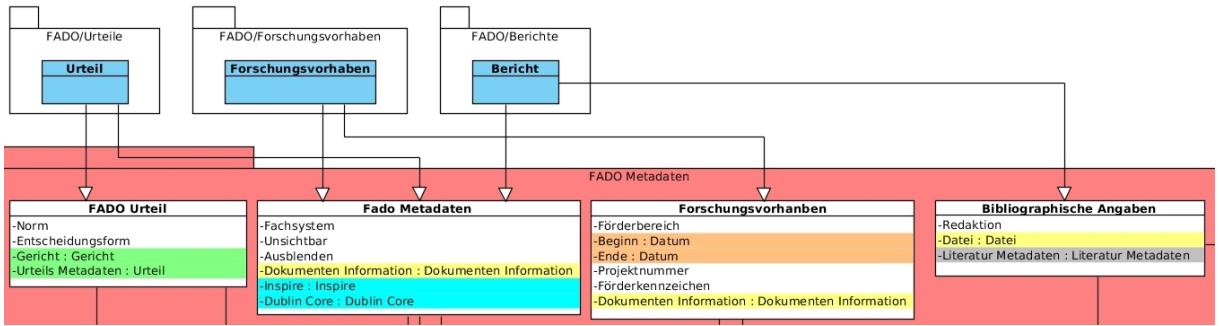


Abbildung 2: FADO Metadaten

6.2 DRS Metadaten / Bildarchiv Metadaten

Die Abbildung 3 zeigt die Datensammlungen des DRS und des Bildarchivs. Hier sind die obersten Datensammlungen zu sehen, welche zum einen eigene Attribute enthalten und zum anderen wieder auf untergeordnete Datensammlungen verweisen.

Das Mapping zu den Metadaten ist im Abschnitt 6.5 zu finden.

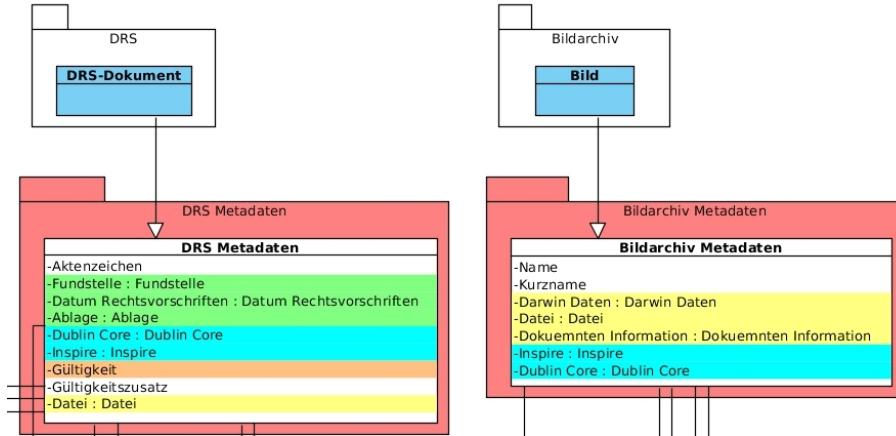


Abbildung 3: DRS und Bildarchiv Metadaten

6.3 ICT-ENSURE Metadaten

Abbildung 4 zeigt die Hauptdatensammlungen der Metadaten für ICT-ENSURE. Eine Besonderheit ist hier, dass die Datensammlung geteilt ist und zwar in Artikel Metadaten, welche die Metadaten zu einem Artikel enthält und in Konferenz, welche die Metadaten zu einer Konferenz enthält.

Alle farbig hinterlegten Verweise sind in den nachfolgenden Abschnitten genauer erklärt und das Mapping zwischen alten und neuen Namen ist im Abschnitt 6.5 zu finden.

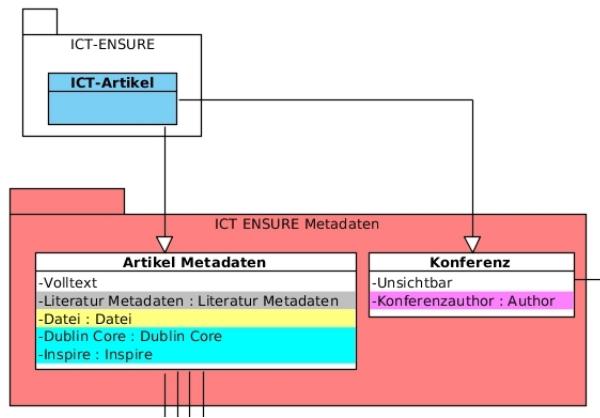


Abbildung 4: ICT-ENSURE Metadaten

6.4 Untergeordnete Datensammlungen

In den nun folgenden Abschnitten werden die eben in den Abbildungen 2 bis 4 farbig hinterlegten Datensammlungen mit ihren Attributen genauer beschrieben, wenn die Bezeichnung nicht schon eindeutig sein sollte.

6.4.1 Gerichtbarkeit

Im Package **Gerichtbarkeit**, welches in Abbildung 5 zu sehen ist, sind alle Datensammlungen welche für gerichtliche Urteile, Beschlüsse oder Richtlinien wichtig sind, zusammengefasst.

Die Datensammlungen **Gericht** und **Urteil** werden wie in Abbildung 2 zu sehen in der Datensammlung **FADO Urteil** verwendet.

Die Datensammlung **Gericht** stellt, wie der Name schon sagt, ein Gericht dar. Hierbei enthält sie die Attribute **Standort** und **Art**. Im Attribut **Standort**, wird der Standort des Gerichts und im Attribut **Art** die Art des Gerichts, wie zum Beispiel „Oberlandesgericht“ festgelegt. So ergibt sich für ein Gericht der Datensatz aus Standort und Art mit dessen Hilfe ein Datum nach der Art „Oberlandesgericht Karlsruhe“ gebildet werden kann.

Der Datensatz **Urteil** enthält die schon analysierten Attribute, welche das FADO-System verwendet. Die Attribute **Vorgericht** und **Nachgericht** verweisen wiederum auf ein Gericht. Das Erscheinungsdatum enthält eine Datensammlung des Typs **Datum**, welche im Abschnitt 6.4.6 genauer beschrieben ist.

Fundstelle, **Datum Rechtsvorschriften** und **Ablage** sind Datensammlungen, welche in der Sammlung **DRS Metadaten** vorkommen. Sie enthalten weitere Attribute zur Gerichtbarkeit, welche im DRS benötigt werden. (siehe Abschnitt 6.2)

Eine Fundstelle hat die Attribute **Name**, **Jahr** und **Seite**, welche eine Fundstelle, wie sie im DRS wiedergegeben wird, genauer beschreiben.

Alle Attribute der Datensammlung **Datum Rechtsvorschriften** verweisen auf ein Datum, welches im Abschnitt 6.4.6 genauer beschrieben ist.

Die Datensammlung **Ablage** enthält weitere Attribute, welche das DRS zur Beschreibung seiner Dokumente verwendet.

Das Mapping zu den gegebenen Änderungen in den Namen, der Attribute, ist im Abschnitt 6.5 zu finden.

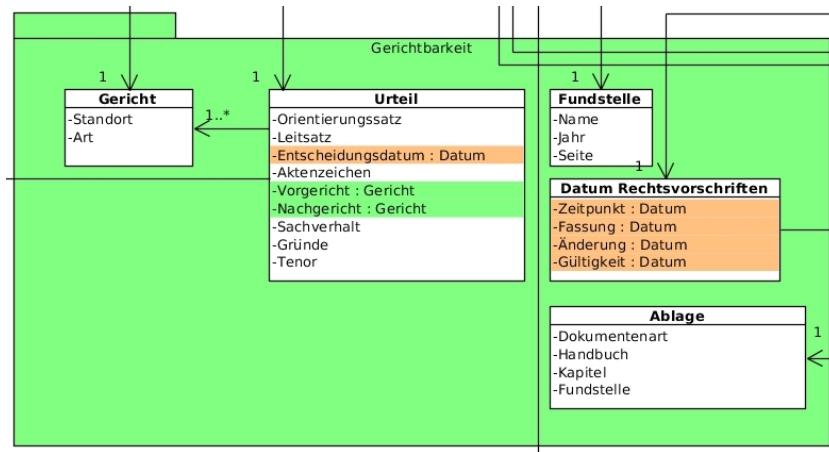


Abbildung 5: Package Gerichtbarkeit

6.4.2 Bibliographie Metadaten

Das Package **Bibliographie Metadaten**, welches in Abbildung 6 zu sehen ist, enthält zwei Datensammlungen, welche sich mit der Beschreibung von Literatur befassen.

Die wichtigsten Angaben stehen dabei in **Literatur Metadaten**, welche die Datensammlungen **Bibliographische Angaben** (Abbildung 2) und **Artikel Metadaten** (Abbildung 4) verwendet.

Die Attribute **Seitenanzahl**, **Abstract**, **Reihe**, **Bandnummer**, **Beginn Seite** und **End Seite** sind selbsterklärend und werden deshalb nicht genauer beschrieben. Das Attribut **Kapitel** verweist auf die gleichnamige Datensammlung und enthält wiederum Angaben zum Kapitel.

Die Attribute **PublikationsID** und **Dokumenten Information** verweisen auf Datensammlungen im Package **Abstrakte Metadaten**, welche im Abschnitt 6.4.3 genauer beschrieben sind.

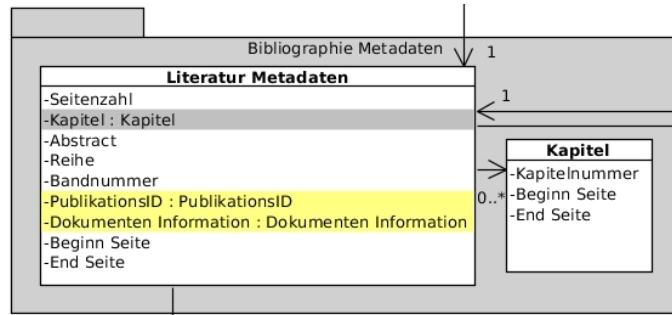


Abbildung 6: Package Bibliographie

6.4.3 Abstrakte Metadaten

Abstrakte Metadaten ist ein Package, welches Datensammlungen zu verschiedenen Themenbereichen enthält und ist in Abbildung 7 zu sehen.

Die Datensammlung **Dokumenten Information** enthält Attribute, welche Dokumente beschreiben. **Kommentar**, **Kurztitel**, **Kurzbeschreibung**, **Untertitel**, **Kurzname**, **Bemerkung** und **Version** sind selbsterklärend. Das Attribut **Stand** verweist wieder auf ein **Datum**, welches im Abschnitt 6.4.6 beschrieben ist.

Datei ist eine Datensammlung, welche verschiedene Attribute für eine „reale“ Datei bereithält. Die Attribute **Größe** und **Verfügbar** benötigen daher keiner weiteren Beschreibung, die beiden Verweise **URL** und **Technische Daten** schon.

Der Verweis **URL** beschreibt, wie der Name schon sagt, eine URL, unter der die beschriebene Datei zu finden ist (siehe Abschnitt 6.4.6). **Technische Daten** ist ein Verweis, welcher die Datei noch einmal genauer beschreibt (siehe Abschnitt 6.4.5).

PublikationsID beschreibt eine ID, wobei hier das Format und die genaue Art der ID dem Benutzer überlassen wird. Mit dem Attribut **Art** wird festgelegt, um welche standardisierte Form von ID es sich handelt. **Nummer** enthält dann die eigentliche ID, welche das Dokument besitzt. Hier können somit verschiedenste Systeme vom Benutzer frei verwendet werden.

Im Abschnitt 5.2.1 wurde der „Darwin Core“-Standard genauer untersucht. Es zeigte sich nun jedoch, dass der Einsatz von „Darwin Core“ zu umfangreich wäre, da nur ein Bruchteil der zur Verfügung stehenden Attribute überhaupt von der LUBW im Bildarchiv verwendet werden.

Aus diesem Grund wurde in Rücksprache mit der Arbeitsgruppe entschieden, eine eigene Datensammlung zu erstellen, welche die wenigen benötigten Attribute beinhaltet. Daraus entstand bei der Erarbeitung des Datenmodells die Datensammlung **Darwin Daten**, welche zwei Attribute enthält. Zum einen den lateinischen Namen und zum anderen den deutschen Namen. Weitere Biodaten werden nicht verwendet, da sie keine Verwendung in den Systemen finden.

Die letzten beiden Datensammlungen im Package sind **Person**, welche einen Namen und Vornamen enthält und **Land**, welche den Landesnamen und die ISO-Abkürzung nach ISO 3166-1 beinhaltet.

Beide Datensammlungen könnten durchaus weitere Attribute enthalten, dies ist jedoch nicht notwendig, da in den Systemen keine weiteren Daten zur Verfügung gestellt werden.

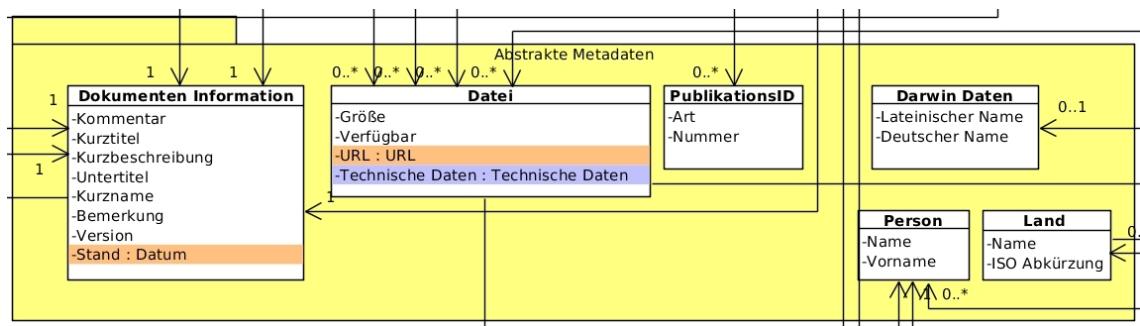


Abbildung 7: Package Abstrakte Metadaten

6.4.4 Standard Metadaten

Das Package **Standard Metadaten** enthält Datensammlungen für standardisierte Metadaten. Im Speziellen sind das **Inspire** und **Dublin Core**, welche in den Abschnitten 5.2.2 und 5.4.1 schon analysiert wurden.

Die Datensammlungen im Package enthalten keine eigenen Attribute, sondern verweisen lediglich auf die in ihnen enthaltenen Datensammlungen, welche im Abschnitt 6.4.5 genauer beschrieben werden. Ausnahme hierbei ist das Datum, welches **Inspire** inne hat und einen Verweis in das Package **Grundlegende Metadaten** darstellt. (siehe Abschnitt 6.4.6)

Im Anhang 14.4 ist die Abbildung 7 noch einmal vergrößert zu finden.

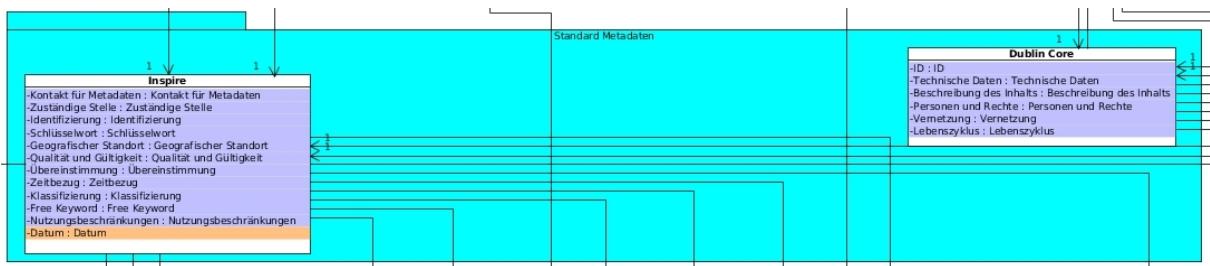


Abbildung 8: Package Standard Metadaten

6.4.5 Abstrakte Standard Metadaten

Abstrakte Standard Metadaten ist das Package, in welchem alle Datensammlungen zu finden sind, die in den Sammlungen im Package **Standard Metadaten** verwendet werden.

In Abbildung 9 sind die Datensammlungen zu sehen, welche bei INSPIRE Verwendung finden. Sie werden an keiner anderen Stelle referenziert, da die **Inspire**-Datensammlung in den jeweiligen Obersammlungen anzutreffen ist. (siehe Abschnitt 6.1 - 6.3 und Abbildung 2 - 4) Im Abschnitt 5.2.2 wurde dieser Standard schon einmal angesprochen und seine Funktionalität erklärt.

In der Datensammlung **Kontakt für Metadaten** sind die Attribute **Name der Stelle** und **EMail Adresse** vorhanden, mit denen der Ansprechpartner der Datei angegeben wird. Über **Zuständige Stelle** wird mit dem Attribut **Funktion der Stelle** und dem Verweis **Zuständige Stelle** die Stelle, welche die Datei veröffentlicht hat, genauer beschrieben.

Identifizierung gibt mit den Attributen **Ressourcenbezeichnung**, **Ressourcenüberblick**, **Ressourcenverweis** und **Ressourcensprache** die verwendete Ressource genauer an. Zusätzlich lassen sich **Bezeichner** einfügen. Alle Attribute sind Freitexte und können von der jeweiligen „Stelle“, welche die Datei veröffentlicht, vergeben werden. Einzig **Bezeichner** und **Ressourcensprache** sind Verweise und werden im Abschnitt 6.4.6 genauer beschrieben.

Zeitbezug enthält verschiedene Daten, welche auf die Sammlung **Datum**, im Abschnitt 6.4.6, verweisen. Im Speziellen sind das **Erstellungsdatum**, **Datum der Veröffentlichung** und **Datum der letzten Änderung**. Zusätzlich enthält die Sammlung noch einen Verweis auf **Zeitliche Ausdehnung**, welche im Abschnitt 6.4.6 beschrieben ist.

Die Datensammlung **Klassifizierung** enthält eine **Themenkategorie**, welche vorgegeben ist und zum Beispiel im Editor für INSPIRE gefunden werden kann⁵.

⁵<http://inspire-geoportal.ec.europa.eu/editor/>

Schlüsselwörter sind in INSPIRE fest vorgegeben und können ebenfalls im Editor der EU eingesesehen werden. Diese Schlüsselwörter werden im Model in der Datensammlung **Schlüsselwort** abgebildet.

Zusätzlich zu den vorgegebenen Schlüsselwörtern bietet INSPIRE auch die Möglichkeit, eigene, frei wählbare Schlüsselwörter, zu erstellen. Im Metadatenmodell ist dies in der Datensammlung **Free Keyword** umgesetzt. Diese enthält zum einen den Wert des Schlüssels und zum anderen einen Verweis **Herkunft des Vokabulars**, welcher im Abschnitt 6.4.6 genauer beschrieben wird.

Die Datensammlung **Geografischer Standort** beinhaltet Attribute für die vier Koordinaten, welche bei der Navigation auf der Erde nötig sind und zwar **N Breitengrad**, **E Längengrad**, **S Breitengrad** und **W Längengrad**. Hierbei werden zur Standortbeschreibung immer nur zwei benötigt, was davon abhängig ist, auf welcher Erdseite der Punkt sich befindet. Zusätzlich gibt ein Attribut das Land beziehungsweise die Länder (**Countries**) und den Ort (**Bezeichnung**) an.

Qualität und Gültigkeit enthält wiederum das Freitext-Attribut **Herkunft** und den Verweis **Räumliche Auflösung**, welches im Abschnitt 6.4.6 genauer beschrieben wird.

Nutzungsbeschränkungen enthält die Freitext-Attribute **Bedingungen für Zugang** und **Beschränkung des öffentlichen Zugangs**. Außerdem ist ein Verweis auf die Lizenz angegeben, welcher im Abschnitt 6.4.7 beschrieben wird. Es werden somit alle Bestimmungen für einen Zugang und Nutzung der Datei genauer erklärt.

Übereinstimmung ist eine Datensammlung für die Quellenangabe. Hierbei wird die Quelle im Attribut **Specifications** angegeben. Zusätzlich dazu ist ein Attribut **Grad** angegeben, welches beinhaltet ob die Quelle konform ist oder nicht. Über den Verweis **Datum mit Typ** wird angegeben, von wann die Quelle ist. (siehe Abschnitt 6.4.6)

Im Anhang 14.4 ist die Abbildung 9 noch einmal vergrößert zu finden.

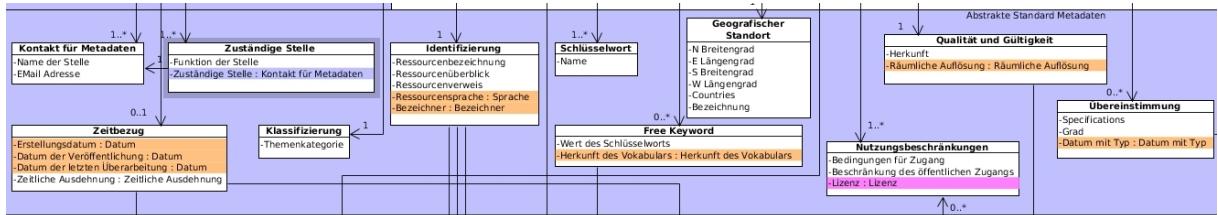


Abbildung 9: Package Abstrakte Standard Metadaten Teil 1 INSPIRE

In Abbildung 10 sind die Datensammlungen aufgezeigt, welche „Dublin Core“ verwendet. Diese befinden sich ebenfalls im Package **Abstrakte Standard Metadaten**. „Dublin Core“ wurde im Abschnitt 5.4.1 schon analysiert und nun im Modell verwendet.

Alle nun folgenden Attribute stammen aus der Norm und wurden im Inhaltsumfang gegebenenfalls noch weiter eingeschränkt und spezifiziert.

ID gibt mit dem Attribut **Identifier** eine eindeutige ID des Dokuments an, welche im System einmalig vergeben wird. Über diese ID sollen später eindeutige Verweise möglich sein.

Die Sammlung **Technische Daten** beinhaltet die Attribute **Format** und **Typ**. Zusätzlich wird ein Verweis auf eine **Sprache** gemacht (siehe Abschnitt 6.4.6). Über diese Felder wird eine „reale“ Datei genauer beschrieben.

Die Beschreibung des Inhalts wird mit der gleichnamigen Datensammlung **Beschreibung des Inhalts** umgesetzt. Hierfür stehen die Attribute **Titel**, **Thema**, **Reichweite** und **Beschreibung** zur Verfügung, welche eindeutig sind.

Personen und Rechte hat die beiden Attribute **Rechteverwerter** und **Herkunft**. Zusätzlich sind die beiden Verweise **Urheber** und **Herausgeber** zu finden, welche im Abschnitt 6.4.7 genauer erläutert werden. Der Verweis **Mitarbeiter** zeigt auf die Datensammlung Person und ist im Abschnitt 6.4.3 aufgeführt.

