
**Diplomarbeit**

# Evaluierung eines Frameworks für das Workflowmanagement zur Pflege digitaler Informationsobjekte

Roland Koppe

Abteilung Wirtschaftsinformatik

Erstgutachter: Prof. Dr.-Ing. Axel Hahn
Zweitgutachter: Dipl.-Inf. Stephan große Austing

Betreuerin: Dr. Ana Macario

Oldenburg, 26. März 2009

# Inhaltsverzeichnis

Inhaltsverzeichnis ..... III
Abkürzungsverzeichnis und Glossar ..... V
Abbildungsverzeichnis ..... IX
Tabellenverzeichnis ..... X
Listings ..... XI
1 Einleitung ..... 1
1.1 Szenario von Workflow und digitalen Informationsobjekten ..... 2
1.2 Fragestellungen ..... 3
1.3 Ziele der Diplomarbeit ..... 5
1.4 Aufbau der Arbeit ..... 5
2 Heranführung und Grundlagen ..... 7
2.1 Semantic Web ..... 7
2.1.1 Resource Description Framework (RDF) ..... 9
2.1.2 Ontology Web Language (OWL) ..... 11
2.2 Service-orientierte Architektur ..... 11
2.3 Web Services ..... 13
2.4 Workflows ..... 14
2.4.1 Ereignisgesteuerte Prozesskette ..... 16
2.5 Digitale Informationsobjekte ..... 18
3 Integration von Workflows und digitalen Informationsobjekten ..... 23
3.1 Anforderungen an die Integration ..... 23
3.2 Ansatz der Integration ..... 25
3.3 Anforderungen an die Definition von Workflows ..... 26
3.4 Sprache zur Definition von Workflows ..... 27
3.5 Szenario ..... 30
3.6 Vom Szenario Prozess zum Workflow ..... 32
3.7 Ontologie für digitale Informationsobjekte ..... 34
4 Architektur des Systems ..... 37
4.1 Architektur ..... 37
4.1.1 Repository für digitale Informationsobjekte ..... 39
4.1.2 Workflow-System ..... 39
4.1.3 Verwaltung von Objekttypen ..... 41
4.1.4 Rechte und Rollen ..... 41
4.1.5 Grafische Benutzungsschnittstelle und Frontends ..... 41
4.2 Architektur und Aufbau von Muradora ..... 42
5 Entwurf und Implementierung ..... 45
5.1 Verwendete Technologien ..... 45
5.2 Entwurf ..... 46
5.2.1 Anbindung an das Repository ..... 46
5.2.2 Workflow-System ..... 50
5.2.3 Verwaltung von Objekttypen ..... 54
5.2.4 Rechte und Rollen ..... 55

5.3 Implementierung ..... 57
5.3.1 Frontend für Funktionen ..... 57
5.3.2 Frontend für erweiterte Funktionen ..... 59
6 Evaluierung ..... 63
6.1 Einordnung der Evaluierung ..... 64
6.2 Evaluierung der Anforderungen ..... 64
6.2.1 Bewertung von Muradora und der Anwendungsumgebung ..... 74
6.3 Beantwortung der Fragestellungen ..... 79
6.4 Abschließende Bewertung mit Ausblick ..... 83
7 Zusammenfassung und Ausblick ..... 89
7.1 Ergebnisse ..... 89
7.2 Ausblick ..... 90
7.3 Erfahrungen ..... 92
8 Literatur ..... 95
Anhang ..... 101
A Checkliste der Anforderungen ..... 101
B XML Schema für Workflows ..... 103
C XML Dokument des Workflows im Szenario ..... 106
D Verwendete Technologien ..... 108

## Abkürzungsverzeichnis und Glossar

Dieser Abschnitt fasst das Abkürzungsverzeichnis und das Glossar für die vorliegende Arbeit zusammen. Bei einigen Begriffen wird auf entsprechende Stellen in dieser Arbeit oder auf weiterführende Literatur verwiesen.