Mit der Datensammlung **Vernetzung** kommen die vier Attribute **Quelle**, **Verweis**, **Zielgruppe** und **Lehrmethode**. Die Felder **Quelle**, **Verweis** und **Zielgruppe** sind eindeutig und werden an dieser Stelle nicht weiter erläutert. Das Attribut **Lehrmethode** gibt an, mit welcher Lehrmethode sich die Daten am besten vermitteln lassen beziehungsweise wie sie vermittelt werden.

Lebenszyklus gibt einen Verweis auf ein Datum (siehe Abschnitt 6.4.6) und ein Attribut **Zustand**, mit dem sich beschreiben lässt, um was für ein Datum es sich handelt.

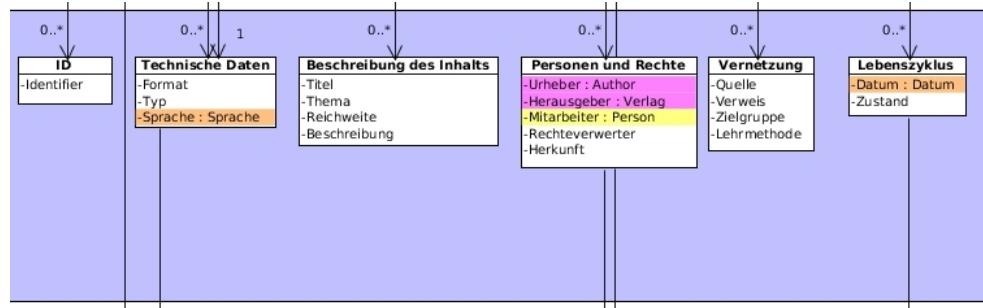


Abbildung 10: Package Abstrakte Standard Metadaten Teil 2 „Dublin Core“

6.4.6 Grundlegende Daten

Das Package **Grundlegende Daten**, welches in Abbildung 11 zu sehen ist, beinhaltet grundlegende Datensammlungen, welche an vielen Stellen im Modell verwendet werden.

Die Datensammlung **Datum** enthält die eindeutigen Attribute **Tag**, **Monat** und **Jahr**. Die Sammlung wird vielfach verwendet, wovon auch die Pfeile zur Sammlung in der Abbildung 11 zeugen.

Sprache enthält lediglich das eine Attribut **Name**, in welchem die Sprache des Dokuments angegeben wird.

URL enthält ebenfalls nur ein Attribut **Link**, welches einen internen oder externen Link repräsentiert.

Die Sammlung **Bezeichner** enthält die Attribute **Code** und **Namensraum**. Im INSPIRE-Standard gibt der **Code** eine eindeutige ID an, welche durch den **Namensraum** eingeschränkt und genauer spezifiziert wird. (siehe Abschnitt 6.4.5)

Mit der Datensammlung **Herkunft des Vokabulars** wird angegeben, aus welchem Vokabular das frei gewählte Schlüsselwort stammt. Zusätzlich wird ein Datum verlangt, wann das Schlüsselwort entstand, geändert oder veröffentlicht wurde, was mit Hilfe des Attributs **Datentyp** geschieht.

Zeitliche Ausdehnung ist eine Sammlung, welche ein **Anfangsdatum** und ein **Enddatum** als Verweis auf die Sammlung **Datum** enthält.

Räumliche Auflösung enthält drei Attribute und zwar **Äquivalenter Maßstab**, **Auflösungsabstand** und **Längeneinheit**. Sie werden benötigt, um eine genaue geografische Auflösung des Standortes zu gewährleisten.

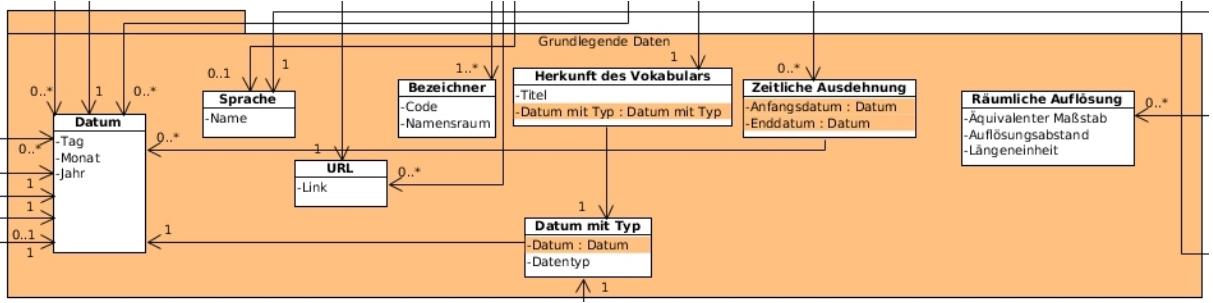


Abbildung 11: Package Grundlegende Daten

6.4.7 Urheber Metadaten

Im Package **Urheber Metadaten** sind Datensammlungen zusammengefasst, welche sich mit der Urheberschaft von Dateien befassen.

Verlag enthält die Attribute **Name** und **Ort**. Zusätzlich einen Verweis auf die Datensammlung **Land** aus dem Package **Abstrakte Metadaten**, welches im Abschnitt 6.4.3 beschrieben ist. Außerdem ist ein Verweis auf eine **Lizenz** zu finden, welche sich im selben Package befindet.

Die Datensammlung **Lizenz** hat das Attribut **Lizenzzart**, in welchem der Lizenzname festgehalten wird. Außerdem ist ein Verweis auf eine **Person** vorhanden, welche den Lizenzinhaber ausweist. (siehe Abschnitt 6.4.3)

Der **Autor** enthält das Attribut **Institution** und die Verweise auf **Person** und **Land**, welche zum Package **Abstrakte Metadaten** gehören und im Abschnitt 6.4.3 beschrieben sind.

Das Attribut **Institution** könnte weiter aufgeschlüsselt werden, dies ist jedoch nicht sinnvoll, da die Informationen von den Systemen nicht verarbeitet werden.

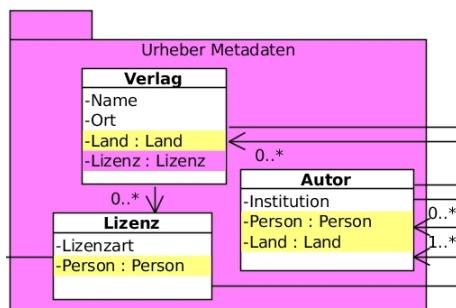


Abbildung 12: Package Urheber Metadaten

6.5 Metadatenmapping im neuen Modell

Da sich durch das Zusammenfassen und das Aufstellen eines übergreifenden Modells der Metadaten einige Bezeichnungen von Attributen geändert haben, wird in diesem Abschnitt nun das Mapping zwischen dem alten und neuen Modell beschrieben. Hierfür werden die Metadaten der einzelnen Systeme aufgeschlüsselt.

6.5.1 Metadatenmapping FADO

Im folgenden Abschnitt sind die Metadaten des FADO-Systems in Tabellenform aufgelistet. Die Auflistung der Metadaten und ihrer Wertebereiche ist im Anhang 14.1 zu finden.

In den dargestellten Tabellen 1, 2 und 3 sind die alten Namen neben den neuen Namen zu finden. Die zusätzliche dritte Spalte listet auf, in welcher Datensammlung die neuen Attribute zu finden sind.

Alt:	Neu:	Klasse:
Fachsystem	Fachsystem	FADO-Metadaten
ID	ID	ID
Titel	Titel	Beschreibung des Inhalts
Tenor	Tenor	Urteils Metadaten
Kommentar	Beschreibung	Beschreibung des Inhalts
Orientierungssatz	Orientierungssatz	Urteils Metadaten
Norm	Norm	Urteil
Leitsatz	Leitsatz	Urteils Metadaten
Gericht	Gericht	Urteil
Entscheidungsform	Entscheidungsform	Urteil
Entscheidungsdatum	Entscheidungsdatum	Urteils Metadaten
Aktenzeichen	Aktenzeichen	Urteils Metadaten
Vorgericht	Vorgericht	Urteils Metadaten
Nachgericht	Nachgericht	Urteils Metadaten
Sachverhalt	Sachverhalt	Urteils Metadaten
Gründe	Gründe	Urteils Metadaten
Unsichtbar	Unsichtbar	FADO-Metadaten
Ausblenden	Ausblenden	FADO-Metadaten

Tabelle 1: Mapping der FADO Attribute für Urteile

Alt:	Neu:	Klasse:
Fachsystem	Fachsystem	FADO-Metadaten
ID	ID	ID
Titel	Titel	Beschreibung des Inhalts
Kurzbeschreibung	Kurzbeschreibung	Dokumenten Information
Kommentar	Kommentar	Dokumenten Information
Förderbereich	Förderbereich	Forschungsvorhaben
Beginn	Beginn	Forschungsvorhaben
Ende	Ende	Forschungsvorhaben
Projektnummer	Projektnummer	Forschungsvorhaben
Förderkennzeichen	Förderkennzeichen	Forschungsvorhaben
Unsichtbar	Unsichtbar	FADO-Metadaten
Ausblenden	Ausblenden	FADO-Metadaten

Tabelle 2: Mapping der FADO Attribute für Forschungsvorhaben

Die Attribute **HTML-Datei**, **PDF-Datei**, **Weitere Datei** und **Format dieser Datei** werden komplett durch die Datensammlung **Datei** ersetzt, wie in der Tabelle 3 zu sehen ist.

Da während der Bearbeitung der Metadaten eine neue Version des FADO-Systems verabschiedet wurde, ergaben sich einige Änderungen in den Metadaten. Hierdurch entfallen die Attribute **Seiten (von-bis)**, **Shoprelevant**, **Shoplink** und **Preis** ersatzlos.

Alt:	Neu:	Klasse:
Fachsystem	Fachsystem	FADO-Metadaten
ID	ID	ID
Titel	Titel	Beschreibung des Inhalts
Kurzbeschreibung	Kurzbeschreibung	Dokumenten Information
Kommentar	Kommentar	Dokumenten Information
Kurztitel	Kurztitel	Dokumenten Information
Untertitel	Untertitel	Dokumenten Information
Fachthema	Thema	Beschreibung des Inhalts
Herausgeber	Herausgeber	Personen und Rechte
Redaktion	Redaktion	Bibliographische Angaben
Version	Version	Dokumenten Information
Stand	Stand	Dokumenten Information
Seitenzahl	Seitenzahl	Literatur Metadaten
Seite (von-bis)	ENTFÄLLT	ENTFÄLLT
Reihe	Reihe	Literatur Metadaten
Bandnummer	Bandnummer	Literatur Metadaten
ISSN	PublikationsID	Literatur Metadaten
ISBN	PublikationsID	Literatur Metadaten
Preis	ENTFÄLLT	ENTFÄLLT
Medium	Format	Technische Daten
Shoprelevant	ENTFÄLLT	ENTFÄLLT
Shoplink	ENTFÄLLT	ENTFÄLLT
HTML-Datei	Datei	Bibliographische Angaben
PDF-Datei	Datei	Bibliographische Angaben
Weitere Datei	Datei	Bibliographische Angaben
Format dieser Datei	Format	Technische Daten
Unsichtbar	Unsichtbar	FADO-Metadaten
Ausblenden	Ausblenden	FADO-Metadaten

Tabelle 3: Mapping der FADO Attribute für Berichte

6.5.2 Metadatenmapping DRS

Wie eben schon im Abschnitt 6.5.1 erklärt, wurden die Namen der Attribute auch für das DRS geändert. Die alte Bezeichnung ist in der linken Spalte, der neue Name im Modell in der mittleren Spalte und die zugehörige Datensammlung in der rechten Spalte, in der Tabelle 4, zu finden.

Die Auflistung der Metadaten und ihrer Wertebereiche ist im Anhang 14.1.4 zu finden.

Alt:	Neu:	Klasse:
Gültigkeit	Gültigkeit / Gültigkeitszusatz	DRS-Metadaten
Titel	Titel	Beschreibung des Inhalts
Aktenzeichen	Aktenzeichen	DRS-Metadaten
Kurz-Titel	Kurztitel	Dokumenten Information
Dokumentart	Dokumentart	Ablage
Herausgeber	Herausgeber	Personen und Rechte
Erscheinungsort	Herkunft	Personen und Rechte
Handbuch	Handbuch	Ablage
Kapitel	Kapitel	Ablage
Fundstelle	Fundstelle	Ablage
Fassung	Fassung	Datum Rechtsvorschriften
Änderung	Änderung	Datum Rechtsvorschriften
Größe	Größe	Datei
Formate	Format	Technische Daten

Tabelle 4: Mapping der DRS Attribute

6.5.3 Metadatenmapping Bildarchiv

Auch die Metadaten des Bildarchivs mussten zum Teil umbenannt werden um ein einheitliches Modell zu erstellen. Wie in den Abschnitten 6.5.1 und 6.5.2 ist die Tabelle 5 nach dem gleichen Muster aufgebaut. Links die alten Namen, in der Mitte die neuen Namen und Rechts die zugehörige Datensammlung.

Im alten Attribut **Bemerkung** steht der Name des biologischen Objekts in deutsch und lateinisch, dies wird im neuen System durch die Datensammlung **Darwin Daten** getrennt verwaltet und gespeichert, was ein Abrufen der einzelnen Attribute deutlich vereinfacht.

Die Auflistung der Metadaten und ihrer Wertebereiche ist im Anhang 14.1.5 zu finden.

Alt:	Neu:	Klasse:
Objektart	Thema	Beschreibung des Inhalts
Objektnname	Lateinischer Name	Darwin Daten
ID	ID	ID
URL	URL	Datei
Dateityp	Format	Technische Daten
Name	Name	Bildarchiv-Metadaten
Kurzname	Kurzname	Bildarchiv-Metadaten
Erstellt am	Datum	Lebenszyklus
Autor	Urheber	Personen und Rechte
Besitzer	Rechteverwerter	Personen und Rechte
Bemerkung	Darwin Daten	Darwin Daten

Tabelle 5: Mapping der Bildarchiv Attribute

6.5.4 Metadatenmapping ICT-ENSURE

Die Tabelle 6 folgt dem Schema aus den Abschnitten 6.5.1, 6.5.2 und 6.5.3.

Dadurch, dass die Metadaten der ICT-ENSURE ursprünglich in einer relationalen Datenbank abgelegt wurden, entfallen hier nun einige Attribute, über die früher die Relationen abgebildet wurden.

Die Auflistung der Metadaten und ihrer Wertebereiche ist im Anhang 14.2 zu finden.

Alt:	Neu:	Klasse:
Editor	Konferenzautor	Konferenz
Publisher	Herausgeber	Personen und Rechte
Year of Publishing	Datum	Lebenszyklus
ISBN	PublikationsID	Literatur Metadaten
Conferenc	Konferenztitel	Konferenz
Editor	Urheber	Personen und Rechte
Publisher	ENTFÄLLT	ENTFÄLLT
Year of Pblishing	ENTFÄLLT	ENTFÄLLT
ISBN	ENTFÄLLT	ENTFÄLLT
Conferenc	Thema	Beschreibung des Inhalts
Name	Titel	Beschreibung des Inhalts
Konferenz	ENTFÄLLT	ENTFÄLLT
Kapitel	ENTFÄLLT	ENTFÄLLT
Author	ENTFÄLLT	ENTFÄLLT
Dateityp	Format	Technische Daten
Titel	ENTFÄLLT	ENTFÄLLT
Sprache	Ressourchensprache	Sprache
Beginn Seite	Beginn Seite	Literatur Metadaten
End Seite	End Seite	Literatur Metadaten
Schlüsselwörter	Schüsselwort / Free Keyword	Inspire
Abstract	Abstract	Literatur Metadaten
Volltext	Volltext	Artikel-Metadaten

Tabelle 6: Mapping der ICT-ENSURE Attribute

7 Technologievergleich

Für das Projekt soll ein DMS verwendet werden, mit welchem sich die vorhandenen Dokumente der LUBW, der GAA und der ICT-ENSURE einfach und bequem in einem einzigen System verwalten lassen. Hierbei soll kein ECM-Tool von Grund auf neu entwickelt werden, sondern eine Standardsoftware den Erfordernissen angepasst werden.

Es soll, wie in der Aufgabenstellung im Lastenheft (siehe Kapitel 3) festgehalten, ein passendes Tool gesucht werden, welches die gegebenen Anforderungen bestmöglich erfüllt. Die Betrachtung erfolgt hierbei unter der Beachtung gängiger Standards.

Das ECM-Tool, welches für das Projekt verwendet werden soll, muss die im Folgenden genannten Eigenschaften aufweisen:

- Grundlegende Metadatenstandards wie zum Beispiel „Dublin Core“ und „EXIF“ müssen unterstützt werden (siehe Kapitel 5)
- Das System muss alle Fachsysteme der LUBW vereinen können, welche im Kapitel 4 beschrieben sind
- Metadaten sollten vom System systematisch gegliedert werden können, wie es im Kapitel 6 erarbeitet wurde
- Das verwendete ECM-Tool sollte möglichst viele Schnittstellen bieten, über welche die Daten abgerufen werden können (REST, *Content Management Interoperability Services* (CMIS), API)
- Dateien müssen versioniert werden können, um auch ältere Versionen einsehen zu können
- Da das Projekt als Prototyp behandelt wird, sollen für die Umsetzung keine oder nur geringe Lizenzkosten entstehen
- Open Source wird in politischen Softwareprojekten immer gern gesehen, weshalb dies auch ein Kriterium für eine sätere Umsetzung des Projekts sein kann

Im Folgenden werden nun verschiedene namenhafte ECM-Tools vorgestellt und auf die eben genannten Eigenschaften geprüft. Am Ende werden die verschiedenen Produkte verglichen und das beste ausgewählt.

7.1 agorum core

Bei „agorum core“ handelt es sich um eine Open Source Software, welche von der baden-württembergischen Firma „agorum Software“ stammt. „agorum core“ wird in mehreren Varianten angeboten. Zum einen gibt es eine freie Version, welche ohne Lizenzkosten genutzt werden kann, zum anderem gibt es mehrere kostenpflichtige Versionen, die in ihrem Versionsumfang variieren. [ago15d]

Agorum bietet viele zusätzliche Funktionen an, welche jedoch hinzugebucht werden müssen. Um die zusätzlichen Funktionen zu nutzen, muss entweder die entsprechende Version, welche die Funktion enthält, gekauft werden oder es muss die entsprechende Funktion hinzugebucht werden. Somit entstehen auf jeden Fall Kosten, wenn die freie Version von „Agorum Core“ nicht die gewünschten Funktionen bietet. Weiterhin fällt negativ auf, dass ein Hinzubuchen von Funktionen unter der freien Version nicht möglich ist. [ago15b] [Dam11]

In Abbildung 13⁶ ist das Web-Interface von „agorum core“ zu sehen, wobei hier im Speziellen der „Metadaten Designer“ zu sehen ist. Dieses Tool ist jedoch ein Zusatzfeature, welches entsprechend hinzugebucht werden muss, was nur innerhalb einer „Pro“-Version von „agorum core“ möglich ist. [ago15a]

Der „Metadaten Designer“ kann verwendet werden, um nutzerspezifische Metadatensätze zu erstellen. Da für die Arbeit keine „Pro“-Version von „agorum core“ gekauft wurde, kann auf die genaue Verwendung leider nicht eingegangen werden. [ago15c]

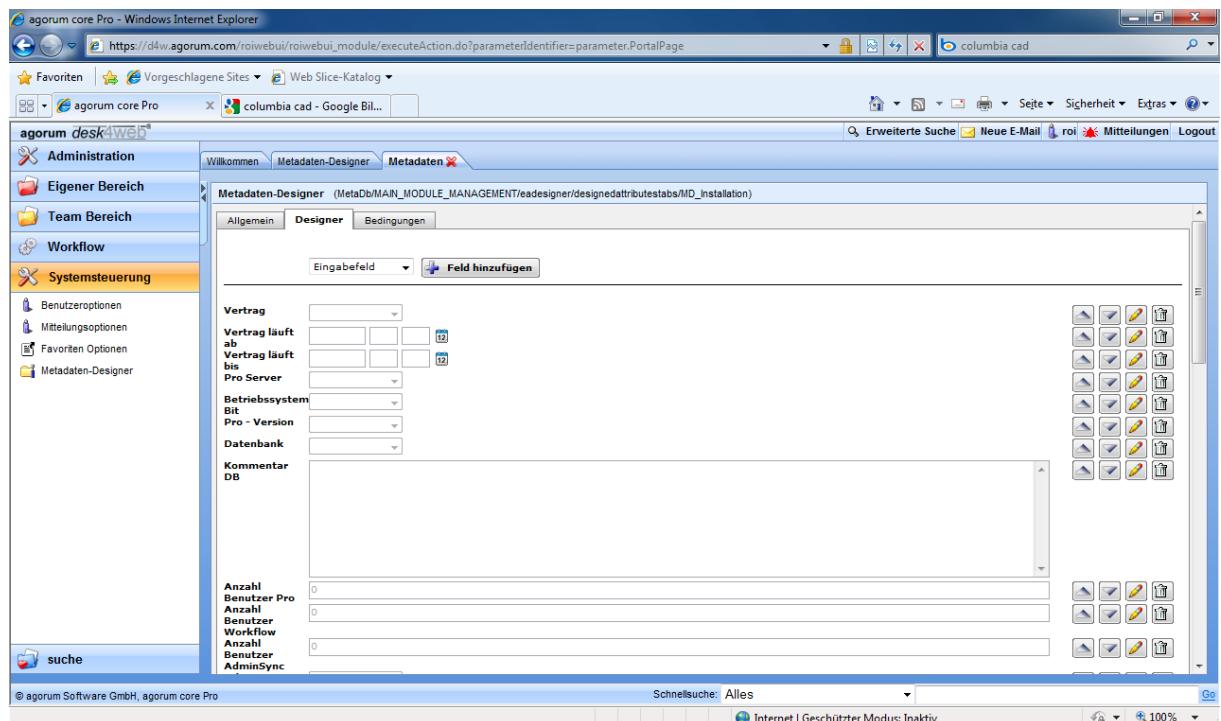


Abbildung 13: Metadaten Designer von „agorum core“ im Web-Interface

Das Einlesen und Bereitstellen von Dokumenten in „agorum core“ funktioniert schon mit der freien Version. Jedoch kann hier nur eine „Standard-Suche“, welche nicht nach Metadaten sucht, und eine „Standard-Ablage“ verwendet werden. [ago15b]

Das automatische Erkennen von Standard-Metadaten in Dateien funktioniert jedoch auch nur mit der entsprechenden „Pro“-Version oder einer Zubuchung des Features genau wie die Versierung von Dateien.

„agorum core“ verfügt über verschiedene Schnittstellen, welche von Frontends angesprochen werden können.

7.2 Alfresco

Alfresco ist eines der heute am weitesten verbreiteten ECM-Tools, welches Open Source ist und in der Community Edition frei heruntergeladen werden kann. Die Enterprise Version bietet zusätzlichen vierundzwanzigstündigen Support. Flexibilität, Akzeptanz und Skalierbarkeit sind jedoch nur einige der Stärken, die Alfresco zum Open-Source-Marktführer der ECM-Branche gemacht haben. [Wen13] [Wik15a] [Sha06]

⁶http://www.agorum.com/uploads/pics/agorum-core-metadatendesigner_01.png

Alfresco stammt ebenfalls von einer deutschen Firma, der „Alfresco Software AG“, und ist in der deutschen Sprache verfügbar. Nach Angaben der Herstellers, nutzen namenhafte Unternehmen wie „Airbus“ oder „Michelin“ die Software. [Alf15c]

In Abbildung 14 ist das Administrator-Dashboard von Alfresco zu sehen, welches im Webbrowser dargestellt wird. Im oberen Bildteil ist die Navigationsleiste und das Dash mit möglichen ersten Schritten.

Links ist zum einen das Dash „Meine Sites“ zu sehen, in welchem alle Seiten zu sehen sind, denen der Nutzer angehört. Außerdem ist darunter ein Dash, welches Aufgaben anzeigt, die vom Benutzer abgearbeitet werden sollen. In Alfresco ist es üblich mit Dokumenten und Dateien auch Aufgaben zu verknüpfen und Workflows zu gestalten. Dies geht für jede Datei einzeln oder aber auch für einen Stack von Dateien.

Im Dash neben „Meine Sites“ sind die letzten Aktivitäten zu sehen. Hier kann der Nutzer einstellen, in welchen Zeitraum, für welche Aktivität oder für welche Elemente er Aktivitäten anzeigen lassen möchte. Im Dash darunter kann der Nutzer sehen, welche Dokumente er zuletzt hochgeladen oder geändert hat.

The screenshot shows the Alfresco Administrator Dashboard. At the top, there is a navigation bar with links: Home, Meine Dateien, Freigegebene Dateien, Sites, Aufgaben, Mitarbeiter, Repository, Admin-Tools, and a search bar. The main area is divided into several sections:

- Willkommen bei Ihrem Dashboard, Administrator**: A welcome message with a link to remove it.
- Los geht es...**: A section with four cards:
 - Lernen**: A question mark icon with the text "Lernen Sie in diesem Tutorial, was Sie mit Alfresco machen können." and a link "Tutorials ansehen".
 - Freigeben**: A globe icon with the text "Erstellen Sie eine Site, um Inhalte für andere Site-Mitglieder freizugeben." and a link "Site erstellen".
 - Personalisieren**: A user profile icon with the text "Aktualisieren Sie persönliche und geschäftliche Informationen, damit andere Sie besser kennenlernen können." and a link "Profil bearbeiten".
 - Cloud**: A cloud icon with the text "Registrieren Sie sich noch heute und sichern Sie sich Ihr Gratis-Netzwerk in der Cloud." and a link "Anmelden".
- Meine Sites**: A dashboard showing a list of sites:
 - Schneller Zugriff auf Ihre Sites**: A site entry for "Schneller Zugriff auf Ihre Sites" with a brief description and a link "Site erstellen".
 - Sample: Web Site Design Project**: A site entry for "Sample: Web Site Design Project" with a brief description and a link "Favorit".
- Meine Aktivitäten**: A dashboard showing a list of activities:
 - Folgen Sie, was sich auf Ihren Sites tut**: A list of activities with a brief description and a link "In den letzten 7 Tagen".
- Meine Aufgaben**: A dashboard showing a list of tasks:
 - Aktive Aufgaben**: A list of active tasks with a link "Workflow starten".
 - Ihnen zugewiesene Aufgaben überprüfen**: A list of tasks assigned to the user with a brief description and a link "Workflow starten".
- Meine Dokumente**: A dashboard showing a list of documents:
 - Habe ich kürzlich geändert?**: A list of recently changed documents with a link "Verfolgen Sie Ihre eigenen Inhalte".

At the bottom, there is a footer with the Alfresco Community logo and a note: "Lieferung frei ohne Support, Zertifizierung, Wartung, Garantie, Schadensersatz von Alfresco oder dessen Zertifizierten Partnern. Klicken Sie hier, um Support zu erhalten." and "Alfresco Software Inc. © 2005-2015. Alle Rechte vorbehalten."

Abbildung 14: „Administrator Dashboard“ von Alfresco

Wird eine Datei in Alfresco eingefügt, was per Drag & Drop geht, werden automatisch schon vorhandene Metadaten aus der Datei herausgelesen. Hierbei werden zum Beispiel bei Bildern auch Exif-Daten gelesen und angezeigt, wie in der Abbildung im Anhang 14.3 zu sehen ist.

Durch die „Sites“, welche Alfresco bietet, ist es ganz leicht möglich, für alle bestehenden Systeme der LUBW eine eigene „Site“ zu erstellen. Da Alfresco für jedes Dokument eine eindeutige ID vergibt, ist es sogar möglich, Dokumente von verschiedenen Seiten zu verlinken. [Dam11]

Die Voraussetzung, dass neue Metadaten hinzugefügt werden und diese gruppiert werden können, ist in Alfresco erfüllt. Eine Beschreibung des Modells muss für Alfresco in XML geschehen. [Pot07] [CNF⁺10]

Alfresco bietet die Möglichkeit, ein Frontend durch verschiedene Schnittstellen anzubinden. Eine Anbindung kann zum Beispiel über REST, über WebDAV, über CMIS oder die „Alfresco One API“ erfolgen. [Alf15a]

Auch eine Versionierung von Dokumenten ist möglich, wodurch eine neue Version die alte nicht einfach ersetzt. Alte Versionen können auch weiterhin eingesehen werden.

Da Alfresco Open Source ist, ist es nicht nur kostenlos (Community Edition), sondern kann auch mit eigenen Tools erweitert werden. Dies kann zum einen über direkte Programmierung im Quellcode geschehen oder zum anderen über Aspekte, da Alfresco aspektorientiert programmiert ist. [Wen13]

7.3 Open-Xchange

Open-Xchange, ist ein Groupware-Server, welcher von der Open-Xchange AG, die ihren Sitz in Nürnberg hat, entwickelt wird und als Open Source unter der *General Public License* (GPL) verfügbar ist. [Xch15] [Ihl08]

Ursprünglich wurde Open-Xchange als E-Mail- und Groupware-Lösung entwickelt und stellte somit eine Alternative zu „Microsoft Exchange“ dar. [Wik15h]

In Abbildung 15 ist die Dateianzeige einer Datei in Open-Xchange dargestellt. Das Bild wurde mit Hilfe der freien Vorschauversion, welche Open-Xchange bietet, erstellt⁷. Diese Vorschau ist gut zum Testen der Funktionalität des Servers, befindet sich aber zur Zeit noch in der Beta-Phase. Vorteil dieses Vorschau ist auf jeden Fall, dass kein Nutzer sich mehr mühsam einen Server zum Testen der Funktionalität aufsetzen muss.

Auffällig ist, dass die Oberfläche von Open-Xchange ähnlich wie die von Alfresco aussieht, was in Abbildung 14 gut zu erkennen ist. Die liegt zum einen an der ähnlichen Gestaltung der Oberflächen, zum anderen daran, dass beide das freie HTML und CSS Framework „Bootstrap“⁸ verwenden.

Im oberen Teil der Abbildung 15 ist die Navigationsleiste zu sehen, welche es ermöglicht, die gesamte Funktionalität schnell und einfach zu erreichen. Der linke Abschnitt zeigt einen Navigator, der alle Ordner, welche sich gerade im System befinden, darstellt. Mittig ist die Liste der Dateien zu sehen, welche sich im links gewählten Ordner befinden. Wählt der Nutzer nun eine Datei aus, so erscheint der rechte Bildabschnitt mit einer Dateivorschau und Informationen zur Datei.

Open-Xchange ist weder in der Lage, die schon vorhandenen Metadaten der Datei anzuzeigen, noch eigene Attribute hinzuzufügen.

Über viele Schnittstellen ist es möglich, Daten von Open-Xchange auszulesen, so zum Beispiel auch über eine HTTP-API. [Oxp15]

Der Schwerpunkt von Open-Xchange ist eher die Kommunikation über E-Mail und die Verwendung als Groupware. Als ECM-Tool, wie es für die Arbeit benötigt wird, ist es jedoch ungeeignet, da es weder Metadaten einlesen noch verarbeiten kann.

⁷http://beta.core.ox.io/welcome/#pooled_gold_source_en_eb14204d2e764492ad76045ff7fd4646,
trymeWelcome

⁸<http://getbootstrap.com/>

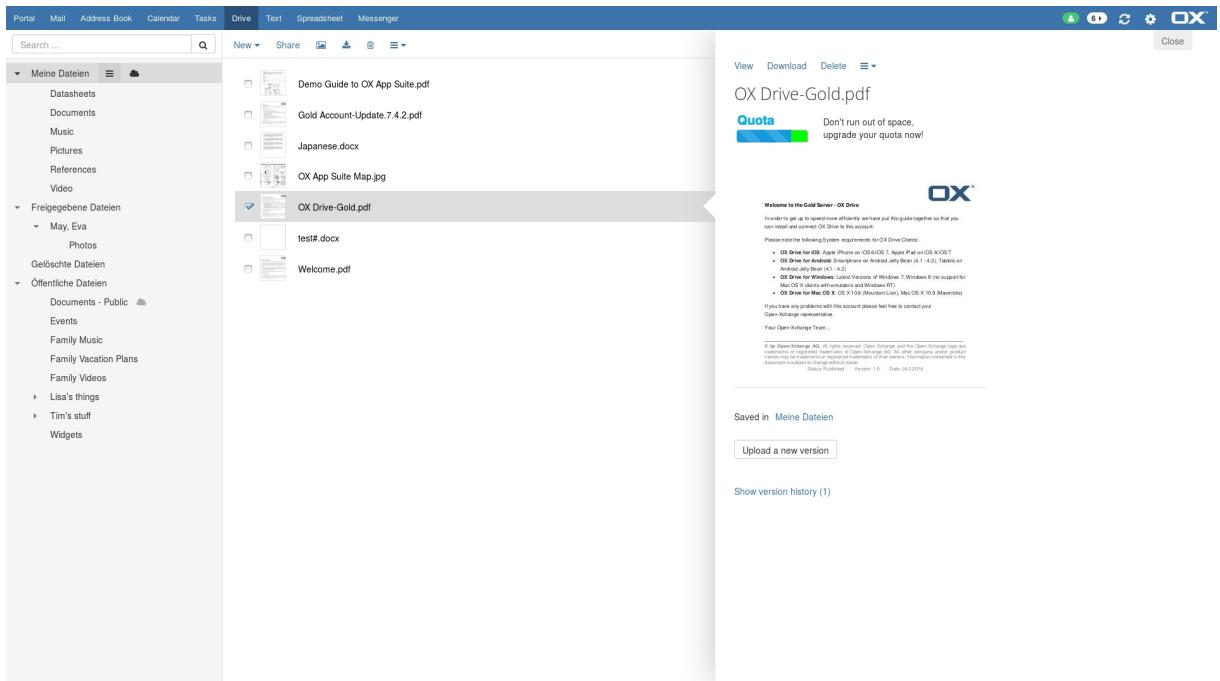


Abbildung 15: Datei-Anzeige von Open-Xchange

7.4 Liferay als ECM-Tool

Für die Entwicklung des Frontends soll Liferay⁹ verwendet werden, welches ein Tool für die Erstellung von Webportalen ist. Liferay wird im Kapitel 9 genauer beschrieben. Da Liferay auch eine Verwaltung für Dokumente besitzt, wird hier natürlich auch geprüft, ob eine Umsetzung des Projekts direkt auf Basis von Liferay möglich ist.

Liferay ist eigentlich als Portalserver ausgelegt, bietet jedoch die Möglichkeit die Dokumente, die auf einer Webseite benötigt werden, zu speichern und zu verwalten. Hierbei werden beim Hochladen von Dateien die standardisierten Metadaten ausgelesen und angezeigt, bei einem JPEG-Bild wären das zum Beispiel die vorhandenen Exif-Daten.

Für die gespeicherten Dateien bietet Liferay eine Versionierung innerhalb des Systems, ältere Versionen einer Datei können jedoch nicht von außen aufgerufen werden. Liferay verwendet auch keine IDs, um ein Dokument eindeutig zu identifizieren, verlangt aber eindeutige Namen bei der Speicherung von Dateien, wodurch IDs entfallen. Zur internen Datenbankspeicherung werden natürlich auch IDs verwendet, diese sind jedoch auf der Oberfläche nicht sichtbar und somit nicht nutzbar.

Es können im System zusätzlich zu den vorhandenen Metadaten eigene Metadatensätze erstellt werden, dies geschieht entweder per *User Interface* (UI) oder per XML-Definition. Liferay ermöglicht es nicht, Metadatensätze zu schachteln, wodurch die Strukturierung des Modells verloren gehen würde.

Ein Vorteil an Liferay ist, dass spezielle Dokumenten-Typen erstellt werden können, die alle Metadaten-Schlüsselelementen enthalten. So kann zum Beispiel ein Datentyp FADO-Urteil speziell für Urteile aus FADO erstellt werden. Leider ist es nicht möglich, einen Metadatensatz zweimal in einen Dokument zu verwenden, wie es bei der Datensammlung Urteil nötig wäre, da sie die Attribute **Vorgericht** und **Nachgericht** enthält, welche beide auf ein **Gericht** verweisen.

⁹<http://sourceforge.net/projects/lportal/files/Liferay%20Portal/>

Liferay bietet verschiedene Schnittstellen, um nach außen hin zu kommunizieren, so zum Beispiel WebDAV.

7.5 Auswertung der Möglichkeiten

Nach dem in den Abschnitten 7.1, 7.2 und 7.3 die drei wichtigsten Vertreter der ECM/DMS-Tools betrachtet wurden, muss nun eine Entscheidung getroffen werden, welches System sich am besten für das Projekt eignet. Zusätzlich wurde im Abschnitt 7.4 geprüft, ob Liferay als ECM-Tool geeignet ist.

Hierfür wurden die im Abschnitt 7 aufgeführten Anforderungen noch einmal in Tabellenform dargestellt (siehe Tabelle 7). Die einzelnen Systeme wurde in der Tabelle nebeneinander gestellt, wobei ein ✓ für volle Umsetzung im System, ein ✓ X für eine Umsetzung im System die zubuchbar und ein X für keine Umsetzung im System steht.

In der Tabelle 7 fällt auf, das „agorum core“ zwar für das Projekt geeignet wäre, jedoch durch die hohen Lizenzkosten nicht verwendet werden kann, wie es im Abschnitt 7.1 schon dargelegt wurde. Die für die Arbeit benötigten Komponenten sind kostenpflichtig und nur in Verbindung mit einer lizenzierten „Pro“-Version nutzbar. Es würden geschätzte Kosten von 17.000€ anfallen.

Open-Xchange ist eine interessante Alternative für das von Microsoft stammende Exchange, und ist kostenfrei, da es sich um Open-Source-Software handelt. Jedoch bietet Open-Xchange nicht die Unterstützung, welche für diese Arbeit benötigt wird. Vor allem in den beiden Punkten, wo es um die Gliederung von Metadaten und die Implementierung von Metadatenstandards geht, kann Open-Xchange nicht mithalten. Da diese Punkte eine sehr hohe Priorisierung haben, kann Open-Xchange nicht verwendet werden.

Das Verwalten, Versionieren und das Anzeigen von Dokumenten funktioniert sehr gut, jedoch wird von Seiten Open-Xchanges aus keine Unterstützung von Metadaten geboten. Eine eigene Implementierung dieser Funktionalität wäre zwar denkbar, da der Quellcode unter der GPL steht, jedoch im Rahmen des zeitlichen Umfangs der Arbeit nicht umsetzbar.

Liferay verwaltet zwar Dokumente, die auf einer Website benötigt werden, ist jedoch nicht in der Lage Metadaten so elegant zu verwalten wie zum Beispiel Alfresco. Zusätzlich ist es nicht möglich Dokumente in ihren Versionen zu verwalten. Aus diesen Gründen wurde der Ansatz mit Liferay als ECM-Tool nicht weiter verfolgt.

Die vielversprechendste Alternative ist Alfresco, welches alle Anforderungen erfüllt. Zum einen ist es Open Source und somit frei verfügbar, zum anderen bietet es sehr viele Schnittstellen an, um mit anderen Programmen zu kommunizieren. Standardmäßig liest Alfresco alle Metadaten einer Datei beim Hinzufügen in das System aus, und stellt diese auch dar. Zusätzlich lassen sich weitere eigene Metadaten zu den Dateien hinzufügen.

Das Verwalten von eingenen Metadaten geschieht hier auf der Basis von XML-Dokumenten, welche zu Alfresco hinzugefügt werden müssen. Hierzu jedoch mehr in Kapitel 8.

Anforderung	Priorisierung	agorum	Alfresco	Open-Xchange	Liferay
Metadatenstd. implementiert	sehr hoch	✓	✓	✗	✓
Mehrere Systeme in einem	gering	✓	✓	✓	✓
Gliederung von Metadaten	sehr hoch	✓✗	✓	✗	✗
Verschiedene Schnittstellen	hoch	✓	✓	✓	✓
Versionierung von Dateien	hoch	✓✗	✓	✓	✗
Kostenfreie Nutzung	hoch	✓✗	✓	✓	✓
Open Source	hoch	✓	✓	✓	✓

Tabelle 7: Tabellarischer Vergleich der betrachteten ECM-Systeme

8 Implementierung des Backends auf Basis von Alfresco

Für das Backend eignet sich der Analyse aus Kapitel 7 folgend am besten Alfresco. In den nun folgenden Abschnitten geht es um die Implementierung des Metadatenmodells aus Kapitel 6 in Alfresco. Hierfür wird zum einen die Installation und die Arbeitsweise erläutert und zum anderen wird im Hauptteil auf die Implementierung des Datenmodells in Alfresco eingegangen.

8.1 Installation

Die Installation von Alfresco ist denkbar einfach und unter Windows, sowie unter Linux möglich. Für den Download der Community Edition muss man sich mit einer gültigen E-Mail-Adresse bei Alfresco anmelden¹⁰.

Die Anmeldung hat den Vorteil, dass der Nutzer zu Webinars und anderen neuen Dingen rund um Alfresco immer auf dem Laufenden ist. Die kostenpflichtige Variante von Alfresco bietet zusätzliche technische Unterstützung und einige Enterprise Features, welche jedoch für diese Arbeit nicht benötigt werden. [Wik15a]

Nach dem Download führt unter Linux ein Skript die Installation durch. Hierbei wird der Nutzer ausführlich über alle durchgeführten Schritte informiert. Der Nutzer muss während der Installation ein Passwort für das Administrator-Konto angeben.[Wen13]

Ist die Installation abgeschlossen und der Server gestartet, kann Alfresco im Browser lokal unter <http://127.0.0.1:8080/share/page/> aufgerufen werden.

War die Anmeldung erfolgreich, gelangt der Nutzer auf das „Administrator Dashboard“, welches in Abbildung 14 im Abschnitt 7.2 zu sehen ist.

8.2 Metadatenmodell in Alfresco

Um ein eigenes Metadatenmodell in Alfresco einzufügen zu können, muss es mittels XML beschrieben werden. Unter **ALFRESCO_HOME** wird im Folgenden das Home-Verzeichnis des Alfresco-Servers zu verstehen sein.

In Abbildung 16 ist das Content-Modell von Alfresco in UML dargestellt. Das Hauptelement Klasse (Class) beschreibt den Aufbau eines Content-Modells und enthält Aspekte (Aspect) und Typen (Type) welche ganz oben im Diagramm zu sehen sind. [CNF⁺¹⁰]

Klassen im Content-Modell können ihre Eigenschaften, das heißt ihre Aspekte und Typen, an Unterklassen vererben. In Alfresco ist nur eine einfache und keine Mehrfachvererbung zulässig.

Eine Klasse kann Assoziationen auf andere Klassen (Peer Association) oder Objekte (Property) enthalten. Eine Assoziation auf eine andere Klasse ist unter Alfresco jedoch nur zulässig, wenn schon ein Content (Instanz der Klasse) der jeweiligen Klasse existiert, andernfalls muss er vorher angelegt werden. Objekte oder auch Attribute können aus einem Constraint oder einem einfachen Datentyp (Data Type) bestehen.

Um das Metadatenmodell, wie in Kapitel 6 beschrieben, umsetzen zu können, müssten Datentypen auch wieder Klassen enthalten können (Komposition). Dies ist jedoch unter keinem der beschriebenen ECM-Tools möglich. Jedoch bietet Alfresco den besten Ansatz, weshalb das Da-

¹⁰<https://www.alfresco.com/de/products/community/download>

tenmodell nun noch einmal für Alfresco angepasst werden muss. Wie genau das Metadatenmodell angepasst wurde, ist im Abschnitt 8.3 genauer erläutert.

Unter Alfresco kann eine Klasse mehrere Typen enthalten, ein Typ kann zum Beispiel ein Bild, ein Gesetz oder ähnliches sein. Jeder Typ kann Attribute beinhalten, welche fest mit ihm verbunden sind. Wird eine PDF-Datei in Alfresco also zu einem „Gesetz“ also zu ein Dokument mit dem Datentyp **Gesetz** umgewandelt, enthält es unweigerlich nach der Änderung alle Attribute, welche direkt im Typ beschrieben und verankert sind.

Ein Aspekt kann frei definiert und zu einem beliebigen Typ hinzufügt werden. Möchte ein Nutzer also die Attribute von INSPIRE hinzufügen, so reicht es, wenn er den entsprechenden Aspekt zu dem gewünschten Dokument hinzufügt. Die jeweiligen Datentypen müssen hierbei als Aspekt definiert sein.

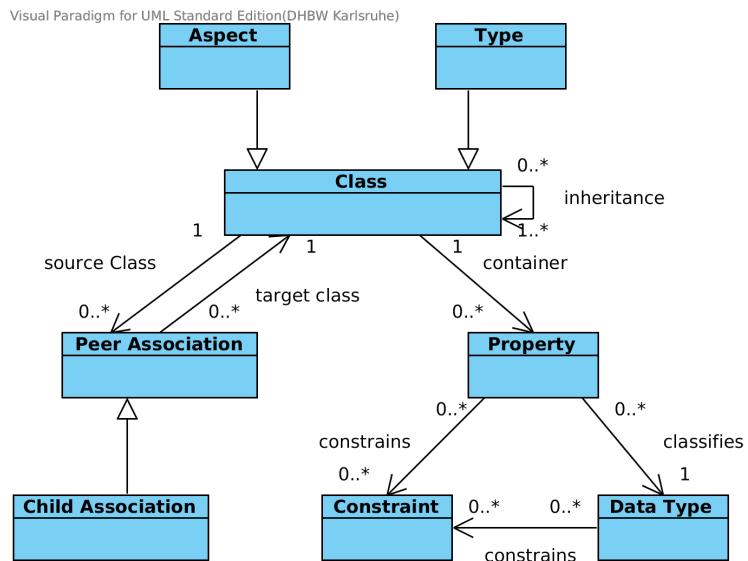


Abbildung 16: Content-Modell in Alfresco

8.3 Verändertes Datenmodell für Alfresco

Wie im Abschnitt 8.2 schon ausführlich beschrieben, ist es nicht möglich, die zuvor im Kapitel 6 erstellte Metadatenstruktur umzusetzen. Daher wird nun beschrieben, wie das Datenmodell für Alfresco angepasst werden kann.