| | |
|---|---|
| Ajax | Asynchronous JavaScript and XML ist ein Begriff, der Ansätze zusammenfasst, die es unter anderem erlauben Inhalte dynamisch von Servern zu laden und Benutzern damit die dynamische Interaktion mit einer Web-Anwendung ermöglichen (Wenz 2007). |
| Aggregation | Aggregation meint die Zusammenfassung von einzelnen Daten zu einem Datum. |
| Assoziation | Eine Assoziation bezeichnet die Verknüpfung von Objekten oder die Beziehung zwischen Objekten. |
| Annotation | Eine Annotation beschreibt Daten (→ semantisch) mit Hilfe von Metadaten. |
| Anwendungs-domäne | Eine Anwendungsdomäne bezeichnet ein als relevant betrachtetes und abgegrenztes Feld mit entsprechenden Anforderungen, Aufgaben und Problemstellungen (vgl. Dumke 2003, S. 11). |
| API | Application Programming Interface |
| Audit | Ein Audit Datastream speichert hier Informationen zur Bearbeitung eines digitalen Objekts und ermöglicht es damit den Status und die Korrektheit, sowie Rechtmäßigkeit der Bearbeitung zu kontrollieren. |
| AWI | Alfred-Wegener-Institut für Polar- und Meeresforschung |
| Backend | Backend meint eine Software, die hinter einem → Frontend agiert. |
| BPM | Business Process Management beschäftigt sich mit dem Management von Geschäftsprozessen, siehe Abschnitt 2.4. |
| CMA | Content Model Architecture, wird in → Fedora 3 zur Definition der Struktur von Inhalten verwendet. |
| DB | Datenbank |
| DC | Dublin Core ermöglicht durch standardisierte Bezeichnungen die anwendungsunabhängige Beschreibung von Dokumenten mit Metadaten. |
| DataBean | Eine DataBean ermöglicht das Anfragen und Zwischenspeichern von Daten, die über die Benutzungsschnittstelle einer Web-Anwendung präsentiert oder eingegeben werden. |
| Datastream | Ein Datastream ist eine bestimmte Ressource eines → digitalen Objekts, die eindeutig identifiziert werden kann, siehe Abschnitt 2.5. |
| digitales Objekt | Ein digitales Objekt kann verschiedene Daten in → Datastreams kapseln und eindeutig identifiziert werden, siehe Abschnitt 2.5. |
| Disseminator | Ein Disseminator eines digitalen Objekts kann in → Fedora 2 auf Dienste, wie Web Services, verweisen, siehe Abschnitt 2.5. |

| | |
|---|---|
| Dokument | Als Dokument werden in dieser Arbeit zusammenhängende und speicherbare beziehungsweise gespeicherte Daten und Informationen bezeichnet, die von Programmen verarbeitet werden können. |
| EPK | ereignisgesteuerte Prozesskette, siehe Abschnitt 2.4.1 |
| Fedora | Flexible Extensible Digital Object Repository Architecture. Das Fedora Framework stellt ein → Repository für digitale Objekte bereit. |
| Frontend | Frontend meint eine Software, die eng mit der Interaktion von Benutzern verknüpft ist und eine (grafische) Benutzungsschnittstelle enthält. |
| Generizität | etwa Allgemeinheit, Gegenteil von Spezialisierung |
| Granularität | Granularität bezeichnet den Grad der Feinkörnigkeit eines Systems, in etwa die Passgenauigkeit oder Detailliertheit. |
| Inferenz | Inferenz meint das logische (computergestützte) Schlussfolgern. |
| Instanz | Eine Instanz ist eine konkrete Ausprägung einer abstrakten Definition. So ist eine → Workflow-Instanz eine konkrete Ausprägung einer Workflow-Definition. |
| JSF | JavaServer Faces ist ein standardisiertes Framework für Java Web-Anwendungen zur Entwicklung von Benutzungsoberflächen nach dem → MVC Muster. |
| JSP | JavaServer Pages ermöglichen die Einbettung von Java-Code in statische Inhalte und dadurch die dynamische Präsentation von Inhalten einer Web-Anwendung. |
| Kardinalität | Die Kardinalität oder Multiplizität gibt die Mächtigkeit einer Beziehung an. Eine Publikation kann beispielsweise einen oder mehrere Autoren besitzen. |
| Klassifikation | Eine Klassifikation sammelt, bildet und ordnet abstrakte Klassen nach bestimmten Merkmalen, mit dem Ziel der Strukturierung. |
| Kurator | Als Kurator wird hier eine für digitale Objekte verantwortliche Person bezeichnet, in etwa Vorgesetzter. |
| LDAP | Lightweight Directory Access Protocol. Ein LDAP Server stellt ein Verzeichnis für beliebige Informationen bereit und ist im Regelfall für lesende Zugriffe optimiert. |
| multi-value | Multi-value meint die Angabe von multiplen Daten für einzelne Felder, beispielsweise die Angabe mehrerer Autoren für eine Publikation. |
| Migration | Die Migration meint hier die Umstellung von Software auf ein neues System oder neue Technologien und die damit verbundenen Aufgaben. |
| MVC | Model-View-Controller ist ein Konzept zur Trennung und Strukturierung von Software in die Zuständigkeiten Modell, Präsentation und Steuerung. |
| Objekttyp | Ein Objekttyp ist hier als → Prototyp – eine Art Modell oder auch Java-Klasse – für konkrete Objekte zu verstehen. |