Da Alfresco den „Dublin Core“-Standard bereits implementiert hat, kann dieser direkt genutzt werden. Auch die Datensammlung **Datei** entfällt, da die entsprechenden Daten automatisch bei jeder hochgeladenen Datei von Alfresco ausgelesen und erstellt werden.

Die im Abschnitt 6.1 gezeigten Datensammlungen werden nun zu Datentypen oder Aspekten, welche das jeweilige Dokument beschreiben. Alle farbig hinterlegten Attribute stellen Aspekte dar, die von anderen Klassen aus Alfresco übernommen wurden. Packages stellen in diesem Diagramm Klassen dar. Die UML-Klassen stellen Aspekte oder Typen dar. Worum es sich genau handelt, steht jeweils auch an den Datensammlungen.

Alle in den folgenden Abschnitten nicht aufgeführten Datensammlungen sind direkt in den Aspekten untergebracht, da Alfresco keine Klassen oder Typen als Attribute zulässt.

8.3.1 Die Datentypen

Die in den Abschnitten 6.1 bis 6.3 gezeigten Grunddatentypen bleiben im angepassten Modell weitestgehend identisch. Da unter Alfresco eine Datei nur von genau einem Datentyp sein kann, wurde die Datensammlung FADO Metadaten den anderen FADO-Sammlungen übergeordnet, wie in Abbildung 17 zu sehen ist. Der Typ Fado Metadaten findet sich nun in der Klasse Fado, welche wiederum die Oberklasse für die Typen FADO Urteil, Forschungsvorhaben und Bibliografische Angaben bildet.

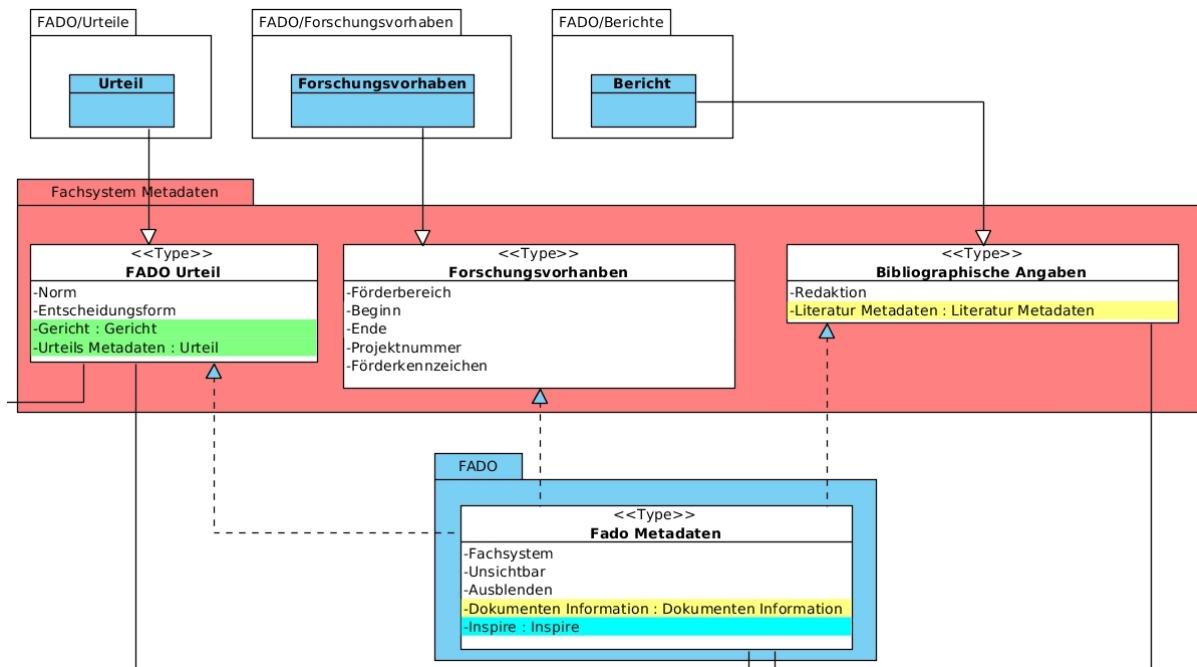


Abbildung 17: FADO Datentypen für Alfresco

Die Datentypen DRS-Metadaten und Bildarchiv Metadaten sind im Grunde gleich geblieben. Hier haben sich nur geringfügige Änderungen ergeben, da die Datensammlungen wieder zusammengefasst worden sind. Die beiden Datensammlungen Artikel Metadaten und Konferenz sind zusammengefasst worden und nun im Typ Artikel Metadaten zu finden. Die eben beschriebenen Änderungen sind in den Abbildungen 18 und 19 zu sehen.

Im Abschnitt 8.3.2 wird noch einmal genauer erwähnt, warum die Attribute Fundstelle, Datum Rechtsvorschriften und Ablage nicht mehr existieren.

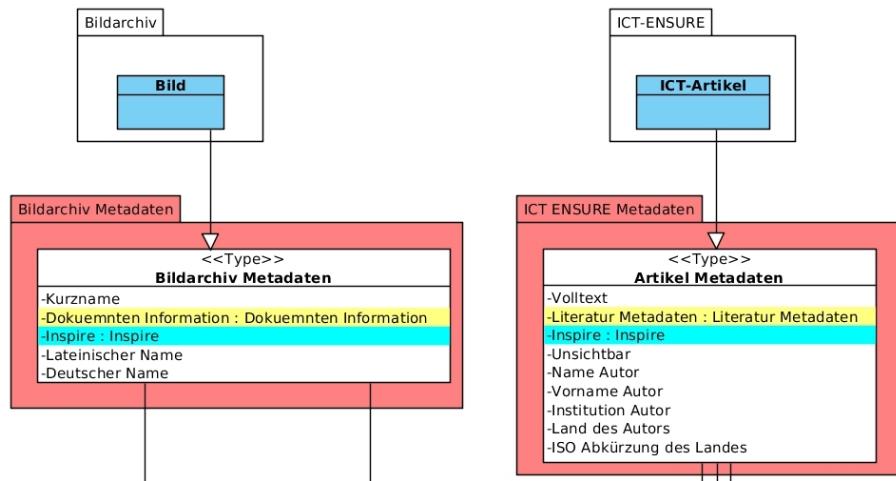


Abbildung 18: Bildarchiv und ICT-ENSURE Datentypen für Alfresco

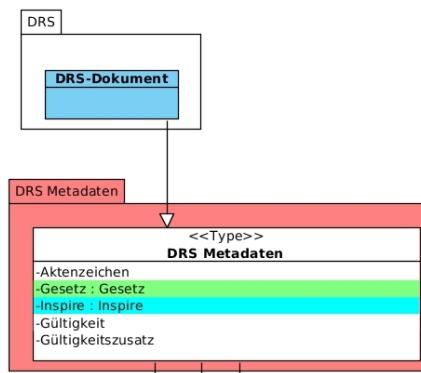


Abbildung 19: DRS Datentyp für Alfresco

8.3.2 Die Klasse Gerichtbarkeit

Das Package **Gerichtbarkeit** beschreibt eine Klasse, welche drei Aspekte beinhaltet. Diese Aspekte sind **Gesetz**, welcher die Datensammlungen **Fundstelle**, **Datum der Rechtsvorschriften** und **Ablage** zusammengefasst, **Urteil** und **Gericht**.

Die Datensammlung **Gericht** geht ohne Veränderung in einen Aspekt über und wird im Typ FADO **Urteil** verwendet. Die Sammlung **Urteil** musste etwas abgeändert werden. Die Verweise auf das Vorgericht und das Nachgericht wurden aufgeschlüsselt, da Alfresco bekanntlich keine Klassen als Attribute unterstützt.

In Abbildung 20 ist die Klasse **Gerichtbarkeit** noch einmal dargestellt. Zu beachten ist, dass das Package hier für eine gesamte Klasse steht und die eigentlichen Klassen nur Aspekte darstellen.

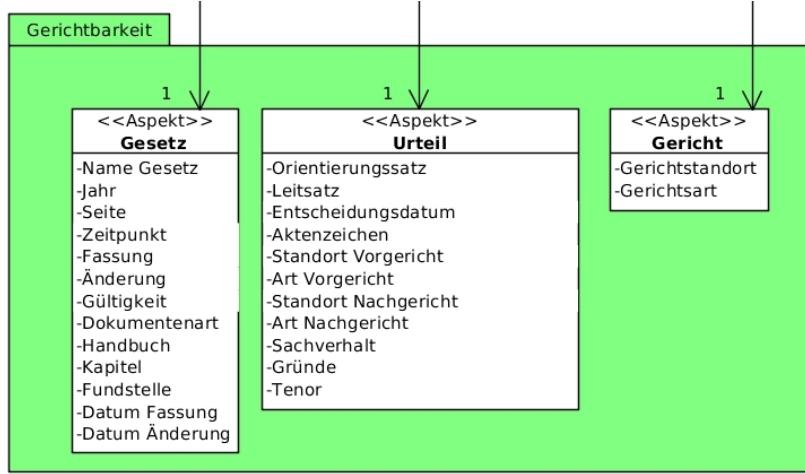


Abbildung 20: Die Klasse Gerichtbarkeit

8.3.3 Die Klasse Standard-Aspekte

Das Package **Standard-Aspekte**, ist wieder eine Metadaten-Klasse von Alfresco, welche den Aspekt **Inspire** enthält. Durch die Änderungen am Datenmodell wurden die Attribute, welche sich in den verweisten Datensammlungen befunden haben, nun direkt in der Datensammlung **Inspire** untergebracht, welche nun ein Aspekt ist.

Die Abbildung 21 zeigt noch einmal den Aspekt **Inspire**, welcher sich in der Klasse (dem Package) **Standard-Aspekte** befindet.

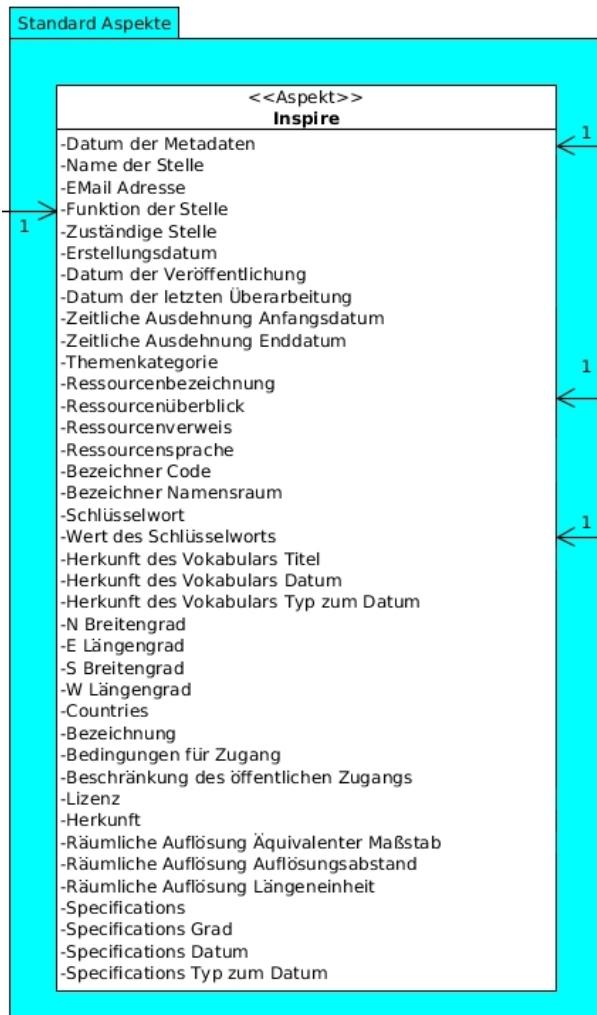


Abbildung 21: Die Klasse Standard Aspekte

8.3.4 Die Klasse Bibliografische Aspekte

In Abbildung 22 ist die Klasse Bibliografische-Aspekte zu sehen, welche die beiden Aspekte Dokumenten-Information und Literatur-Metadaten enthält.

Beim Aspekt Dokumenten-Information ergaben sich keine Änderungen, außer dass das Attribut „Stand“ nun eigenständig ist und nicht mehr auf ein Datum verweist. Der Aspekt Literatur-Metadaten enthält nun die Attribute der Datensammlungen Kapitel und Publikations ID.

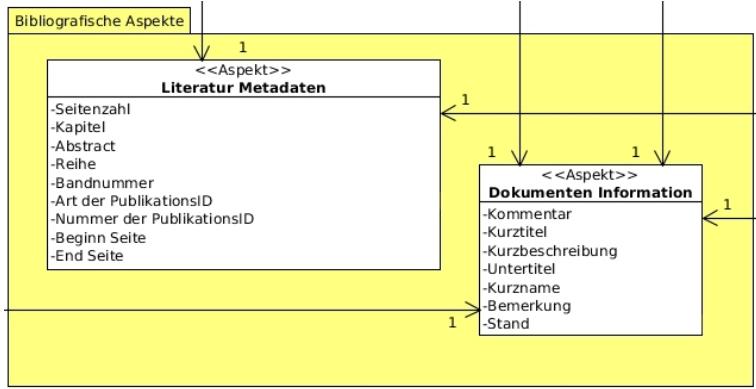


Abbildung 22: Die Klasse Standard Aspekte

8.4 Implementierung einer Datensammlung

Eine Datensammlung in Alfresco wird im Verzeichnis `ALFRESCO_HOME/tomcat/shared/classes/alfresco/extensions` abgelegt. Wie ein Datenmodell in Alfresco implementiert wird, ist anhand der Datensammlung **Bildarchiv Metadaten** gezeigt, welche im Listing 4 weiter unten zu sehen ist.

Mit dem Tag „model“ wird ein Datenmodell in Alfresco beschrieben, alle weiteren Einstellungen werden innerhalb dieses Tags vollzogen. Der Name des Modells wird durch das Prefix und die Namespace-URI gebildet.

Die Tags „description“, „author“ und „version“ geben grundlegende Informationen über das entsprechende Modell und wer dafür verantwortlich ist.

Innerhalb des Tags „imports“ werden Datenmodelle importiert, welche in dem Modell entsprechend verwendet werden. Grundsätzlich werden die Modelle <http://www.alfresco.org/model/dictionary/1.0> und <http://www.alfresco.org/model/content/1.0> immer importiert, da sie den Grundstock eines jeden Modells bilden.

Zusätzlich werden die beiden Modelle **Bibliografische-Aspekte** und **Standard-Metadaten** importiert (siehe Abschnitt 8.3.4 und 8.3.3), da ihre Aspekte innerhalb des Modells verwendet werden. Hierzu später mehr.

Der Tag „namespaces“ definiert den Namensraum, in welchem das Datenmodell verwendet wird. Mit „namespace“ wird der spezifische Namensraum zum einen durch eine URI, zum anderen durch ein ein Präfix festgelegt. Das Präfix ist eine Abkürzung, um nicht immer die komplette URI angeben zu müssen. Die URI wiederum gibt an, wo das Datenmodell abgelegt wird, beziehungsweise zu finden ist.

„constraints“ ist ein Tag, welcher mehrere Constraints beinhalten kann, wobei ein Constraint einen regulären Ausdruck (Regex) enthält und eine Zeichenfolge genauer beschreibt. Wird ein entsprechendes Constraint auf ein Textfeld angewendet, so wird beim Speichern des Dokuments von Alfresco überprüft, ob das jeweilige Attribut der Vorgabe des Constraints entspricht. Im Datenmodell **Bibliografische-Aspekte** gibt es keine Constraints, weshalb der entsprechende Tag leer ist.

Zusätzlich können mit Constraints auch Auswahllisten realisiert werden. Hierbei werden im Constraint alle möglichen Einträge vorgegeben, welche später zu Auswahl stehen sollen.

Innerhalb des Tags „types“ können mehrere Dokumenten-Typen spezifiziert werden. Jeder einzelne Typ ist grundsätzlich so aufgebaut wie der im Beispielcode in Listing 4 unten. Der Typ hat einen Namen, welcher mit dem Präfix des jeweiligen Namensraums beginnt.

Im Tag „title“ wird der Standardtitel angegeben, welcher angezeigt wird, sollte keine I18N-Übersetzung vorliegen. Die Übersetzung wird später noch genauer erklärt.

Sollte „parent“ keine eigene Klasse sein, so steht hier standardmäßig `cm:content`, welches die Standard-Oberklasse von Alfresco ist.

Unter dem Tag „properties“ sind alle Attribute zu finden, welche der jeweilige Typ enthält. Innerhalb einer „property“ wird wiederum der Name des Attributs angegeben. Im Tag „title“ findet sich der Standardtitel. „type“ gibt an, von welchem Typ das Attribut ist. Eine genaue Auflistung aller Datentypen von Alfresco ist in der Dokumentation¹¹ zu finden. Im Tag „constraints“ können Verweise zu einzelnen Constraints des Datenmodells oder aber zu Constraints anderer Datenmodelle angegeben werden.

Über den Tag „mandatory-aspects“ können Aspekte angegeben werden, welche der Dokumententyp zwingend beinhalten muss. Im Beispiel wären das die Aspekte **Inspire** und **Dokumenten-Information**. Beide Aspekte sind auch im Datenmodell in Abbildung 18 zu sehen.

„aspects“ kann mehrere Aspekte der jeweiligen Klasse enthalten. Diese Aspekte können dann innerhalb der Klasse, sowie klassenübergreifend eingesetzt werden. Grundsätzlich sind die Beschreibungen von Aspekten genau wie die von Typen aufgebaut und beinhalten die selben Tags. Einzige Ausnahme ist der Tag „parent“, da ein Aspekt keinen übergeordneten Datentyp beinhaltet. Im Beispiel sind keine Aspekte vorhanden, da das Datenmodell dies für die Klasse **Bildarchiv Metadaten** nicht vorsieht.

¹¹<http://docs.alfresco.com/4.0/concepts/metadata-model-props.html>

```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <model name="lupo-BM:Bildarchiv-Metadaten" xmlns="http://www.alfresco.org/model/
   dictionary/1.0">
4
5      <description>Class for a picture of the archiv</description>
6      <author>Sebastian Rieger</author>
7      <version>1.0</version>
8
9      <imports>
10         <import uri="http://www.alfresco.org/model/dictionary/1.0" prefix="d"/>
11         <import uri="http://www.alfresco.org/model/content/1.0" prefix="cm"/>
12         <import uri="Bibliografische-Aspekte" prefix="lupo-BA"/>
13         <import uri="Standard-Metadaten" prefix="lupo-SA"/>
14     </imports>
15
16     <namespaces>
17         <namespace uri="Bildarchiv-Metadaten" prefix="lupo-BM"/>
18     </namespaces>
19     <constraints>
20     </constraints>
21     <types>
22         <type name="lupo-BM:Bildarchiv-Metadaten">
23             <title>Bildarchiv-Metadaten</title>
24             <parent>cm:content</parent>
25             <properties>
26                 <property name="lupo-BM:Kurzname">
27                     <title>Kurzname</title>
28                     <type>d:text</type>
29                     <constraints>
30                     </constraints>
31                 </property>
32                 <property name="lupo-BM:Lateinischer Name">
33                     <title>Lateinischer Name</title>
34                     <type>d:text</type>
35                     <constraints>
36                     </constraints>
37                 </property>
38                 <property name="lupo-BM:Deutscher Name">
39                     <title>Deutscher Name</title>
40                     <type>d:text</type>
41                     <constraints>
42                     </constraints>
43                 </property>
44             </properties>
45             <mandatory-aspects>
46                 <aspect>lupo-SA:Inspire</aspect>
47                 <aspect>lupo-BA:Dokumenten-Information</aspect>
48             </mandatory-aspects>
49         </type>
50     </types>
51     <aspects>
52     </aspects>
53 </model>
```

Listing 4: Inhalt der Datei Bildarchiv-Metadaten.xml

8.4.1 Das Modell in Alfresco einbinden

Um das Datenmodell nun in Alfresco bekannt zu machen, muss eine Context-Datei erstellt werden, welche das Java-Bean beschreibt. Die Context-Datei endet auf „-context.xml“ und muss im Ordner **ALFRESCO_HOME/tomcat/shared/classes/alfresco/extensions** abgelegt werden.

Im Listing 5 ist die Context-Datei zu sehen, welche beschreibt wo genau die Klasse zu finden ist. Das „bean“ hat zum einen eine **id** und ein **parent**, zum anderen ein Attribut, auf welchen Beans es basiert, was im Attribut **depends-on** beschrieben ist. Im Beispiel sind das die beiden verwendeten Klassen welche auch importiert werden (siehe Listing 4).

Unter dem Tag „property“ wird der Speicherort unserer Modell-Datei angeben.

```
1 <?xml version='1.0' encoding='UTF-8'?>
2 <!DOCTYPE beans PUBLIC '-//SPRING//DTD BEAN//EN' 'http://www.springframework.org
   /dtd/spring-beans.dtd'>
3
4 <beans>
5
6   <bean id="Bildarchiv-Metadaten.dictionaryBootstrap" parent="
      dictionaryModelBootstrap" depends-on="Bibliografische-Aspekte.
      dictionaryBootstrap Standard-Metadaten.dictionaryBootstrap">
7     <property name="models">
8       <list>
9         <value>alfresco/extension/Bildarchiv-Metadaten.xml</value>
10        </list>
11      </property>
12    </bean>
13
14 </beans>
```

Listing 5: Inhalt der Datei Bildarchiv-Metadaten-context.xml

Sind beide Dateien korrekt und an der richtigen Stelle muss Alfresco neu gestartet oder die Web-Skripte müssen neu geladen werden um das Modell aufzunehmen. Um die Web-Skripte von Alfresco neu zu laden, muss der Button „Refresh Web Scripts“ unter <http://localhost:8080/share/service/index> betätigt werden.

8.4.2 Übersetzungen in Alfresco

Um Mehrsprachigkeit in Alfresco zu realisieren verwendet das System Property-Dateien, in welchen die verschiedenen Übersetzungen zu finden sind. Die Property-Dateien müssen unter **ALFRESCO_HOME/tomcat/shared/classes/alfresco/messages** abgelegt werden.