| | |
|---|---|
| Ontologie | Eine Ontologie ist eine formale Spezifikation eines Konzepts von Begriffen und Beziehungen für eine Anwendungsdomäne, siehe Abschnitt 2.1. |
| Open Source | Open Source ist Software, für die der Quelltext unter bestimmten Lizenzen öffentlich zur Verfügung gestellt wird. |
| OWL | Ontology Web Language oder auch Web Ontology Language ist eine standardisierte Sprache zur Beschreibung von → Ontologien, siehe Abschnitt 2.1.2. |
| Persistenz | Persistenz meint die Möglichkeit Daten in nicht-flüchtigen Speichern, wie einem Dateisystem oder einer Datenbank, abzulegen und die Daten bei Bedarf wieder laden zu können. |
| Prototyp | Ein Prototyp stellt ein (abstraktes) Urbild für ein konkretes Objekt bereit. |
| RDF | Resource Description Framework ist eine standardisierte Sprache zur Beschreibung von Ressourcen mit Metadaten, siehe Abschnitt 2.1.1. |
| RDFS | → RDF Schema ermöglicht die Definition von Schemata zur Beschreibung der Struktur von RDF Beschreibungen. |
| Resource | Eine Ressource ist im Allgemeinen eine identifizierbare Sache. Siehe → RDF |
| Repository | Ein Repository ist ein System zur Speicherung von Objekten und insbesondere von Metadaten über beliebige Objekte (vgl. Eicker 2008). |
| Semantic Web | Semantic Web beschreibt ein Konzept, mit dem die Möglichkeit geschaffen werden soll, Daten mit einer Bedeutung zu annotieren, siehe Abschnitt 2.1. |
| Semantik | Semantik oder auch Bedeutungslehre bezieht sich auf die Bedeutung von Zeichen, hier von Daten. |
| Singleton | Singleton ist ein Entwurfsmuster, welches sicherstellt, dass von einer Klasse nur eine konkrete Ausprägung, ein Objekt, existieren kann. |
| Statement | Ein Statement ist eine Aussage. In → RDF beschreibt ein Statement als Tripel den Zusammenhang von Subjekt, Prädikat und Objekt. |
| SOA | Service-oriented architecture ist ein Architekturstil, bei dem verteilte Systeme und Komponenten (Dienste) Funktionalität anbieten und über Nachrichten miteinander kommunizieren. Grundlegende Ideen sind Wiederverwendbarkeit und Integration von Software, siehe Abschnitt 2.2. |
| SOAP | Simple Object Access Protocol (ehemalige Bezeichnung, heute als Name verwendet) dient als leichtgewichtiges Protokoll zum Austausch von Nachrichten. |
| SPARQL | Simple Protocol and RDF Query Language ist eine Anfragesprache für → RDF, ähnlich → SQL. |
| SQL | Structured Query Language ist eine Sprache zur Anfrage, Definition und Manipulation von Daten in (relationalen) Datenbanken. |