Die Dateien können beliebig benannt werden, müssen jedoch das ISO-Sprachkürzel am Namensende enthalten. Eine deutsche Übersetzung würde dann zum Beispiel in der Datei **I18N_de_DE.properties** und eine US-Amerikanische unter **I18N_en_US.properties** zu finden sein. Alfresco nutzt die Übersetzungen dann automatisch anhand der eingestellten Systemsprache.

Die Datei beinhalten pro Zeile ein Schlüssel-Wert-Paar, welches eine Übersetzung für genau einen String darstellt. Im Listing 6 ist nur ein Ausschnitt der Datei **I18N_de_DE.properties** zu sehen, um das System zu verdeutlichen.

In den Property-Dateien werden Strings für Typen mit einem Schlüssel **type.<Namespace>_<Typname>** bezeichnet. Aspekte werden ähnlich wie auch Typen bezeichnet

und zwar mit aspect.<Namespace>_<Aspektname>. Andere Keys, wie zum Beispiel die Keys für Hilfetexte, können frei gewählt werden.

```
1 ######
2 #FADO I18N Uebersetzung DEFAULT#
3 #####
4 #Uebersetzungen fuer die Datei Fado-Metadaten.xml
5 #
6 ##TYPEN
7 type.lupo-FADO_Fado-Metadaten=FADO-Obertyp
8 #
9 #Uebersetzungen fuer die Datei Fachsystem-Metadaten.xml
10 #
11 ##TYPEN
12 type.lupo-FM_Forschungsvorhaben=Forschungsvorhaben
13 type.lupo-FM_FADO-Urteil=Urteil
14 type.lupo-FM_Bibliografische-Angaben=Bericht
15 #
16 #Uebersetzungen fuer die Datei Standard-Metadaten.xml
17 #
18 ##Aspekte
19 aspect.lupo-SA_Inspire=Inspire
```

Listing 6: Ausschnitt des Inhalts der Übersetzungsdatei I18N_de_DE.properties

8.5 Modell und Übersetzungen in Alfresco anzeigen

Nach dem in den vorherigen Abschnitten gezeigt wurde, wie eine Datenmodell für Alfresco erstellt und eingebunden wird, muss nun noch geklärt werden, wie sich Übersetzungen und Datenmodell in Alfresco anzeigen lassen.

8.5.1 Das Datenmodell anzeigen

Um ein Datenmodell in Alfresco mit all seinen Metadaten in der Präsentation und in den Formularen anzusehen, reicht es nicht aus, nur das Bean zu erstellen. Zusätzlich muss auch angegeben werden, welche Attribute Alfresco wie darstellen soll. Die Darstellung wird in der Datei `share-config-custom.xml` beschrieben, welche sich im Verzeichnis `ALFRESCO_HOME/tomcat/shared/classes/alfresco/web-extensions` befindet.

Da diese Config-Datei sehr lang werden kann, werden im Folgenden immer nur Ausschnitte gezeigt.

Innerhalb des Tags „alfresco-config“ ist der im Listing 7 gezeigte Abschnitt zu finden, welcher jedoch hier gekürzt dargestellt ist. Innerhalb des Tags „field-visibility“ müssen alle Attribute aufgezählt werden die später angezeigt werden sollen. Dies geschieht über den Tag „show“ in welchem die ID des jeweiligen Attributs angegeben werden muss.

Die einzelnen Attribute werden, wenn sie nicht anders gegliedert sind in der Reihenfolge dargestellt wie sich hier eingetragen wurden.

Unter Tag „field-visibility“ ist der Tag „appearance“ zu sehen. Dieser gibt an, wie die einzelnen Attribute angezeigt werden. Als erstes werden mit dem „set“-Tag gruppen erzeugt, welche im Beispiel als „bordered-panel“ angezeigt werden sollen.

Als nächstes werden die einzelnen Attribute mit dem Tag „field“ noch einmal aufgelistet. Hier wird nun aufgeführt, zu welcher Gruppe das Attribut gehört und welchen Hilfetext es haben

soll. Natürlich gibt es noch weitere Einstellungsmöglichkeiten¹², welche im Beispiel jedoch nicht verwendet werden.

¹²https://wiki.alfresco.com/wiki/Forms_Examples#Changing_Set_Appearance

```

1 <config evaluator="node-type" condition="lupo-FADO:Fado-Metadaten">
2   <forms>
3     <form>
4       <field-visibility>
5         <show id="cm:name" />
6         <show id="cm:title" force="true" />
7         <show id="cm:description" force="true" />
8
9         <!-- lupo-FADO properties -->
10        <show id="lupo-FADO:Fachsystem" />
11        <show id="lupo-FADO:Unsichtbar" />
12        <show id="lupo-FADO:Ausblenden" />
13
14        <!-- cm:dublincore aspect -->
15        <show id="cm:publisher"/>
16        <show id="cm:contributor"/>
17        <show id="cm:type"/>
18        <show id="cm:identifier"/>
19        <show id="cm:dcsource"/>
20        <show id="cm:coverage"/>
21        <show id="cm:rights"/>
22        <show id="cm:subject"/>
23
24        <!-- Weitere Attribute stehen hier -->
25      </field-visibility>
26      <appearance>
27        <set id="cm:content" appearance="bordered-panel" label="Content" />
28        <set id="lupo-FADO" appearance="bordered-panel" label="Fado Metadaten" />
29        <set id="cm:dublincore" appearance="bordered-panel" label="Dublin Core" />
30        <!-- Weitere Beschreibungen stehen hier -->
31
32        <field id="cm:name" set="cm:content" />
33        <field id="cm:title" set="cm:content" />
34        <field id="cm:description" set="cm:content" />
35
36        <!-- lupo-FADO properties -->
37        <field id="lupo-FADO:Fachsystem" set="lupo-FADO" description-id="FachsystemDESC" />
38        <field id="lupo-FADO:Unsichtbar" set="lupo-FADO" description-id="UnsichtbarDESC" />
39        <field id="lupo-FADO:Ausblenden" set="lupo-FADO" description-id="AusblendenDESC" />
40
41        <!-- cm:dublincore aspect -->
42        <field id="cm:publisher" set="cm:dublincore" />
43        <field id="cm:contributor" set="cm:dublincore" />
44        <field id="cm:type" set="cm:dublincore" />
45        <field id="cm:identifier" set="cm:dublincore" />
46        <field id="cm:dcsource" set="cm:dublincore" />
47        <field id="cm:coverage" set="cm:dublincore" />
48        <field id="cm:rights" set="cm:dublincore" />
49        <field id="cm:subject" set="cm:dublincore" />
50
51        <!-- Weitere Beschreibungen stehen hier -->
52      </appearance>
53    </form>
54  </forms>
55</config>
```

Listing 7: Beschreibung einer Klasse des Datemodells in der `share-config-custom.xml`

Wie sich das setzen der Gruppierung in Alfresco auswirkt und wie es in der Dokumentenübersicht genau aussieht, ist im Anhang 14.5 zu finden.

8.5.2 Aspekte anzeigen

Um die im Datenmodell erstellten Aspekte anzuzeigen, müssen diese ebenfalls in der `share-config-custom.xml` angegeben werden. Dies geschieht im schon vorhandenen Tag `<config evaluator="string-compare" condition="DocumentLibrary" replace="true">` wie im Listing 8 zu sehen ist. Innerhalb des Tags „visible“ müssen alle zuzuzeigenden Aspekte des Datamodells angegeben werden.

Ist diese Einstellung richtig vorgenommen wurden, können die eigenen Aspekte nun auch zum Datentyp hinzugefügt werden.

Zu beachten ist jedoch, dass nur Attribte von Aspekten angezeigt werden, die auch wie im Abschnitt 8.5.1 gezeigt aufgelistet sind.

```
1 <!-- Document Library config section -->
2   <config evaluator="string-compare" condition="DocumentLibrary" replace="true"
3     >
4
5     <!-- Weitere Einstellungen -->
6
7     <aspects>
8       <!-- Aspects that a user can see -->
9       <visible>
10
11         <!-- Weitere Aspekte -->
12
13         <aspect name="lupo-SA:Inspire" />
14         <aspect name="lupo-GB:Gericht" />
15         <aspect name="lupo-GB:Urteil" />
16         <aspect name="lupo-GB:Gesetz" />
17         <aspect name="lupo-BA:Literatur-Metadaten" />
18         <aspect name="lupo-BA:Dokumenten-Information" />
19
20       </visible>
21
22     <!-- Weitere Einstellungen -->
23
24   </config>
```

Listing 8: Beschreibung der Aspekte in der `share-config-custom.xml`

8.5.3 Datentyp umwandeln

Damit die eigenen Datentypen verwendet werden können, müssen die Dokumente erst in diese Typen umgewandelt werden, denn standardmäßig sind alle Dokumente vom Typ „cm:content“.

Damit die Typen in der Oberfläche von Alfresco geändert werden können, muss dies auch in der `share-config-custom.xml` beschrieben werden. Hierfür muss wie im Listing 9 ein neuer „config“-Tag angelegt werden. Im Tag „types“ wiederum werden die einzelnen Typen definiert, von welchen aus in andere umgewandelt werden soll. So kann im Beispiel der Typ „cm:content“ in vier andere Typen umgewandelt werden. Dies ist möglich, da diese vier Typen von „cm:content“ erben.

Umgekehrt, ist eine Rück-Umwandlung nicht möglich, da hier möglicherweise Daten verloren gehen können.

Die unteren drei Datentypen erben von „lupo-FADO:Fado-Metadaten“ und können daher nicht direkt aus „cm:content“ erstellt werden.

```

1 <config evaluator="string-compare" condition="DocumentLibrary">
2   <types>
3     <type name="cm:content">
4       <subtype name="lupo-FADO:Fado-Metadaten" />
5       <subtype name="lupo-DRS:DRS-Metadaten" />
6       <subtype name="lupo-BM:Bildarchiv-Metadaten" />
7       <subtype name="lupo-ICT:Artikel-Metadaten" />
8     </type>
9     <type name="lupo-FADO:Fado-Metadaten">
10      <subtype name="lupo-FM:Forschungsvorhaben" />
11      <subtype name="lupo-FM:FADO-Urteil" />
12      <subtype name="lupo-FM:Bibliografische-Angaben" />
13    </type>
14  </types>
15</config>
```

Listing 9: Beschreibung der Typumwandlung in der `share-config-custom.xml`

8.5.4 Übersetzungen einbinden

Alfresco können mehrere Dateien mit Übersetzungen bereitgestellt werden, welche davon verwendet wird entscheidet Alfresco automatisch Anhand der eingestellten Systemsprache. Sollte für die eingestellte Sprache keine spezielle Übersetzung vorliegen, so wird die Standard-Properties-Datei verwendet, welche keine Sprachkürzel enthält, in diesem Fall wäre das die Datei `I18N.properties`.

Es muss angegeben werden, wo sich die Sprachdateien befinden. Hierfür muss ein Eintrag in der `custom-slingshot-application-context.xml` welche sich im Verzeichnis `ALFRESCO_HOME/tomcat/shared/classes/alfresco/web-extensions` befindet angelegt werden.

Innerhalb des Tags „list“, muss ein neuer „value“-Tag erstellt werden, wie es im Listing 10 zu sehen ist, welcher den Pfad zur Standardübersetzungsdatei angibt.

```

1 <beans xmlns="http://www.springframework.org/schema/beans"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns:hz="http://www.hazelcast.com/schema/spring"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans
5     http://www.springframework.org/schema/beans/spring-beans-2.5.
6     http://www.hazelcast.com/schema/
7     http://www.hazelcast.com/schema/spring/hazelcast-spring-2.4.xsd"
8   >
9
9   <bean id="webscripts.resources" class="org.springframework.extensions.surf.
10    util.ResourceBundleBootstrapComponent">
11     <property name="resourceBundles">
12       <list>
13         <value>webscripts.messages.webscripts</value>
14         <value>alfresco.messages.common</value>
15         <value>alfresco.messages.slingshot</value>
16         <value>alfresco.messages.I18N</value>
17     </list>
18   </property>
```

Listing 10: Beschreibung der `custom-slingshot-application-context.xml`

8.6 Metadatenmodell Editor

Eine der Anforderungen an FADO ist, dass das Metadatenmodell einfach und möglichst ohne Programmierkenntnisse verändert werden kann. Leider konnte nach einiger Recherche kein Editor gefunden werden der unter der aktuellen Version 5.0 lauffähig ist.

Aus diesem Grund sollte bei der Weiterführung des Projekts darüber nachgedacht werden eine eigene Lösung als eigenständiges Programm, oder als Alfresco-Plugin zu implementieren.

8.7 Bulk-Import von Dateien

Alfresco bietet die Möglichkeit Dateien und ihre Metadaten auch automatisiert einzulesen, dies ist vor allem hilfreich, wenn ein schon bestehendes System auf Alfresco migriert werden soll.

Der Bulk-Import geschieht über `http://localhost:8080/alfresco/service/bulkfsimport`. Hier muss der Ordner angegeben werden, von dem die Daten eingelesen werden sollen. Zusätzlich muss ein Ordner innerhalb von Alfresco angegeben werden, in dem die eingelesenen Daten gespeichert werden soll.

Es lassen sich zusätzlich zum Beispiel auch Stapelgröße und eine Threadanzahl einstellen, mit denen importiert wird.

Um nun auch Metadaten automatisiert einzulesen und zu speichern, müssen diese in XML-Dateien abgelegt sein. Die beschreibenden XML-Dateien müssen folgenden Namen tragen: `<Name der Realdatei>.<Dateiendung der Realdatei>.metadata.properties.xml`

Wird nun ein Bulk-Import gestartet, so werden aus den beschreibenden XML-Dateien die Attributwerte abgelesen.

Im Anhang 14.6 ist eine XML-Datei zu sehen, welche es ermöglicht, Urteile in Alfresco zu integrieren. Es wurde bewusst auf den Inhalt, zugunsten der Lesbarkeit, verzichtet.

8.8 Verwendung von langen Textfeldern

Bei der Implementierung des Datenmodells wurde festgestellt, dass ein Textfeld in Alfresco standardmäßig 1.024 Zeichen lang sein kann. Problematisch wird dies vor allem bei FADO-Dokumenten des Typs **Urteil**, denn hier können vor allem die beiden Attribute **Tatbestand** und **Gründe** die Länge sehr leicht überschreiten. Sogar Längen mit über 27.000 Zeichen sind im aktuellen Datenbestand der LUBW möglich. Aus diesem genannten Grund, wurde hier etwas mehr Entwicklungsaufwand getätig.

Zum einen sollte ein so langer Text nicht als einzeiliges Textfeld dargestellt werden, zum anderen muss es möglich gemacht werden, Texte einzugeben, welche die Länge von 1.024 Zeichen überschreiten. Daher wurde die `share-config-custom.xml` noch einmal erweitert.

Das entsprechende Feld (im Beispiel das Attribute **Gründe**) muss in seiner „appearance“ angepasst werden, wie es im Listing 11 zu sehen ist. Um ein mehrzeiliges Textfeld zu erzeugen, muss das „Control-Template“ `textarea.ftl` verwendet werden. Das Template muss wie angegeben mit

einer vollen Pfadangabe versehen werden, denn an dieser Stelle könnten auch eigene Templates eingebunden werden.

Über den Tag „control-param“ wird mit `maxLength` die maximale Länge des Feldes angegeben. Im Beispiel sind es 40.000 Zeichen, was für die aktuellen FADO-Urteile ausreicht. Wird keine maximale Länge angegeben, so wird das Feld standardmäßig auf 1.024 Zeichen begrenzt.

Durch die Verwendung der „Textarea“ ist es einem Nutzer möglich, die Größe des Feldes auf seine Bedürfnisse zu skallieren, was er mittels der Schaltfläche am unteren rechten Textfeldrand machen kann. [Alf15b]

Es wäre auch möglich, die Zeilenlänge fest vorzugeben, was jedoch als unpraktikabel empfunden wurde.

```
1 ...
2 <appearance>
3   ...
4     <field id="lupo-GB:Gründe" set="lupo-GB:Urteil" description-id="">
5       <control template="/org/alfresco/components/form/controls/textarea.ftl">
6         ...
7         <control-param name="maxLength">40000</control-param>
8       </control>
9     </field>
10    ...
11  </appearance>
12 ...
```

Listing 11: Appearance Anpassungen in der `share-config-custom.xml`

In Abbildung 23 ist das Textfeld nach der Bearbeitung der `share-config-custom.xml` zu sehen. Zu beobachten ist die lange Scrollleiste am rechten Rand, welche auf einen größeren Inhalt hindeutet.

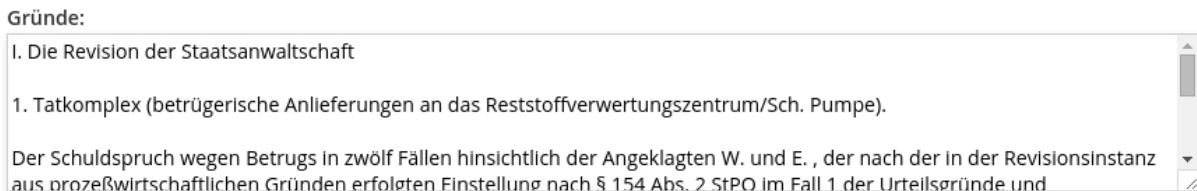


Abbildung 23: Das Textfeld Gründe nach der Bearbeitung

8.9 Erfahrungen aus der Alfresco-Konfiguration

Im Internet finden sich viele Hilfestellungen und Tutorials zu Alfresco, leider nicht zu allen relevanten Themen der Arbeit mit Alfresco. Deshalb sollen einige Fehlerquellen und Probleme aufgezeigt werden und wie diese gelöst wurden.

8.9.1 Aufspaltung der `share-config-custom.xml`

Da alle Klassen des Content-Modells mit ihren anzuseigenden Attributen, in der `share-config-custom.xml` angegeben werden müssen, wächst die Datei schnell an und wurde entsprechend unübersichtlich.

Daher wurde die Datei aufgespaltet und die einzelnen Bestandteile in separaten Dateien untergebracht. Hierzu müssen die neuen Dateien als Bean angegeben werden, damit Alfresco diese berücksichtigt.

Im Listing 12 ist ein Ausschnitt der Datei `slingshot-application-context.xml` zu finden, welche sich im Verzeichnis `ALFRESCO_HOME/tomcat/webapps/share/WEB-INF/classes/alfresco` befindet.

In ihr werden die einzelnen Teile der `share-config-custom.xml` mit ihren Speicherorten angegeben. Im Beispiel wurden alle Beschreibungen der Modellklassen jeweils in eigene Dateien ausgelagert.

```

1 <bean id="webframework.configsource" class="org.springframework.extensions.
2   config.source.UrlConfigSource">
3     <constructor-arg>
4       <list>
5
6         <!-- Hier sind weitere Einstellungen vorhanden -->
7
8         <!-- Share custom config -->
9         <value>classpath:alfresco/web-extension/share-config-custom.xml</
10           value>
11        <value>jar:!*!/META-INF/share-config-custom.xml</value>
12        <value>classpath:alfresco/web-extension/share-config-custom-dev.xml
13           </value>
14        <value>jar:!*!/META-INF/share-config-custom-dev.xml</value>
15
16        <!-- Share custom config files, which contains the model -->
17        <value>classpath:alfresco/web-extension/share-config-model-Fado-
18          Metadaten.xml</value>
19        <value>classpath:alfresco/web-extension/share-config-model-
20          Forschungsvorhaben.xml</value>
21        <value>classpath:alfresco/web-extension/share-config-model-Urteil.
22          xml</value>
23        <value>classpath:alfresco/web-extension/share-config-model-
24          Bibliografische-Angaben.xml</value>
25        <value>classpath:alfresco/web-extension/share-config-model-DRS.xml</
26          value>
27        <value>classpath:alfresco/web-extension/share-config-model-
28          Bildarchiv.xml</value>
29        <value>classpath:alfresco/web-extension/share-config-model-ICT.xml</
30          value>
31      </list>
32    </constructor-arg>
33  </bean>
```

Listing 12: Bean zur Aufteilung der `custom-slingshot-application-context.xml`

8.9.2 Reihenfolge der Java-Beans beachten

Basieren die einzelnen Modellklassen aufeinander, kann es zu Fehlern kommen, wenn die einzelnen Java-Beans nicht in der richtigen Reihenfolge geladen werden.

Importiert eine Modellklasse eine andere, so muss das in der Bean-Beschreibung auch als Abhängigkeit angegeben werden. Dies wurde im Abschnitt 8.4.1 und speziell im Listing 5 schon einmal erläutert.

8.10 Fehlende Funktionalität im Backend

Das Lastenheft im Kapitel 3 forderte eigentlich, Autoren wie Entitäten abzubilden, um zum Beispiel alle Werke eines Autors zu finden, ohne dabei jedes Dokument abfragen zu müssen. Leider ist dieses Vorgehen unter Alfresco nicht möglich.

Alfresco ist ein ECM-System, und verlangt für das Anlegen von Metadaten oder eines Datensatzes immer explizit ein physisches Dokument, welches entweder hochgeladenen oder direkt mit Alfresco erstellt werden muss. Es ist dagegen nicht möglich eigenständige Datensätze zum Beispiel Autoren ohne ein solches Dokument anzulegen.

Dieses Problem äußert sich auch, da ein FADO-Urteil keine physische Datei besitzt. Ein Urteil ist im alten System einfach als Datenbankeintrag abgebildet, ohne dass jemals eine physische Datei besteht. Bisher wurde dies im neuen System damit gelöst, dass einfach eine leere Textdatei angelegt wurde, welcher dann die entsprechenden Metadaten zugeordnet werden können. Ein Änderung dieses Mechanismus würde einen tieferen Eingriff in Alfresco notwendig machen, der im Rahmen dieser Arbeit jedoch nicht angegangen wird.

Verweise auf andere Fachsysteme können durch Verweise auf Ordner (welche alle Dateien zu einem System beinhalten) nachgebildet werden.

9 Implementierung des Frontends auf Basis von Liferay

Liferay ist ein hochperformanter Portalserver, welcher wie auch Alfresco Open Source ist. Der Nutzer hat bei Liferay die Auswahl zwischen einer kostenlosen Community Edition und einer kostenpflichtigen Enterprise Edition. Vorteil der kostenpflichtigen Enterprise Edition ist, dass hier ein Support zur Verfügung steht. Außerdem werden die kostenpflichtigen Versionen vor einer Veröffentlichung genauer getestet, als die Community Edition. [Wen13] [Wik15f]

Liferay nutzt für die Darstellung von Content sogenannte „Portlets“. Diese Portlets sind Anwendungen, welche Nutzern im Web zur Verfügung gestellt werden. Dabei können mehrere Portlets zu dynamischen (Web-)Seiten zusammengestellt werden. Jedes Portlet besteht daher aus einer clientseitigen Anzeige, kann jedoch auch auf Serverseite Anwendungslogik enthalten, die zum Beispiel den Zugriff auf Daten regelt.

9.1 Alfresco als IFrame im Liferay

In Liferay ist es möglich, einen Webcontent, wie das Alfresco-Portal einer ist, als IFrame darzusellen. Der Vorteil einer solchen Umsetzung ist, dass sie schnell und einfach über ein Web-Content-Portlet umgesetzt werden kann. Ein großer Nachteil eines solchen Iframes ist, dass es sich nur um ineinander geschachtelte Ansichten von getrennten Systemen handelt. Darstellung, Daten und Logik bleiben - wie auch die Benutzerverwaltung - komplett getrennt. Eine einheitliche Anwendung lässt sich auf diese Weise nicht gestalten. Daher kann die Einbindung per IFrame bestenfalls als Notlösung dienen.

9.2 Liferay-Repository als Frontend für Alfresco Dateien

Auch Liferay selbst bietet grundlegend die Möglichkeit, eines Asset- und Dokumentenmanagementsystems. Zusätzlich können auch Dateien aus anderen Systemen als Repository zu Liferay hinzugefügt werden. Sie werden dann so behandelt, als seien sie Liferay-eigene Objekte. Tatsächlich werden „echte“, redundante Liferay-Objekte erstellt, die automatisch mit der Quelle synchronisiert werden. Der Vorteil dieses Ansatzes ist, dass über diese Repository Objekte aus fremden Datenquellen über dieselben Mechanismen behandelt werden können wie die Liferay-eigenen. So können diese Dateien, zumindest konzeptionell, zum Beispiel über Standard-Liferay-Portlets gefiltert, sortiert und angezeigt werden.

Das Hinzufügen eines Repositorys ist im Grunde ganz einfach, vorausgesetzt die Einstellungen der Benutzerverwaltung sind entsprechend gesetzt, denn um auf ein Repository von Alfresco zugreifen zu können, muss sich der Benutzer authentifizieren. (siehe Abschnitt 9.2.1)

9.2.1 Benutzerverwaltung für eine Repositoryzugriff konfigurieren

Das im Abschnitt 9.2.2 verwendete CMIS-Repository funktioniert nur, wenn zuvor eine Authentifizierung des Nutzers stattgefunden hat. Normalerweise wird so eine Benutzerverwaltung von einem LDAP-Server oder über SSO vorgenommen.

Es reicht für die prototypische Implementierung in dieser Arbeit aber vollkommen aus, den gleichen Benutzer in beiden Systemen zu erstellen. Hierbei müssen die Passwörter und die Benutzernamen in beiden Systemen übereinstimmen.

Ist der selbe Nutzer in beiden Systemen erstellt, muss in die Liferay-Datei `portal-setup-wizard.properties` welche im Order `LIFERAY_HOME` liegt, noch die im Listing 13 zu sehende Zeile hinzugefügt werden. Dies erlaubt es Liferay, das Passwort innerhalb einer Sitzung zu speichern und es für die CMIS-Verbindung zu nutzen. [CMI]

1 `session.store.password=true`

Listing 13: Erweiterung in der Datei portal-setup-wizard.properties

Zusätzlich muss in Alfresco unter `Admin → Kontrollbereich → Portaleinstellungen → Authentifizierung` im Feld `Wie authentifizieren sich Nutzer?` „Mit Benutzername“ angegeben werden.

Nach einem Neustart des Servers muss sich ab sofort mit dem Benutzernamen angemeldet werden und nicht wie zuvor mit der Mail-Adresse. [CMI15]

9.2.2 Alfresco als Repository im Liferay

In Liferay kann auf der Seite „Inhalte“ unter „Dokumente und Medien“ ein neues Repository erzeugt werden. Es muss ein Name für das Repository gewählt werden, unter dem dieses später zu finden sein soll. Als Repositorytyp muss ein CMIS-Repository des Typs „AtomPub“ gewählt werden.

Die passende URL von Alfresco kann auf der Seite `http://127.0.0.1:8080/alfresco/` gefunden werden. Es können die AtomPub Version 1.0 oder 1.1 gewählt werden. Zusätzlich finden sich hier auch weitere Links, wie zum Beispiel auf die WebDav-Schnittstelle von Alfresco.

Ist die richtige URL im Feld „AtomPub-URL“ eingetragen, muss noch die Berechtigung gewählt und anschließend gespeichert werden. Die Felder „Beschreibung“ und „Repository ID“ können frei bleiben.

Ist alles richtig konfiguriert, sind nun alle Dateien aus Alfresco auch in Liferay abrufbar (siehe Abbildung 29 im Anhang 14.7) und können innerhalb von Liferay verwendet und dargestellt werden. [CMI]

Jedoch stellt Liferay nur Metadaten dar, welche in der Datei selbst vorhanden sind. Alle anderen Metadaten, welche in Alfresco vorhanden sind, werden ignoriert. Im folgenden Abschnitt 9.2.3 wird genauer untersucht, warum die Alfresco-Metadaten nicht dargestellt werden. [Aga13]

9.2.3 Untersuchungen zum Liferay-Repository

Zunächst muss überprüft werden, ob Alfresco überhaupt alle vorhandenen Metadaten über die CMIS-Schnittstelle zur Verfügung stellt.

Über den Link im Listing 14 können alle Informationen zu einem Dokument (Node) anhand seiner ID abgefragt werden. [Get15]

1 `http://localhost:8080/alfresco/api/-default-/public/cmis/versions/1.1/atom/id?id=67cbcd24-a038-4141-a0f9-d884e791724a`

Listing 14: Abfrage aller Propertys eines Nodes mittels CMIS 1.1

Beim Aufrufen des Links muss zuerst eine Authentifizierung für Alfresco angeben werden. War diese korrekt, so wird im Browser das entsprechende Element mit allen Metadaten in XML

dargestellt. Dies ist zum einen die Bestätigung, dass der CMIS-Dienst von Alfresco ohne Probleme läuft, aber zum anderen auch, dass Alfresco alle vorhandenen Metadaten übermittelt.

Das Problem liegt also nicht auf der Seite von Alfresco und es muss untersucht werden, ob die Daten im Liferay-Server richtig ankommen und verarbeitet werden.

Um zu verifizieren, ob Liferay die Metadaten von Alfresco gelesen hat, muss in die Datenbank von Liferay geschaut werden.

Liferay nutzt zur internen Datenspeicherung standardmäßig eine HyperSQL-Datenbank, welche vollständig in Java implementiert ist und unter der BSD-Lizenz steht. Um die Datenbank auszulesen, wird ein Tool benötigt, welches jedoch mit dem Liferay-Paket mitgeliefert wird. Die jar-Datei befindet sich unter `LIFERAY_HOME/tomcat-7.0.42/lib/ext/hsqldb.jar`.

Nach dem Start muss als URL `jdbc:hsqldb:<Pfad zum Liferay-Home>/LIFERAY_HOME/data/hsqldb/1portal` angegeben werden um sich mit der Datenbank zu verbinden. Vorher muss der Liferay-Server heruntergefahren werden, da dieser ein Lock auf die Datenbank hält und sie somit sperrt.

Liferay kann aber auch mit einer anderen Datenbank, wie zum Beispiel MySQL, eingerichtet werden. Voraussetzung hierfür ist, dass der Datendank-Server dann auf dem System vorhanden ist.

Innerhalb der Datenbank sind in der Tabelle `PUBLIC.DDMCONTENT` auch die Metadaten der Dateien abgelegt, welche über das Repository eingefügt wurden. In der Spalte `XML` werden alle Metadaten zu einem Dokument abgelegt. Hier fällt auf, dass nicht alle Metadaten, welche Alfresco bietet, angegeben sind. Lediglich standardisierte Metadaten, welche aus der Datei ausgelesen wurden, sind hier zu finden.

Daher ist es nicht verwunderlich, dass Liferay keine weiteren Metadaten anzeigt, denn sie sind einfach nicht bekannt. Offensichtlich werden die Alfresco-Metadaten über die CMIS-Schnittstelle von Liferay nicht vollständig übernommen und auf entsprechende Liferay-Metadatenfelder abgebildet.

Im Folgenden muss nun geprüft werden, wie die Metadaten in Liferay bekannt gemacht werden können. Hierbei müssen die Metadaten aus Alfresco ausgelesen, in Liferay gespeichert und angezeigt werden.

Nach einiger recherche stellte sich heraus, dass Liferay nicht in der Lage ist eigene Metadatamodelle aus Alfresco zu verarbeiten, da diese nicht standardisiert sind. Im Abschnitt 9.3.1 wird hierauf noch einmal eingegangen. [Cho12]

9.3 Alternativen zum bestehenden Liferay-Repository

In den vorherigen Kapiteln wurde beschrieben, wie es möglich ist, Alfresco-Daten in Liferay zu integrieren. Es wurde jedoch festgestellt, dass Liferay standardmäßig nicht in der Lage ist, die Metadaten von Alfresco-Dokumenten vollständig zu verarbeiten oder anzuzeigen. Daher wird im Folgenden nun auf Alternativen eingegangen.

Grundsätzlich gibt es zwei mögliche Alternativen zum Liferay-CMIS-Repository. Zum einen besteht die Möglichkeit ein eigenes, unabhängiges Portlet, einen Hook oder eine andere Technologie für Liferay zu entwickeln, zum anderen ist es möglich, in den bestehenden Quellcode von Liferay einzugreifen. Die Möglichkeit eines Eingriffs in den Quellcode ist grundsätzlich möglich, da es sich, wie schon erwähnt, bei Liferay um ein Open-Source-Projekt handelt. Es wäre somit

möglich, die CMIS Schnittstelle von Liferay zu erweitern, so dass sie die von Alfresco gegebenen Metadaten verwalten kann. Natürlich gibt es noch andere Möglichkeiten, welche in den Folgenden Abschnitten beschrieben werden.

9.3.1 Eingriff in den Liferay Quellcode

Im Liferay Quellcode¹³ konnten mehrere Stellen identifiziert werden, wo die Apache-Chemistry Bibliothek, welche den Open-CMIS-Standard implementiert, verwendet wird. Nähere Infos hierzu sind im nächsten Abschnitt zu finden.

Die Verwendung dieser Bibliothek deutet eindeutig auf die Implementierung von CMIS hin. Im Quellcode konnten somit mehrere Klassen gefunden werden, welche für die Verbindung mit Repositorys zuständig sind. So auch die Klasse `CMISRepository.java`¹⁴ welche die Abfrage von Dokumenten übernimmt. An dieser Stelle müsste in den Quellcode eingegriffen werden, damit Metadaten von Alfresco nicht nur geladen, sondern auch in Liferay gespeichert werden können.

Damit jedoch nicht genug, es muss auch geschaut werden wie die Speicherung und die Anzeige der gelandeten Metadaten umgesetzt werden kann. Hierfür war jedoch im Rahmen der Arbeit nicht genügend Zeit, weshalb dieser Ansatz nicht weiter verfolgt wurde.

9.3.2 Entwicklung eines neuen Liferay-Hooks

Eine Möglichkeit, die fehlenden Metadaten aus Alfresco zu laden, wäre über einen Hook. Dieser sogenannte Hook ist ein Hintergrundprozess, der ohne grafische Schnittstelle auskommt. [Sez12]

Um überhaupt eine Erweiterung für Liferay schreiben zu können, wird die Liferay IDE¹⁵ benötigt. Hierbei handelt es sich um eine modifizierte Eclipse-Plattform, welche einen Nutzer in die Lage versetzt, Komponenten für Liferay zu entwickeln.

Die Klasse des Hooks implementiert das Interface `ModelListener`, welcher auf Änderungen der Klasse `RepositoryEntry` achtet. In der implementierten Methode `onAfterCreate()` werden dann die Metadaten von Alfresco über die CMIS-Schnittstelle abgeholt und in die Liferay-Datenbank gespeichert. Die Methode wird aufgerufen, sobald ein Element aus dem Repository gelesen und in der Liferay-Datenbank angelegt wird. Dies geschieht zum Zeitpunkt der ersten Nutzung. Die Synchronisation von Daten muss später der Hook selbst übernehmen, denn automatisch werden nur neue oder geänderte Daten geladen.

Für das Abrufen der Alfresco-CMIS-Schnittstelle wurde die Apache Bibliothek „Chemistry“¹⁶ verwendet, welche es ermöglicht, CMIS-Schnittstellen anzusprechen.

Die genaue Implementierung ist im Listing 15 zu sehen. Es wird zuerst eine Session mit den benötigten Parametern wie Username, Password und AtomPub-URL erzeugt. Dies geschieht in der selbst erstellten Methode `createCmisSession()`, welche im Anhang 14.8 zu sehen ist.

Anschließend wird für das gerade erstellte Element die ID geholt und mit ihrer Hilfe das `CmisObjekt` erstellt. Handelt es sich um einen Ordner, wird nichts weiter unternommen. Ist das Element jedoch ein Dokument, so werden alle Propertys geholt.

¹³<https://github.com/liferay/liferay-portal>

¹⁴<https://github.com/liferay/liferay-portal/blob/a3529628fb7bab5f22d4885ea32cce50703b995f/modules/apps/document-library/document-library-repository-cmis-impl/src/com/liferay/document/library/repository/cmis/internal/CMISRepository.java>

¹⁵<http://sourceforge.net/projects/lportal/files/Liferay%20IDE/>

¹⁶<https://chemistry.apache.org/>

Über einen Iterator wird durch alle Propertys durchgegangen und diese mit Hilfe der `ExpandoBridge` in die Liferay-Datenbank eingebracht.

```

1  public class TestListener2 implements ModelListener<RepositoryEntry> {
2
3      @Override
4      public void onAfterCreate(RepositoryEntry arg0)
5          throws ModelListenerException {
6          try {
7              Session s = createCmisSession();
8
9              ObjectId id = s.createObjectId(arg0.getMappedId().toString());
10
11             CmisObject o = s.getObject(arg0.getMappedId());
12
13             if (o.getProperty(PropertyIds.BASE_TYPE_ID).getValueAsString().
14                 contains("cmis:folder")) {
15                 //Es handelt sich um einen Ordner
16             } else {
17                 List<Property<?>> l = s.getObject(id).getProperties();
18                 Iterator<Property<?>> i = l.iterator();
19
20                 while (i.hasNext()) {
21                     Property<?> p = i.next();
22                     Object value = p.getValue();
23                     PropertyType t = p.getType();
24
25                     try {
26                         if (!arg0.getExpandoBridge().hasAttribute(p.getLocalName()
27                             ())) {
28                             arg0.getExpandoBridge().addAttribute(p.getLocalName()
29                             ());
30                         }
31                     }
32                 }
33             }
34         } catch (PortalException e) {
35             e.printStackTrace();
36         }
37     }
38 ...

```

Listing 15: Prototypische-Implementierung einer CMIS-Schnittstelle mit Apache Chemistry

Dieser kleine Ausflug beweist, dass es möglich ist, die von Alfresco ankommenden Metadaten in Liferay zu integrieren. Somit steht auch dem Anzeigen der Metadaten auf der Oberfläche nichts mehr im Wege. [Che15]

Der Aufwand, eine komplette Portierung der Metadaten umzusetzen ist relativ gering, zumal der im Prototyp vorhandene Code ohne größere Änderungen übernommen werden kann. Es muss lediglich noch ein Filter eingebaut werden, um nur die gewünschten Metadaten in der Datenbank zu speichern.

Die Umsetzung der Anzeige und einer Suchfunktion in dem der „Asset Publisher“ erweitert wird, sollte bei vorhandenen Daten ohne Probleme möglich sein.

Zu beachten ist, dass der erstellte Hook Daten nur einmal lädt. Für die spätere Synchronisation sind weitere Implementierungsschritte notwendig.

9.4 Entwicklung eines neuen Liferay-Portlets

Im Abschnitt 9.3.2 wurde beschrieben, wie die Metadaten von Alfresco abgefragt und in der Datenbank gespeichert werden können.

Eine andere Möglichkeit, ist die Daten nicht über einen Hintergrundprozess abzufragen, sondern direkt in einem Portlet. Dies hat den Vorteil, dass eine Filterung der Metadaten nach gewissen Kriterien von Alfresco vorgenommen wird. Auch entsteht durch die Live-Abfrage keine Daten-duplikation, was jedoch vermutlich die Abfragegeschwindigkeit verlangsamt. Zusätzlich muss für ein Portlet auch eine eigene Oberfläche implementiert werden, welche die abgefragten Daten von Alfresco anzeigt.

Ob sich daher die Implementierung eines Portlets vom Aufwand und der Abfragegeschwindigkeit her lohnt kann leider nicht genau gesagt werden. Dies soll aber im Verlauf der weiteren Entwicklung evaluiert werden.

9.4.1 Entwicklung von Widgets

Innerhalb der Projektgruppe werden oft kleine JavaScript-Widgets verwendet, die gewisse Informationen darstellen und auch Abfragen können. Diese JavaScript-Widgets lassen sich beliebig in jede Webseite und auch in Liferay integrieren und können dort Verwendung finden. Für die möglichst nahtlose Integration in Liferay können Widgets in Portlets verpackt werden, die für Redakteure und Autoren dann wie herkömmliche, native Portlets funktionieren.

Es müsste also ein Widget geschrieben werden, welches die CMIS-Daten von Alfresco ausliest und anzeigt. Zusätzlich muss es auch eine Suchfunktion bieten, welche es erlaubt die Dokumente nach bestimmten Metadaten zu filtern. Der Arbeitsaufwand sollte in etwa ähnlich dem eines Portlets (siehe 9.4) sein, jedoch mit dem Vorteil, dass ein solches Widget überall integriert werden kann, wo JavaScript lauffähig ist.

Für die Anbindung an Alfresco kann direkt die Alfresco-JavaScript-API verwendet werden.

Eine weitere, unabhängige Möglichkeit wäre die Synchronisation der Alfresco-Metadaten mit einer strukturierten Suchmaschine wie zum Beispiel ElasticSearch¹⁷. Diese stellt zwar zunächst nur eine andere Schnittstelle mit denselben Daten zur Verfügung, jedoch existieren bereit Widgets zur Abfrage, Filterung und Darstellung von ElasticSearch-Daten. Mit Alfresco Acitiviti steht ein Plugin zur Synchronisation von Alfresco und ElasticSearch zur Verfügung. Eine Umsetzung auf dieser Basis wurde jedoch im Rahmen dieser Arbeit aus Zeitgründen nicht gemacht. [Wik15d] [ela15]

9.4.2 Liferay Enterprise Edition

Innerhalb der Arbeit wurde auch getestet, ob die Enterprise Edition von Liferay im Gegensatz zur bisher verwendeten Community Edition in der Lage ist, alle CMIS-Daten abzufragen und zu speichern. Hierfür wurde die Testversion genutzt, welche zwar den vollen Funktionsumfang bietet, jedoch auf 30 Tage limitiert ist.

Es wurde kein Unterschied zum Verhalten der Community Edition festgestellt. Auch bietet der Store von Liferay in der Enterprise Edition keine anderen Alfresco- oder CMIS-Plugins als in

¹⁷<https://www.elastic.co/products/elasticsearch>

der Community Edition. Somit bietet die Verwendung der Enterprise Edition für die vorliegende Aufgabe keine entscheidenden Vorteile.

9.4.3 Verwendung von Liferay 7

In Liferay 7, welches sich zur Zeit noch in der Entwicklung befindet, wird zum einen das Dokumentmanagement von Liferay umgebaut und zum anderen die CMIS-Schnittstelle erweitert. [Kub14]

Es ist somit gut möglich, dass Liferay sehr bald schon eigenständig in der Lage ist, alle Metadaten von Alfresco anzuzeigen und zu verwalten. Im Verlauf der Arbeit wurde eine „Mile-Stone“-Version installiert, um die Behauptung genauer zu prüfen.

Es konnte in der Vorabversion von Liferay zwar ein Repository angelegt werden, jedoch war die Version nicht in der Lage, Dokumente zu laden. Die Logausgabe von Liferay legt nahe, dass momentan die Datenbank-Querys noch fehlerhaft sind. Daher konnte die Funktionalität der neuen CMIS-Schnittstelle nicht überprüft werden.

9.5 Auswertung der Möglichkeiten

Es ist ohne weiteres Zutun nicht möglich, die Metadaten aus Alfresco direkt in Liferay anzuzeigen. Es wurde aber im Abschnitt 9.3.2 gezeigt, dass eine nachträgliche Implementierung dieser Funktionalität grundsätzlich möglich ist. Aus zeitlichen Gründen war es aber im Rahmen dieser Arbeit nicht machbar gewesen, ein vollständiges Beispiel zu Implementieren.

Andere Alternativen wie die Entwicklung von Widgets, Portlets oder Hooks sind grundsätzlich möglich. Eine Implementierung ist zwar möglich, jedoch aufwändig, weshalb dieser Ansatz nur prototypisch im Abschnitt 9.3.2 beschrieben wurde. Die nur lose Kopplung von Alfresco und Liferay-Daten über Widgets und Portlets erschwert die Verknüpfung von Daten aus beiden Welten und damit die Erstellung von übergreifenden Anwendungen.

Neben der Erweiterung des CMIS-Repositories, im Liferay-Code, erscheint als bisher die beste Lösung die Erweiterung von Liferay über einen Hook, da es für Liferay keine fertigen Plugins gibt. Eine weitere Möglichkeit besteht darin, abzuwarten was Liferay in der Version 7 bietet. Um die Live-Abfrage von Portlets oder Widgets zu beschleunigen, könnten die Alfresco-Daten zum Beispiel in eine ElasticSearch Suchmaschine eingebracht und über diese abgefragt werden.

10 Vergleich zwischen Altsystem und dem neuen Prototyp

Ein direkter Vergleich der Funktionalität, zwischen Altsystem basierend auf WebGenesis und Alfresco mit dem neuen, vereinigten Metadatenmodell wie im Lastenheft gefordert, kann leider nicht umgesetzt werden. Dies ist nicht möglich, da die Implementierung des Frontends grundsätzlich gescheitert ist.

Auf einen visuellen Vergleich (der Benutzeroberflächen) muss an dieser Stelle verzichtet werden, da es im Frontend-Bereich nicht möglich ist, ein Ergebnis zu präsentieren. Im Backend wäre der Vergleich zwar möglich, jedoch nicht sinnvoll, da die verwendeten Technologien zu unterschiedlich sind. So setzt Alfresco auf eine moderne Datenbank und bietet ein Frontend welches auf „Bootstrap“ basiert. Das alte Front- und Backend basiert auf dem technisch veralteten „WebGenesis“¹⁸.

10.1 Vergleich des Frontends

Es war, wie im Kapitel 9 beschrieben, nicht möglich die Metadaten von Alfresco einfach in Liferay anzuzeigen. Liferay bietet, zumindest zur Zeit, keine Implementierung, um alle CMIS-Daten, welche von Alfresco zur Verfügung gestellt werden, zu verarbeiten. Des Weiteren gibt es auch kein Plugin, welches diese Funktionalität bietet.