| | |
|---|---|
| Taxonomie | Taxonomien erlauben die hierarchische Ordnung von Begriffen, wie in Klassen und Unterklassen. → Klassifikation, mehr → Ontologie |
| URI | Uniform Resource Identifier ermöglicht die Identifizierung von abstrakten oder realen Ressourcen in einer standardisierten Weise. |
| W3C | World Wide Web Consortium bringt als Konsortium Standards oder Empfehlungen hervor, die das World Wide Web (WWW) betreffen. |
| Web 2.0 | Das Schlagwort Web 2.0 sammelt Technologien, die weitergehende Möglichkeiten zur Benutzerinteraktion mit einer Web-Anwendung bieten (vgl. Lackes und Siepermann 2008). |
| Web Service | Ein Web Service ist ein Softwaresystem, welches die Interoperabilität zwischen Maschinen über ein Netzwerk ermöglicht (Booth u. a. 2004), siehe Abschnitt 2.3. |
| WfMC | Workflow Management Coalition ist ein Zusammenschluss von Interessenten des Workflowmanagements, welche die Einführung eines Referenzmodells und einer Sprache für → Workflows zur Aufgabe hat. |
| WF | siehe → Workflow |
| Workflow | Ein Workflow ist eine abstrakte Definition eines oder mehrerer Teile eines Geschäftsprozesses, der automatisiert werden kann, siehe Abschnitt 2.4. |
| WF-Engine | Eine Workflow-Engine ermöglicht das Ausführen eines Workflows als → WF-Instanz. |
| WF-Instanz | Workflow-Instanz, siehe → Instanz |
| WSDL | Web Service Description Language ist eine Sprache zur Beschreibung von → Web Services. |
| XML | Extensible Markup Language ist eine standardisierte textbasierte Auszeichnungssprache für strukturierte Daten; unter anderem mit dem Ziel die Interoperabilität zu unterstützen. |
| XSD | XML Schema Definition ermöglicht die Definition der Struktur von → XML Dokumenten. |
| XSLT | Extensible Stylesheet Language (XSL) Transformation ist eine Sprache, die für die Übersetzung beziehungsweise Transformation zwischen verschiedenen XML Formaten genutzt werden kann. |

# Abbildungsverzeichnis

Abb. 2.1: Semantic Web Layers ..... 8
Abb. 2.2: Als Graph visualisiertes RDF Beispiel ..... 9
Abb. 2.3: Web Service Architektur ..... 14
Abb. 2.4: Beispiel einer Bestellung in EPK Notation ..... 17
Abb. 2.5: Fedora digital object model ..... 18
Abb. 2.6: Fedora digital object Beispiel ..... 19
Abb. 2.7: Fedora Ontologie für Beziehungen ..... 20
Abb. 3.1: Zusammenhang zwischen digitalen Informationsobjekten und Workflows .. 24
Abb. 3.2: Schema eines Workflows als informales Klassendiagramm ..... 28
Abb. 3.3: Prozess zur Veröffentlichung eines Dokuments im Szenario ..... 30
Abb. 3.4: Ontologie für Objekttypen ..... 34
Abb. 4.1: Architektur für Workflows und digitale Informationsobjekte ..... 38
Abb. 4.2: Aufbau von Muradora ..... 43
Abb. 5.1: Klassendiagramm des Pakets digitalobject ..... 47
Abb. 5.2: Aufbau eines digitalen Objektes im Szenario ..... 47
Abb. 5.3: Klassendiagramm des Pakets workflow ..... 50
Abb. 5.4: Sequenzdiagramm WorkflowAction ..... 51
Abb. 5.5: informelles Aktivitätsdiagramm zur Ausführung eines Workflows ..... 52
Abb. 5.6: Klassendiagramm Objekttypen ..... 55
Abb. 5.7: Schema für Rollen ..... 56
Abb. 5.8: Screenshot der Funktion Browse ..... 58
Abb. 5.9: Screenshot der Funktion view ..... 58
Abb. 5.10: Screenshot Start eines Workflows über den Objekttyp ..... 60
Abb. 5.11: Screenshot Workflow Schritt 1, Hochladen von Dokumenten ..... 60
Abb. 5.12: Screenshot Workflow Schritt 2, Eingabe von Metadaten ..... 61
Abb. 5.13: Screenshot der Funktion publish für Kuratoren ..... 61
Abb. 5.14: Screenshot Workflow Schritt 4, Setzen des Status ..... 62
Abb. 6.1: Prozess der Evaluierung ..... 63

# Tabellenverzeichnis

Tab. 6.1: Anforderungen an Workflows ..... 65
Tab. 6.2: Anforderungen an die Workflow-Engine ..... 67
Tab. 6.3: Anforderungen an digitale Objekte und Objekttypen ..... 69
Tab. 6.4: Anforderungen an Rechte und Rollen ..... 71
Tab. 6.5: Anforderungen an die Implementierung ..... 73
Tab. 6.6: Anforderungen an das Szenario ..... 73
Tab. 6.7: Anforderungen an Muradora ..... 78
Tab. 8.1: Checkliste der Anforderungen ..... 102
Tab. 8.2: Übersicht verwendeter Technologien ..... 108

# Listings

Listing 2.1: RDF Beispiel: Beschreibung eines Buches ..... 9
Listing 2.2: Triple der Beschreibung eines Buches ..... 10
Listing 2.3: RDFS Beispiel: Ein Buch ist ein Dokument ..... 10
Listing 3.1: Workflow zur Veröffentlichung eines Dokuments im Szenario ..... 33
Listing 5.1: RELS-EXT Datastream für Beziehungen zwischen digitalen Objekten ..... 48
Listing 5.2: Metadaten eines digitalen Objektes im Szenario ..... 49
Listing 5.3: Einbindung des Workflow-Systems in Struts2 ..... 53
Listing 8.1: XML Schema für Workflows ..... 105
Listing 8.2: Workflow zur Veröffentlichung eines Dokuments im Szenario ..... 107

# 1 Einleitung

In den vergangenen Jahren sind im Bereich der Integration von Software und der Integration von Daten neue Technologien entstanden, beziehungsweise haben sich Technologien als ein Standard etabliert. Die Integration von Software und Softwarekomponenten kann mit einem Blick auf service-orientierte Architekturen, zum Beispiel mit Web Services vorgenommen werden. Die Integration von Daten und auch die Möglichkeit Daten maschinen-verständlich zur Verfügung zu stellen, trifft auf Technologien des Semantic Web.

Die Prinzipien der service-orientierten Architektur, mit der bekannten Ausprägung Web Services, und des Semantic Web setzen sich zunehmend mit praktischer Anwendung auch in Unternehmen durch (Fensel u. a. 2006). Unternehmen und Organisationen speichern und verwalten umfangreiche Datenmengen, die erst mit ihrer Interpretation zu Informationen werden. Mit Hilfe der Technologien des Semantic Web können Daten ausführlicher semantisch, mit so genannten Metadaten, beschriebenen werden. Metadaten können, neben Beschreibungen der Daten selbst, auch weitere Informationen über den Kontext einzelner Daten liefern. Der Kontext eines Datums beschreibt beispielsweise Beziehungen zwischen einzelnen Daten.

Die Geschäftsprozesse eines Unternehmens oder einer Organisation verfolgen bestimmte geschäftliche Ziele. Geschäftsprozesse beinhalten Folgen von Tätigkeiten oder Aktivitäten, die zur Umsetzung eines Zieles erforderlich sind. Eine Folge von automatisierbaren Tätigkeiten wird als Workflow bezeichnet (Workflow Management Coalition 1995, S. 3).

Benutzer, die mit einem Informationssystem interagieren, verfolgen ein bestimmtes Ziel. Im Sinne eines Geschäftsprozesses bietet sich dem Benutzer eine Ausgangssituation von Daten und Informationen und ein Weg, wie er die Ziele der Organisation erreichen kann. Dieser Weg kann in Form von Workflows beschrieben werden.

Es existieren also digitale Informationsobjekte, welche die Daten oder weiter noch das Wissen einer Organisation beschreiben, und Geschäftsprozesse, denen Workflows zugrunde liegen. Dabei definieren Workflows den Ablauf von Tätigkeiten in einer Organisation. Es liegt somit nahe, Geschäftsprozesse und Daten mit Hilfe von Workflows zu integrieren.

In Bezug auf digitale Informationsobjekte, welche weitergehende Informationen besitzen können, stellen sich nun unterschiedliche Fragen zur Flexibilität von Workflows.

## 1.1 Szenario von Workflow und digitalen Informationsobjekten

Das Alfred-Wegener-Institut für Polar- und Meeresforschung[1] (kurz AWI) hat zum Ziel die Arktis, Antarktis und die Ozeane der mittleren und hohen Breiten zu erforschen. Weiterhin sind damit die Klima-, Bio- und Geosysteme der Erde ein wichtiges Forschungsgebiet.

Das AWI sammelt durch seine Forschung fortlaufend große Mengen neuer Daten und Informationen über verschiedenste wissenschaftliche Bereiche, die geeignet gespeichert und verwaltet werden müssen. Dabei sind nicht nur die Speicherung von großen Datenmengen sondern auch die Vernetzung und die Pflege von gesammelten Informationen und damit die Integration dieser Informationen von wesentlicher Bedeutung.