Durch die Entwicklung eines Hooks für Liferay konnte zumindest gezeigt werden, dass Metadaten von Alfresco in Liferay importiert werden können. Eine Implementierung einer Anzeige- und Suchfunktion konnte aus Zeitgründen nicht umgesetzt werden, ist aber grundsätzlich möglich, nachdem die Daten in der Liferay-Datenbank vorhanden sind.

In den Abschnitten 9.3.2 bis 9.4.3 wurden verschiedene Möglichkeiten zur Verwendung eines Hooks und weitere Alternativen aufgezeigt.

10.2 Vergleich des Backends

Das Backend ist prototypisch implementiert und kann genutzt werden. Es ist möglich, Dateien mit allen Metadaten zu versehen, wie es auch im alten System möglich ist. Zusätzlich können weitere Aspekte an Metadaten hinzugefügt oder entfernt werden, wodurch das neue System um einiges flexibler ist als das alte.

Ein weiterer Vorteil ist es, dass neue Attribute, Aspekte oder Typen relativ leicht und ohne viel Aufwand eingefügt oder entfernt werden können. Hierfür muss lediglich das Content-Modell welches, in XML-Dateien beschrieben ist (siehe Kapitel 8), geändert werden. Der zentrale Punkt, dass Aspekte und somit Teile dem Modells wiederverwendet werden können, wurde umgesetzt. Es sollte später noch eine Modellierungssoftware geschrieben werden, welche es ermöglicht, das Modell über eine GUI anzupassen. Hierdurch ist es noch einmal um einiges leichter, das Datenmodell zu verändern, wie es im Lastenheft verlangt wird.

Das offene DMS-System Alfresco bietet neben einer webbasierten Autoren-Redakteurs-Schnittstelle viele weitere Schnittstellen, wie zum Beispiel CMIS oder Web-DAV und gliedert sich somit sehr gut in die serviceorientierte System-Landschaft der LUBW ein.

Alfresco und somit auch das Backend ist in der Lage, alle Metadaten anzuzeigen und nach ihnen zu suchen. Ob dies später einmal von Vorteil sein wird ist fraglich, jedoch ist es kein Nachteil.

¹⁸<http://www.iosb.fraunhofer.de/servlet/is/18052/>

Es kann also gesagt werden das der Prototyp des Backends dem alten System ebenbürtig ist. Zu beachten sind jedoch die Einschränkungen, welche zur Zeit noch im Prototyp bestehen und im Abschnitt 8.10 genauer beschrieben sind.

11 Zusammenfassung

Die vorliegende Arbeit beschäftigte sich mit der Implementierung eines systemübergreifenden Dokumentenmanagementsystems. Hierfür wurde im Verlauf der Arbeit als erstes ein Lastenheft erstellt, welches alle Anforderungen für einen ersten Prototypen festhält. Es wurde auf den zukünftigen Produkteinsatz, sowie auf Produktfunktionen, Produktdaten und Produktleistungen genauer eingegangen.

Für die Entwicklung eines neuen Systems, mussten die bisher bestehenden Systeme, insbesondere deren Metadatenmodelle, genauer analysiert werden. Dafür wurden die Internetportale FADO, DRS und das Naturschutz-Bildarchiv der LUBW betrachtet (siehe Kapitel 4). Zusätzlich wurde auch das Literaturarchiv der ICT-ENSURE in die Analyse einbezogen.

Im Kapitel 4 wurden nach einer oberflächlichen Analyse die Metadaten aller Systeme gesammelt und aufgelistet. Dies war notwendig, da im späteren Verlauf der Arbeit ein übergreifenden Metadatenmodell aller Systeme erstellt wurde.

Da die Europäische Union erlassen hat, dass jede vom Land neu veröffentlichte Datei mit Geodatenbezug einen standardisierten Metadatensatz INSPIRE haben muss, wurde auch in dieser Richtung analysiert. So entstand im Kapitel 5 eine Unterteilung zwischen fachlichen und technischen Metadatenstandards. Hier wurden mehrere Metadatenstandards genauer analysiert, um sie in das übergreifende Metadatenmodell einbeziehen zu können.

Nachdem alle Voraussetzungen gegeben waren, konnte ein übergreifendes Metadatenmodell erstellt werden, welches zunächst möglichst weit aufgefächert und dadurch hochmodular gestaltet ist. Durch diese Modularität ist es relativ einfach, das Modell zu erweitern und neue Dokumententypen können schon vorhandene Untergruppen nutzen. Im Kapitel 6 wird das Modell genauer betrachtet und erklärt.

Nachdem das Datenmodell für die Erstellung eines Prototyps gegeben war, konnten verschiedene ECM-Systeme verglichen und auf ihre Projekttauglichkeit geprüft werden. In der Arbeit wurden die vier ECM-Systeme „agorum core“, „Alfresco“, „Open-Xchange“ und „Liferay“ näher betrachtet. Auch wenn Liferay im eigentlichen Sinn ein Portalserver ist, so besitzt auch er eine konfigurierbare Dokumentverwaltung, welche als ECM-System gesehen werden kann.

Als Ergebnis wurde eine Auswertung nach Kriterien aus dem Lastenheft erstellt. Hierbei ging Alfresco eindeutig als Sieger hervor, weshalb in der weiteren Bearbeitung dieses ECM-Systems für die Realisierung eines Backends gewählt wurde (siehe Kapitel 7).

Das zuvor erstellte allgemeine Metadatenmodell wurde an Alfresco angepasst. Dies war notwendig, da Alfresco keine Unterklassen von Metadaten als Attribute zulässt. Des Weiteren implementiert Alfresco gewisse Metadatenstandards wie das Exif-Format oder Dublin Core standardmäßig, so dass diese Implementierungen der Standards innerhalb des Metadatenmodells genutzt wurden.

Das angepasste Metadatenmodell, wurde in Alfresco implementiert. Hierfür musste es mit Hilfe von XML-Dateien beschrieben werden. Zum einen werden Dateien benötigt, welche das Datenmodell grundlegend beschreiben, zum anderen werden auch Dateien benötigt, die eine korrekte Darstellung der Metadaten innerhalb von Alfresco ermöglichen. Das Metadatenmodell wird beim Start von Alfresco als Java-Beans geladen und kann verwendet werden.

Eine Übersetzung des Metadatenmodells in andere Sprachen wurde aus Zeitgründen nur prototypisch umgesetzt, um die Funktionalität zu demonstrieren. Der massenhafte Import von Daten aus dem FADO-System wurde am Beispiel eines Fachsystems erprobt.

Die Integration der Alfresco-Daten in den Portalserver Liferay konnte nicht vollständig umgesetzt werden, da es keine fertige Implementierung für diesen Use-Case gibt. Liferay ist weder standardmäßig noch über ein Plugin in der Lage, die Metadaten von Alfresco abzufragen und zu speichern.

Im Kapitel 9 wurden daher Möglichkeiten vorgestellt, wie eine nachträgliche Bereitstellung der gewünschten Funktionalität möglich ist. Neben der Erweiterung der entsprechenden Schnittstelle in Liferay selbst ist die beste Möglichkeit die Implementierung eines Hooks, welcher es erlaubt, die Metadaten aus Alfresco auszulesen und sie redundant in der Liferay-Datenbank zu speichern. Dies wurde prototypisch implementiert, um die Machbarkeit zu demonstrieren. Eine Anzeige- und Suchfunktion wurde nicht realisiert, sie sollte aber, nachdem die Daten in der Liferay-Datenbank vorhanden sind, relativ leicht zu implementieren sein.

11.1 Umsetzung des Lastenhefts

Ziel der Arbeit war es ein Backend für die Verwaltung von Dokumenten der LUBW zu erstellen, und dabei ein vereinheitlichtes Metadatenmodell für mehrere Altsysteme zu schaffen. Dies konnte auch erfolgreich umgesetzt werden. Das Frontend, welches auf Liferay basieren sollte, war kritischer, da Liferay nicht in der Lage ist, Metadaten von Alfresco zu verarbeiten. Es wurde zwar eine prototypische Erweiterung für Liferay erstellt, den vollen Funktionsumfang bietet sie jedoch nicht.

Es konnten somit bis auf die Produktfunktionen „LF40“ (Suche in den Metadaten über Frontend) und „LF60“ (Frontend Prototyp) alle Anforderungen erfüllt werden. Die Metadatensuche in Liferay konnte aus Zeitgründen nicht umgesetzt werden. Da ein Anzeigen der Metadaten auch aus Zeitgründen nicht möglich war, konnte der Prototyp nicht mit den alten LUBW-Systemen verglichen werden, es wurde jedoch im Sinne eines proof-of-concept gezeigt, dass dies grundsätzlich möglich ist.

Die im Lastenheft festgelegten Produktdaten (siehe Abschnitt 3.4) beziehen sich alle auf das Backend und konnten komplett umgesetzt werden. Im Verlauf der Arbeit wurde ein modulares Metadaten-Modell entwickelt, welches den Anforderungen des Lastenhefts fast genügt. Alfresco erlaubt es nicht, Metadaten zu erstellen, welchen keine physische Datei zugrunde liegt. Somit muss ein Autor, wie in „LD30“ beschrieben, als Attribut gespeichert werden und kann nicht als Entität existieren. Diese Änderung ist jedoch akzeptabel.

Existierende Standards wurden im Verlauf der Arbeit betrachtet und so weit möglich und sinnvoll implementiert. Hier war vorteilhaft, dass Alfresco bereits einige Standards implementiert hat, so zum Beispiel den „Dublin Core“- oder den „Exif“-Standard.

Die beschriebenen Produktleistungen konnten zwar für das Backend umgesetzt werden, jedoch aus den schon beschriebenen Gründen nicht für das Frontend. Somit konnte der Punkt „LL30“(grobe und feine Suche im Frontend) nur zum Teil umgesetzt werden. Allen anderen Punkten konnte jedoch Genüge getan werden.

12 Ausblick

Da innerhalb der Arbeit weder das gesamte FADO-System noch andere Systeme der LUBW analysiert und in das Modell einbezogen wurden, müssen alle noch fehlenden Dokumentenklassen mit ihren Metadaten zum Modell hinzugefügt werden. Entsprechend muss auch das Datenmodell von Alfresco erweitert werden. Momentan fehlen im FADO-System noch die Fachdokumente für „Altlasten“, „Chemikalien und Arbeitsschutz“, „UIS Medien“, „Umweltbeobachtung“ und „Umweltforschung“, diese haben jedoch dasselbe Datenmodell wie die für "Boden", so dass es sich hier lediglich noch um weitere Export- und Importvorgänge handelt.

Das Lastenheft verlangte, dass eine Änderung der grundlegenden Modells recht einfach gestaltet werden soll. Das Schreiben von XML-Quellcode ist jedoch für Laien nicht zumutbar, weshalb überlegt werden sollte, ob die Erstellung eines Metadateneditors, welcher die XML-Dateien verwaltet, sinnvoll ist. Dieser Editor könnte gegebenenfalls auch die Konsistenz und Vollständigkeit der erstellten bzw. bearbeiteten Dateien sicherstellen.

Das Alfresco-Formular zur Eingabe der Metadaten ist momentan, nach den einzelnen Aspekten und Typen gegliedert. Es werden ausnahmslos alle Metadaten angezeigt, egal ob sie für den entsprechenden Metadatensatz relevant sind oder nicht. Hier sollte in Absprache mit den jeweiligen Fachleuten der LUBW Ordnung entstehen. Das heißt, es müssen optionale Metadaten ausgeblendet werden können und die Gliederung der nach diesem Schritt verbleibenden Attribute sollte auf die Bedürfnisse der Mitarbeiter der LUBW angepasst werden.

Die prototypische Implementierung des Backends ist bis auf die genannten Erweiterungen vollständig.

Der größere Arbeitsaufwand wird bei der Erstellung eines Frontends entstehen. Wie in der Arbeit im Kapitel 9 erwähnt, war es nicht möglich auf eine vollständig implementierte Schnittstelle zwischen Alfresco und Liferay zurückzugreifen. Daher muss entweder das bestehende CMIS-Repository ergänzt oder eine Eigenlösung entwickelt werden.

In der Arbeit wurde prototypisch gezeigt, wie es möglich ist, die Metadaten aus Alfresco über einen Hook in Liferay verfügbar zu machen. Sollte dieser Ansatz in Zukunft weiter verfolgt werden, können die gewonnenen Erkenntnisse und der entstandene Quellcode dieser Arbeit als Grundlage dienen.

Es sollte im weiteren Verlauf des Projekts evaluiert werden, welche der in den Abschnitten 9.3.1 bis 9.4.3 diskutierten Alternativen am besten für einen späteren produktiven Einsatz geeignet ist.

13 Abkürzungsverzeichnis

ECM *Enterprise Content Management*

DMS *Dokumentenmanagementsysteme*

PoC *Proof of Concept*

LUBW *Landesanstalt für Umwelt, Messungen und Naturschutz Baden-Württemberg*

GAA *Gewerbeaufsicht Baden-Württemberg*

REST *Representational State Transfer*

FADO *Fachdokumente Online*

DRS *Document Retrieval System*

INSPIRE *Infrastructure for Spatial Information in the European Community*

API *Application Programming Interface*

EU *Europäische Union*

DCMI *Dublin Core Metadata Initiative*

SEO *search engine optimization*

Exif *Exchangeable Image File Format*

RDF *Resource Description Framework*

ICT-ENSURE *Literature Information System in the Field of ICT for Environmental Sustainability*

ONIX *Online Information Exchange*

GPL *General Public License*

UI *User Interface*

CMIS *Content Management Interoperability Services*

14 Anhang

14.1 Metadaten der LUBW Fachsysteme

In den folgenden Abschnitten sind die Metadaten und Relationen der Untersuchten Dokumente in FADO abgebildet.

14.1.1 Metadaten des Fachsystems FADO / Urteile

Metadaten:	Werte:
Fachsystem	Vorgabe
ID	Generiert
Titel	Freitext
Tenor	Freitext
Kommentar	Freitext
Orientierungssatz	Freitext
Norm	Freitext
Leitsatz	Freitext
Gericht	Freitext
Entscheidungsform	Freitext
Entscheidungsdatum	Datum
Aktenzeichen	Freitext
Vorgericht	Freitext
Nachgericht	Freitext
Sachverhalt	Freitext
Gründe	Freitext
Unsichtbar	Bool
ausblenden	Bool

Tabelle 8: Metadaten der Urteile in FADO

Relationen:	Werte:
Thema	Vorgabe
gehört zu Fachobjekt	Vorgabe
hat Schlagwort	Vorgabe
ist vom Typ	Vorgabe
enthalten in Fachsystem	Vorgabe
wird referenziert von	Vorgabe

Tabelle 9: Relationen der Urteile in FADO

14.1.2 Metadaten des Fachsystems FADO / Forschungsvorhaben

Metadaten:	Werte:
Fachsystem	Vorgabe
ID	Generiert
Title	Freitext
Kurzbeschreibung	Freitext
Kommentar	Freitext
Förderbereich	Freitext
Beginn	Datum
Ende	Datum
Projektnummer	Freitext
Förderkennzeichen	Freitext
Unsichtbar	Bool
ausblenden	Bool

Tabelle 10: Metadaten der Forschungsvorhaben in FADO

Relationen:	Werte:
Thema	Vorgabe
gehört zu Fachobjekt	Vorgabe
hat Abschlussbericht	Vorgabe
hat Forschungsberichtsblatt	Vorgabe
hat Projektskizze	Vorgabe
hat Schlagwort	Vorgabe
hat Zwischenbericht	Vorgabe
wird geleitet von	Vorgabe
wird referenziert von	Vorgabe

Tabelle 11: Relationen der Forschungsvorhaben in FADO

14.1.3 Metadaten des Fachsystems FADO / Berichte

Metadaten:	Werte:
Fachsystem	Vorgabe
ID	Generiert
Titel	Freitext
Kurzbeschreibung	Freitext
Kommentar	Freitext
Kurztitel	Freitext
Untertitel	Freitext
Fachthema	Freitext
Herausgeber	Freitext
Redaktion	Freitext
Version	Freitext
Stand	Datum
Seitenzahl	Freitext
Seite (von-bis)	Freitext
Reihe	Freitext
Bandnummer	Freitext
ISSN	Freitext
ISBN	Freitext
Preis	Freitext
Medium	Freitext
Shoprelevant	Bool
Shoplink	URL
HTML-Datei	Data
PDF-Datei	Data
Weitere Datei	Data
Format dieser Datei	Freitext
Unsichtbar	Bool
ausblenden	Bool

Tabelle 12: Metadaten der Berichte in FADO

Relationen:	Werte:
betrifft Thema	Vorgabe
gehört zu Fachobjekt	Vorgabe
hat Autor	Vorgabe
hat Schlagwort	Vorgabe
ist vom Typ	Vorgabe
enthalten in Fachsystem	Vorgabe
ist Abschlussbericht von	Vorgabe
ist Forschungsberichtsblatt von	Vorgabe
ist Projektskizze von	Vorgabe
ist Zwischenbericht von	Vorgabe
wird referenziert von	Vorgabe

Tabelle 13: Relationen der Berichte in FADO

14.1.4 Metadaten des DRS

Metadaten:	Werte:
Gültigkeit	Vorgabe
Titel	Freitext
Aktenzeichen	Freitext
Kurz-Titel	Vorgabe
Dokumentart	Vorgabe
Herausgeber	Vorgabe
Erscheinungsort	Vorgabe
Handbuch	Vorgabe
Kapitel	Vorgabe
Fundstelle	Vorgabe
Fassung	Vorgabe
Änderung	Vorgabe
Größe	Zahl
Formate	Vorgabe

Tabelle 14: Metadaten des DRS

Zu beachten ist hier, dass die Metadaten-Tags Fassung und Änderung zusammengesetzte Daten enthalten, wie es in der Abbildung 1 zu sehen ist.

14.1.5 Metadaten des Bildarchivs

Metadaten:	Werte:
Objektart	Freitext
Objektname	Freitext
ID	Generiert
URL	URL
Dateityp	Vorgabe
Name	Freitext
Kurzname	Freitext
Erstellt am	Datum
Autor	Freitext
Besitzer	Besitzer
Bemerkung	Freitext

Tabelle 15: Metadaten des Bildarchivs

14.2 Metadaten der ICT-ENSURE

In den folgenden Abschnitten sind die Metadaten der Dokumente der ICT-ENSURE zu finden. Diese werden hier in UML-Form abgebildet, was auch dem eigentlichen Datenmodell hinter ICT-ENSURE entspricht. Alle Felder, die im Modell abgebildet sind, müssen später im ECM-System Verwendung finden, mit Ausnahme der Klassen DocumentServer und URLGenerator. [SLG10]

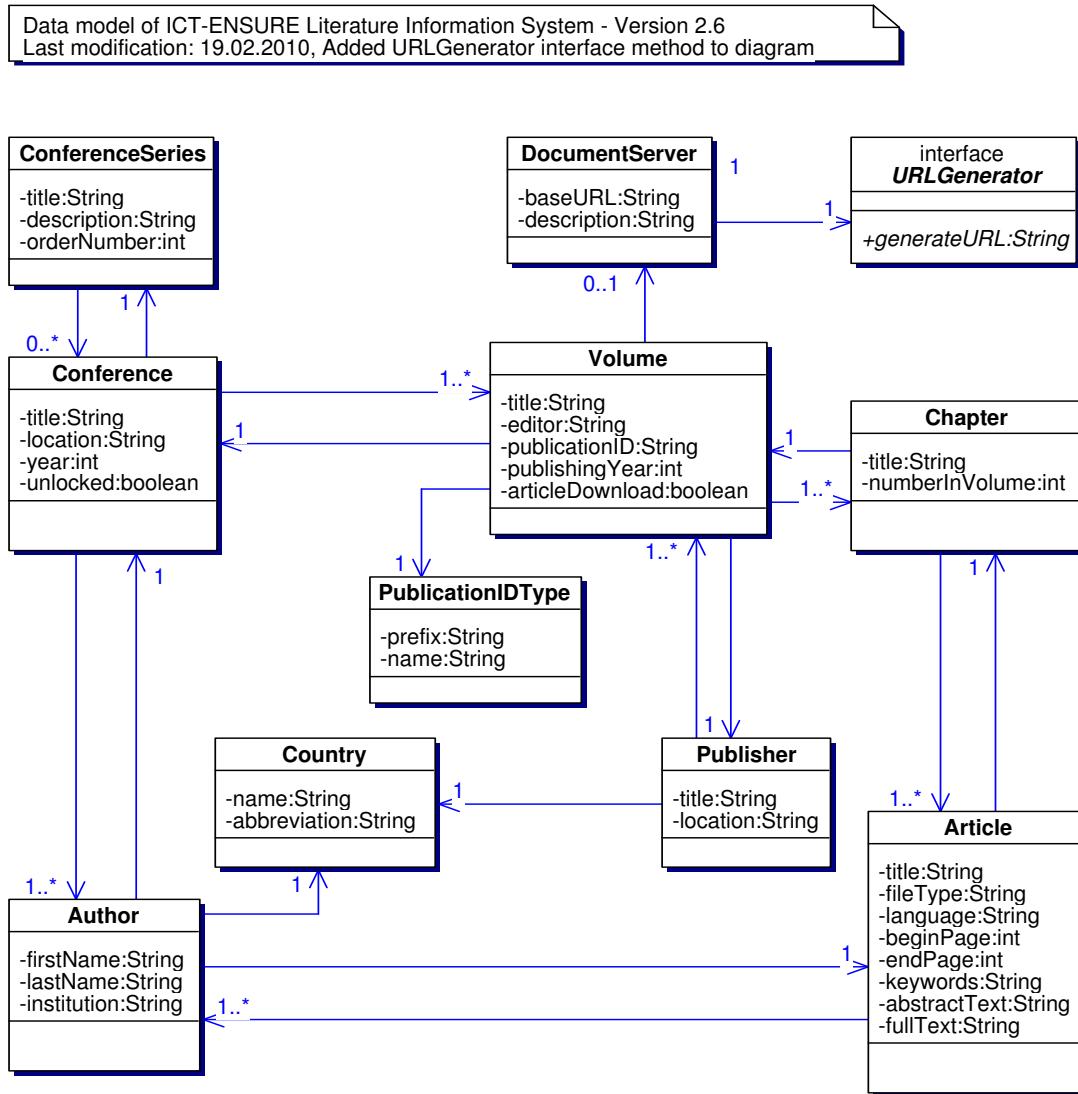


Abbildung 24: Datenmodell des Literaturarchivs ICT-ENSURE [SLG10]