Zum Zwecke der Verwaltung dieser Daten führt das AWI in einem aktuellen Projekt das Fedora[2] (Flexible Extensible Digital Object Repository Architecture) Framework als Backend ein. Das Fedora Framework ermöglicht als Open Source Projekt eine kostengünstige Alternative zu kommerziellen Lösungen. Auf Basis der service-orientierten Architektur ermöglicht Fedora hohe Flexibilität und Erweiterbarkeit und damit die Anbindung an existierende und zukünftige Systeme. Neben der hoch skalierbaren Architektur von Fedora ist auch das verwendete Datenmodell höchst flexibel, welches sich als Vorteil zur Modellierung von Daten und Beziehungen gegenüber relativ festen Datenmodellen beweist. Neben diesen grundlegenden Eigenschaften wird Fedora bereits erfolgreich von zahlreichen Organisationen mit ähnlichem Interesse verwendet (Fedora Development Team 2008, Fedora Commons 2008a).

Für das Frontend setzt das AWI auf das Muradora[3] Framework. Muradora fügt sich an das Fedora Framework an und bietet als Web-Anwendung den Zugriff auf Fedora an. Weiterhin ermöglicht Muradora unter anderem die Authentifikation und Autorisation von Benutzern (Cover 2007) und damit die Beachtung von Rechten für Benutzer-aktionen.

Da das Muradora Framework die Bedienoberfläche für Benutzer bereitstellt und die Zugriffe auf Fedora und damit auch Zugriffe auf Datenbanksysteme und ähnliches kapselt, ist Muradora der Ansatzpunkt für die Betrachtung dieser Arbeit. Die Evaluierung für das Workflowmanagement zur Pflege von digitalen Informations-

[1] http://www.awi.de/ (2009-03-18)
[2] http://fedora-commons.org/ (2009-03-18)
[3] http://www.muradora.org/ (2008-03-18)

objekten bezieht sich dementsprechend auf das Muradora Framework mit Fedora als Grundlage.

Diese Arbeit verwendet ein relativ überschaubares Szenario, welches am AWI von Bedeutung ist. Neben den operativen Daten, die im AWI gesammelt und verarbeitet werden, entstehen wissenschaftliche Publikationen. Das hier verwendete Szenario soll im Bereich des Geschäftsprozesses den Ablauf für Veröffentlichungen am AWI darstellen. Veröffentlichungen selbst werden als digitale Informationsobjekte betrachtet, die mit Metadaten beschrieben werden und Verknüpfungen zu anderen digitalen Informationsobjekten aufweisen können.

In Abschnitt 3.5 wird das Szenario detaillierter vorgestellt, nachdem die Grundlagen in den vorherigen Kapiteln dafür geschaffen wurden. Das Szenario deckt auf der einen Seite die Verwendung von Workflows und auf der anderen Seite den Bereich von digitalen Informationsobjekten ab. Damit kann das Szenario als grundlegender Prototyp betrachtet werden und als Beispiel für andere Geschäftsbereiche des AWI, aber auch generell, über die Grenzen das AWI hinweg und für andere Organisationen, angesehen werden.

## 1.2 Fragestellungen

In dieser Arbeit sollen mit Blick auf die Anforderungen von Organisationen, im Speziellen von Forschungsinstituten, Erkenntnisse für die Beschreibung von Workflows für digitale Informationsobjekte gesammelt werden. So sind in Organisationen wie dem AWI möglichst einfache Beschreibungen von Workflows, sowie die Pflege von digitalen Objekten von Interesse.

Dabei ist diese Untersuchung keinem Selbstzweck zuzuschreiben, sondern bietet einen praktischen Nutzen. Wie im vorherigen Abschnitt 1.1 dargestellt wurde, besteht ein wesentliches Interesse an der Möglichkeit zur Einführung von Workflows für die Pflege von digitalen Informationsobjekten. Workflows sollen dabei Arbeitsabläufe formulieren und unterstützen, welche dem Tagesgeschäft dienen.

Zur Einführung von Workflows stellen sich mit Bezug auf den oben angesprochenen Rahmen dieser Arbeit eine Reihe von Fragen, die hier untersucht werden sollen. Diese Fragestellungen ergeben sich aus persönlichen Gesprächen mit Experten am AWI und beziehen sich auf deren jahrzehntelangen Erfahrungen. Die Fragestellungen lassen sich auch aus konzeptuellen Arbeiten ableiten. Aktuelle Arbeiten befassen sich beispielsweise mit Modellen zum „Curation Lifecycle“ (Higgins u. a. 2007, Treloar und Harboe-

Ree 2008). Es stellen sich in diesem Zusammenhang damit Fragen zur Definition und Verwendung von Workflows. Andere Arbeiten beschäftigen sich mit der Struktur von digitalen Objekten (Blekinge-Rasmussen und Fiedler Christiansen 2008).