14.3 Exif-Metadaten von Alfresco

▼ Eigenschaften	
Name:	BlickvomBalkonGiglio.JPG
Titel:	(Kein)
Beschreibung:	(Kein)
MimeType:	JPEG Image
Autor:	(Kein)
Größe:	463 KB
Ersteller:	admin
Datum der Erstellung:	Die 30 Jun 2015 14:23:20
Bearbeiter:	admin
Datum der Änderung:	Die 30 Jun 2015 14:23:20
Datum und Uhrzeit:	Fre 26 Mai 2006 20:48:50
Bildbreite:	1704
Bildhöhe:	2272
Belichtungszeit:	0.0015625
Blende:	4
Blitz eingeschaltet:	Nein
Brennweite:	10.6875
ISO Empfindlichkeit:	(Kein)
Hersteller der Kamera:	Canon
Kameramodell:	Canon PowerShot G2
Kamerasoftware:	(Kein)
Ausrichtung:	1
Horizontale Auflösung:	180
Vertikale Auflösung:	180
Auflösungseinheit:	Inch

Abbildung 25: Ausgelesene Exif-Daten von Alfresco

14.4 Bilder des Datenmodells

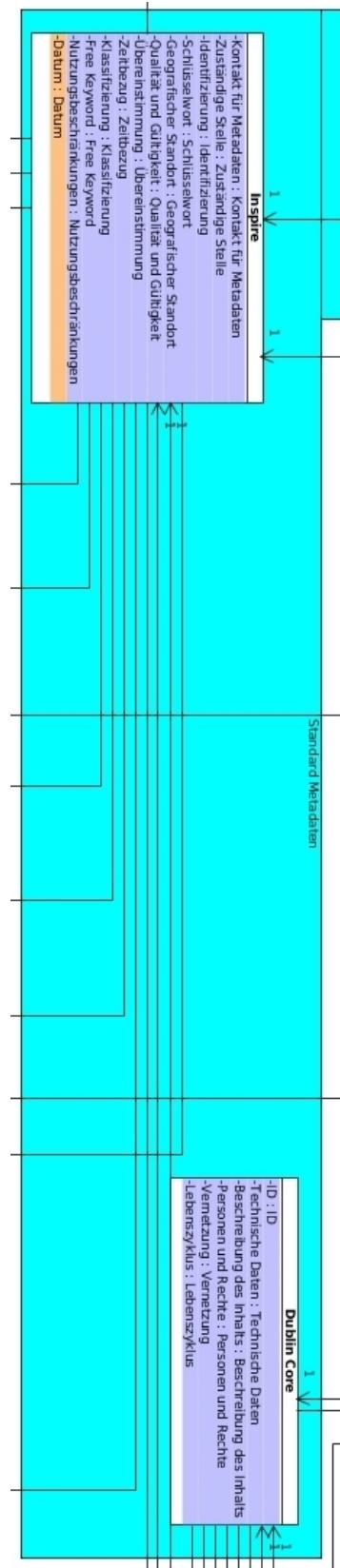


Abbildung 26: Package Standard Metadaten

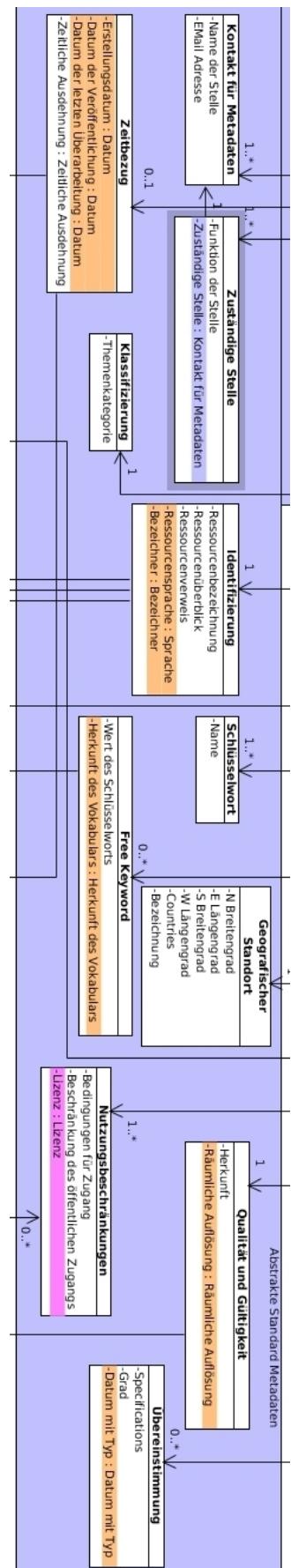


Abbildung 27: Package Abstrakte Standard Metadaten Teil 1 INSPIRE

14.5 Darstellung der Metadaten in Alfresco

<p>Content</p> <p>Name: 20_Jahre_bodendauerbeobachtung.pdf</p> <p>Titel: 20 Jahre Bodendauerbeobachtung in Baden-Württemberg</p> <p>Beschreibung: Die Bodendauerbeobachtung ist das im Landesbodenschutz- und Altlastengesetz verankerte Langzeitmonitoring-Programm Baden-Württembergs. Der Bericht stellt die inzwischen 20-jährige Entwicklung des Programms von klassischen Bodenuntersuchungen der ersten Dekade hin zu medienübergreifenden Umweltbilanzen dar. Anhand ausgewählter Ergebnisse wird verdeutlicht, dass sich Bodenveränderungen durch klassische Bodenuntersuchungen in der Regel nur sehr langfristig nachweisen lassen. Indem auch Stoffflüsse mit berücksichtigt werden, sind über den Bilanzansatz auch kurzfristig Änderungen erkennbar und darüber hinaus Ursachenanalysen sowie Prognosen möglich. Die Ergebnisse belegen die Bedeutung der Bodendauerbeobachtung als wichtige Informationsgrundlage für den vorsorgenden Bodenschutz. Neben Weiterentwicklungen der Konzeption und Bilanzierungsverfahren ist geplant, weitere Bodenschutzhemen wie Erosion und bodenbezogene Klimafolgen in das Programm zu integrieren.</p>
<p>Fado Metadaten</p> <p>Fachsystem: Boden</p> <p>Unsichtbar: Nein</p> <p>Ausblenden: Nein</p>
<p>Bibliografische Angaben Metadaten</p> <p>Redaktion: LUBW Landesanstalt für Umwelt, Messungen und Naturschutz, Referat 22</p>
<p>Relationen</p> <p>hat Schlagwort: Bodendauerbeobachtung, Bodenschutz, Bodenuntersuchung, Monitoring, Umweltbilanz</p> <p>hat Autor: Borho, Werner</p> <p>enthalten in Fachsystem: Boden</p> <p>ist Abschlussbericht von: (Kein)</p> <p>ist Forschungsberichtsblatt von: (Kein)</p> <p>ist Projektskizze von: (Kein)</p> <p>ist Zwischenbericht von: (Kein)</p> <p>wird referenziert von: (Kein)</p>

Abbildung 28: Darstellung der Metadaten in der Dokumentenübersicht

14.6 Darstellung einer Bulk-Import XML-Datei

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
3 <properties>
4     <entry key="type">lupo-FM:FADO-Urteil</entry>
5     <entry key="aspects">cm:versionable,cm:dublincore,lupo-SA:Inspire,lupo-BA:
       Dokumenten-Information,lupo-GB:Gericht,lupo-GB:Urteil</entry>
6     <entry key="cm:name"></entry>
7     <entry key="cm:title"></entry>
8     <entry key="lupo-GB:Tenor"></entry>
9     <entry key="lupo-BA:Kommentar"></entry>
10    <entry key="lupo-GB:Orientierungssatz"></entry>
11    <entry key="lupo-FM:Norm"></entry>
12    <entry key="lupo-GB:Leitsatz"></entry>
13    <entry key="lupo-GB:Standort_Vorgericht"></entry>
14    <entry key="lupo-GB:Art_Vorgericht"></entry>
15    <entry key="lupo-GB:Standort_Nachgericht"></entry>
16    <entry key="lupo-GB:Art_Nachgericht"></entry>
17    <entry key="lupo-GB:Gerichtsstandort"></entry>
18    <entry key="lupo-GB:Gerichtsart"></entry>
19    <entry key="lupo-GB:Sachverhalt"></entry>
20    <entry key="lupo-GB:GrÃ¼nde"></entry>
21    <entry key="lupo-FADO:Unsichtbar"></entry>
22    <entry key="lupo-FADO:Ausblenden"></entry>
23    <entry key="lupo-FM:enthalten_in_Fachsystem"></entry>
24    <entry key="lupo-FM:wird_referenziert_von-Urteil"></entry>
25    <entry key="lupo-FM:hat_Schlagwort"></entry>
26    <entry key="lupo-FM:Thema"></entry>
27    <entry key="lupo-FM:Entscheidungsform"></entry>
28    <entry key="lupo-GB:Entscheidungsdatum"></entry>
29    <entry key="lupo-GB:Aktenzeichen"></entry>
30 </properties>
```

Listing 16: Darstellung einer XML-Datei für den Bulk-Import

14.7 Repositorydarstellung in Liferay

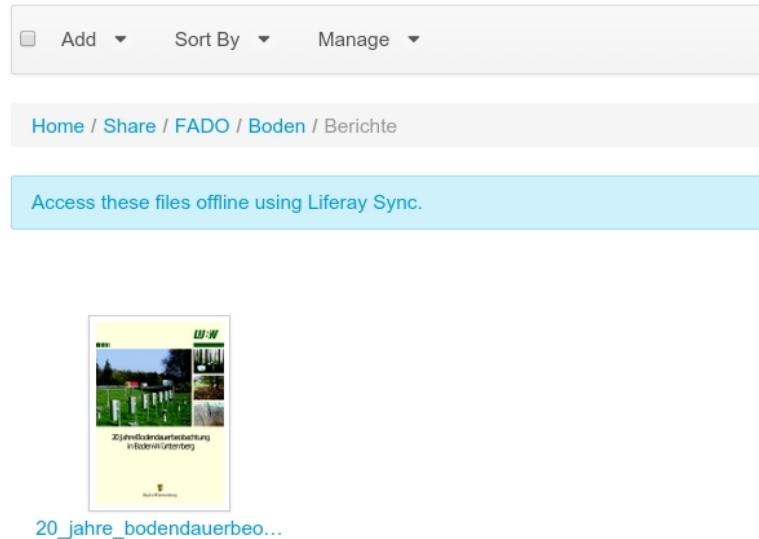


Abbildung 29: Das Textfeld Gründe nach der Bearbeitung

14.8 Die Methode createCmisSession()

```
1 private Session createCmisSession() {
2     SessionFactory f = SessionFactoryImpl.newInstance();
3     Map<String, String> parameter = new HashMap<String, String>();
4
5     // user credentials
6     parameter.put(SessionParameter.USER, "sebastian.rieger2");
7     parameter.put(SessionParameter.PASSWORD, "1234");
8
9     // connection settings
10    parameter.put(SessionParameter.ATOMPUB_URL, "http://localhost:8080/alfresco/
11        api/-default-/public/cmis/versions/1.1/atom");
12    parameter.put(SessionParameter.BINDING_TYPE, BindingType.ATOMPUB.value());
13    parameter.put(SessionParameter.REPOSITORY_ID, "-default-");
14
15    // session locale
16    parameter.put(SessionParameter.LOCALE_ISO3166_COUNTRY, "");
17    parameter.put(SessionParameter.LOCALE_ISO639_LANGUAGE, "de");
18
19    // create session
20    return f.createSession(parameter);
}
```

Listing 17: Die Methode createCmisSession()

Abbildungsverzeichnis

1	Suchmaske des DRS	12
2	FADO Metadaten	20
3	DRS und Bildarchiv Metadaten	20
4	ICT-ENSURE Metadaten	21
5	Package Gerichtbarkeit	22
6	Package Bibliographie	22
7	Package Abstrakte Metadaten	23
8	Package Standard Metadaten	24
9	Package Abstrakte Standard Metadaten Teil 1 INSPIRE	25
10	Package Abstrakte Standard Metadaten Teil 2 „Dublin Core“	26
11	Package Grundlegende Daten	27
12	Package Urheber Metadaten	27
13	Metadaten Designer von „agorum core“ im Web-Interface	35
14	„Administrator Dashboard“ von Alfresco	36
15	Datei-Anzeige von Open-Xchange	38
16	Content-Modell in Alfresco	42
17	FADO Datentypen für Alfresco	43
18	Bildarchiv und ICT-ENSURE Datentypen für Alfresco	44
19	DRS Datentyp für Alfresco	44
20	Die Klasse Gerichtbarkeit	45
21	Die Klasse Standard Aspekte	46
22	Die Klasse Standard Aspekte	47
23	Das Textfeld Gründe nach der Bearbeitung	57
24	Datenmodell des Literaturarchivs ICT-ENSURE [SLG10]	79
25	Ausgelesene Exif-Daten von Alfresco	80
26	Package Standard Metadaten	81
27	Package Abstrakte Standard Metadaten Teil 1 INSPIRE	82
28	Darstellung der Metadaten in der Dokumentenübersicht	83
29	Das Textfeld Gründe nach der Bearbeitung	85

Tabellenverzeichnis

1	Mapping der FADO Attribute für Urteile	28
2	Mapping der FADO Attribute für Forschungsvorhaben	29
3	Mapping der FADO Attribute für Berichte	30
4	Mapping der DRS Attribute	31
5	Mapping der Bildarchiv Attribute	32
6	Mapping der ICT-ENSURE Attribute	33
7	Tabellarischer Vergleich der betrachteten ECM-Systeme	40
8	Metadaten der Urteile in FADO	74
9	Relationen der Urteile in FADO	74
10	Metadaten der Forschungsvorhaben in FADO	75
11	Relationen der Forschungsvorhaben in FADO	75
12	Metadaten der Berichte in FADO	76
13	Relationen der Berichte in FADO	77
14	Metadaten des DRS	77
15	Metadaten des Bildarchivs	78

Listings

1	Darwin Core-Beispiel in XML	15
2	ONIX-Beispiel in XML [ONI09b]	16
3	Dublin Core-Beispiel in HTML	17
4	Inhalt der Datei Bildarchiv-Metadaten.xml	49
5	Inhalt der Datei Bildarchiv-Metadaten-context.xml	50
6	Ausschnitt des Inhalts der Übersetzungsdatei I18N_de_DE.properties	51
7	Beschreibung einer Klasse des Datamodels in der share-config-custom.xml . .	53
8	Beschreibung der Aspekte in der share-config-custom.xml	54
9	Beschreibung der Typumwandlung in der share-config-custom.xml	55
10	Beschreibung der custom-slingshot-application-context.xml	55
11	Appearance Anpassungen in der share-config-custom.xml	57
12	Bean zur Aufteilung der custom-slingshot-application-context.xml	58
13	Erweiterung in der Datei portal-setup-wizard.properties	61
14	Abfrage aller Propertys eines Nodes mittels CMIS 1.1	61
15	Prototypische-Implementierung einer CMIS-Schnittstelle mit Apache Chemistry .	65
16	Darstellung einer XML-Datei für den Bulk-Import	84
17	Die Methode createCmisSession()	85

Literatur

- [Aga13] Navin Agarwal. Integrating Alfresco Repository to Liferay Documents and Media Portlet. <https://www.liferay.com/de/web/navin.agarwal11/blog/-/blogs/integrating-alfresco-repository-to-liferay-documents-and-media-portlet>, Mai 2013.
- [ago15a] agorum core Metadaten-Designer. <http://www.agorum.com/startseite/produkte/zusatzmodule-fuer-agorum-core/agorum-core-metadaten-designer-workflow-masken-editor.html>, Juni 2015.
- [ago15b] agorum core Preisübersicht. <http://mein-dms.agorum.com/pricing/agorum-core-preise-uebersicht>, Juni 2015.
- [ago15c] Der Metadaten-Designer. Technical report, Juni 2015.
- [ago15d] Dokumentenmanagement - agorum core - Mein DMS. <http://mein-dms.agorum.com/>, Juni 2015.
- [Alf15a] Alfresco Documentation. <http://docs.alfresco.com/community/concepts/system-about-community.html>, 2015.
- [Alf15b] Alfresco Wiki Forms. <https://wiki.alfresco.com/wiki/Forms>, März 2015.
- [Alf15c] Dokumenten-Management | Enterprise Content Management | Alfresco, Juni 2015.
- [Bla07] Henk M. [Hrsg.] Blanken, editor. *Multimedia retrieval : with ... 11 tables*. Data-Centric systems and applications. Springer, Berlin, 2007.
- [Che15] Example code for OpenCMIS clients. <https://chemistry.apache.org/java/examples/index.html>, August 2015.
- [Cho12] Alexander Chow. Foren (Alfresco ECM). https://www.liferay.com/de/community/forums/-/message_boards/message/14496303, Juni 2012.
- [CMI] CMIS Repository. <http://www.liferay.com/de/community/wiki/-/wiki/Main/CMIS+Repository>.
- [CMI15] CMIS Repository. <https://www.liferay.com/de/community/wiki/-/wiki/Main/CMIS+Repository>, August 2015.
- [CNF⁺10] David Caruana, John Newton, Michael Farman, Michael Uzquiano, and Kevin Roast. *Professional Alfresco*. Wiley Publishing, Inc., 2010.
- [Dam11] Thomas Dambach. Evaluation eines Dokumentenmanagementsystems. B.s. thesis, Fachhochschule Nordwestschweiz, August 2011.
- [DHW14] Data Warehouse Wiki; Meta-Daten. <http://de.dwhwiki.info/konzepte/metadaten>, Oktober 2014.
- [DRS08] DRS - Document Retrieval System. <http://www.drs.baden-wuerttemberg.de/DRSWeb.dll/p52108.html>, 2008.
- [Dub15] The Metadata Community — Supporting Innovation in Metadata Design, Implementation & Best Practices. <http://dublincore.org/>, Juni 2015.

- [ela15] Elasticsearch | Search & Analyze Data in Real Time. <https://www.elastic.co/products/elasticsearch>, 2015.
- [Exi13] Exchangeable Image File Format for digital still cameras. Technical report, Av&IT Standardisation Committee, Mai 2013.
- [Fac] Flexible Metadatenverwaltung - für den erfolgreichen Einsatz der Daten über den Daten. <http://www.msg-gillardon.de/themengebiete/financial-business-intelligence/financial-bi/unsere-bi-expertise/metadatenverwaltung.html>.
- [Fer13] Divi Fernando. Dublin Core Metadata for SEO and Usability. <http://blog.woorank.com/2013/04/dublin-core-metadata-for-seo-and-usability/>, April 2013.
- [Get15] Getting information on a node. Technical report, Alfresco Software, Inc., 2015.
- [Göt14] Klaus Götzer, editor. *Dokumenten-Management : Informationen im Unternehmen effizient nutzen*. dpunkt-Verl., Heidelberg, 5., vollst. überarb. und erw. aufl. edition, 2014.
- [Ihl08] Jens Ihlenfeld. Groupware-Server Open-Xchange als kostenloser Download. <http://www.golem.de/0408/33264.html>, August 2008.
- [INS08] Amtsblatt der Europäischen Union - Richtlinie 2007/2/EG. <http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32008R1205&from=EN>, Dezember 2008.
- [INS15] INDSPIRE Geoportal. <http://inspire-geoportal.ec.europa.eu/>, Juni 2015.
- [Kub14] David Kubitza. Liferay 7.0 - was uns erwartet. <https://blog.ancud.de/home/-/blogs/liferay-7-0-was-uns-erwartet>, November 2014.
- [Leh08] Dr. Ing. Rainer Lehfeldt. Deutsche Übersetzung der Metadatenfelder des ISO 19115 Geographic information – Metadata. Technical report, Koordinierungsstelle GDI - DE, Dezember 2008.
- [Lei] Ralph Leipert. Das Metadatenmanagement der BI (Business Intelligence). <http://www.business-intelligence24.com/data-management/business-intelligence-metadatenmanagement>.
- [LUB] Landesamt für Umwelt, Messungen und Naturschutz Baden-Württemberg. <http://www.lubw.baden-wuerttemberg.de/servlet/is/35855/>.
- [Nat] Naturschutz-Bildarchiv. rips-dienste.lubw.baden-wuerttemberg.de/rips/ripperservices/apps/naturschutz/bildarchiv/default.aspx.
- [NIS06] Data Dictionary – Technical Metadata for Digital Still Images. Technical report, National Information Standards Organization, Dezember 2006.
- [ONI09a] ONIX. <http://www.editeur.org/83/Overview/>, 2009.
- [ONI09b] ONIX for Books Product Information Message How to describe digital products in ONIX 3 . Technical report, BISG, Dezember 2009.
- [Oxp15] Oxpedia HTTP API. http://oxpedia.org/wiki/index.php?title=HTTP_API, Juli 2015.

- [Pot07] Jeff Potts. Alfresco Developer : Working with Custom Content Types. <http://ecmarchitect.com/images/articles/alfresco-content/content-article.pdf>, Juni 2007.
- [Sez12] Richard L. Jr. Sezov. *Liferay in action : the official guide to Liferay Portal development*. Manning, Shelter Island, 2012.
- [Sha06] Munwar Shariff. *Alfresco Enterprise Content Management Implementation*. Packt Publishing, 2006.
- [SLG10] Christian Schmitt, Richard Lutz, and Werner Geiger. The ICT-ENSURE Information System on Literature in the Field of ICT for Environmental Sustainability . Technical report, Karlsruhe Institute of Technology (KIT), Februar 2010.
- [Wen13] Sebastian Wenzky. *Alfresco und Liferay ECM- und Portallösungen*. Hanser, 2013.
- [Wie15] John Wieczorek. Darwin Core Terms: A quick reference guide. Technical report, Biodiversity Information Standards (TDWG), Juni 2015.
- [Wik13a] Infrastructure for Spatial Information in the European Community. http://de.wikipedia.org/wiki/Infrastructure_for_Spatial_Information_in_the_European_Community, Juli 2013.
- [Wik13b] ISO 19115. http://de.wikipedia.org/wiki/ISO_19115, Juni 2013.
- [Wik15a] Alfresco (Software). http://de.wikipedia.org/wiki/Alfresco_%28Software%29, Februar 2015.
- [Wik15b] Darwin Core. http://en.wikipedia.org/wiki/Darwin_Core, März 2015.
- [Wik15c] Dublin Core. http://de.wikipedia.org/wiki/Dublin_Core, Mai 2015.
- [Wik15d] Elasticsearch. <https://de.wikipedia.org/wiki/Elasticsearch>, August 2015.
- [Wik15e] Exchangeable Image File Format. http://de.wikipedia.org/wiki/Exchangeable_Image_File_Format, April 2015.
- [Wik15f] Liferay Portal. https://de.wikipedia.org/wiki/Liferay_Portal, April 2015.
- [Wik15g] ONIX for Books. April 2015.
- [Wik15h] Open-Xchange. <https://de.wikipedia.org/wiki/Open-Xchange>, Mai 2015.
- [Wik15i] Representational State Transfer. http://de.wikipedia.org/wiki/Representational_State_Transfer, März 2015.
- [Wik15j] Resource Description Framework. http://de.wikipedia.org/wiki/Resource_Description_Framework, Juni 2015.
- [Xch15] Open-Xchange. <http://www.open-xchange.com/en/home>, Juli 2015.