In Bezug auf die in Abschnitt 1.1 beschriebene Anwendungsumgebung ergeben sich für diese Arbeit dadurch die folgenden Fragestellungen:

1. Wie können Workflows für digitale Informationsobjekte definiert werden?
   *Digitale Informationsobjekte können beliebige Daten kapseln und mit Metadaten beschrieben werden. Workflows definieren im Normalfall einen abstrakten Arbeitsablauf. Wie kann nun ein Workflow Eigenschaften eines digitalen Objektes beachten?*

2. Welchen Nutzen würden generische beziehungsweise dynamische Workflows für digitale Informationsobjekte erbringen?
   *Bei der Einführung von Workflows müssen bestimmte Anforderungen gegeben sein, welche den Aufwand für die Einführung und Wartung von Workflows rechtfertigen.*

3. Welche Informationen eines digitalen Informationsobjektes sind für den Ablauf eines Workflows interessant?
   *Ein digitales Informationsobjekt kann über Metadaten verfügen, welche für den Ablauf eines Workflows von Interesse sein können. Zum Beispiel kann ein digitales Informationsobjekt über einen Status verfügen, welcher im Workflow beachtet werden soll. Das Szenario in Abschnitt 3.5 verfügt über einen solchen Fall.*

4. Auf welche Weise können Informationen über Benutzerrechte in Workflows eingebunden werden?
   *Workflows können an bestimmte Rechte gebunden sein. So kann es angebracht sein, dass ein Workflow oder auch einzelne Schritte eines Workflows nur von bestimmten Benutzern ausgeführt werden dürfen.*

5. Wie können Beziehungen von digitalen Informationsobjekten untereinander dargestellt, aufgelöst und für Workflows benutzt werden?
   *Besitzen digitale Informationsobjekte Beziehungen zu anderen Objekten, so müssen auch diese Beziehungen gepflegt werden können.*

## 1.3 Ziele der Diplomarbeit

Die genannten Fragestellungen sollen anhand des Fedora Frameworks und des Muradora Framework untersucht werden.

Als Ergebnisse der Untersuchung der Fragestellungen sollen Probleme und Möglichkeiten aufgezeigt werden, die den Einsatz von Workflows betreffen. Dabei sollen die Erkenntnisse dieser Arbeit als Grundlage und Empfehlung für weitere Arbeiten im Bereich von digitalen Informationsobjekten und Workflows, im Speziellen für Institute wie dem AWI, dienen.

Soweit es im vertretbaren Rahmen dieser Arbeit möglich ist, sollen die Ergebnisse mit einer prototypischen Entwicklung und Implementierung dargestellt werden. Dabei beziehen sich diese Teile auf ein Szenario im Rahmen der Pflege von Publikationen.

Kein Ziel dieser Arbeit ist die Verwendung oder Entwicklung von Workflows für semantische Web Services, wie sie beispielsweise in Fensel u. a. 2006 besprochen werden.

## 1.4 Aufbau der Arbeit

Der zurückliegende Abschnitt umfasste die Motivation für diese Arbeit, sowie die Anforderungen und Ziele der Arbeit. Die Anforderungen werden beginnend mit dem dritten Kapitel weiter konkretisiert.

Abschnitt zwei nennt Grundlagen, die für diese Arbeit essentiell sind. Dabei werden die Grundlagen erläutert und es wird auf weiterführende Literatur verwiesen.

Aufbauend auf den Grundlagen aus dem zweiten Kapitel, wird im dritten Abschnitt eine Möglichkeit entwickelt, wie digitale Informationsobjekte mit Workflows integriert werden können. In diesem Teil wird außerdem das Szenario für diese Arbeit vorgestellt.

Der vierte Teil stellt eine Architektur vor, welche die zuvor dargestellten Anforderungen und Ansätze zur Integration von digitalen Objekten mit Workflows zusammenbringt. Im Anschluss daran wird im fünften Kapitel der Entwurf und die Umsetzung eines Prototypen, basierend auf der Architektur des vierten Kapitels, beschrieben.

Im sechsten Kapitel werden die erhobenen Anforderungen aus den vorherigen Kapiteln zusammengetragen und mit den Ergebnissen dieser Arbeit verglichen und damit evaluiert. Die Ergebnisse dieser Evaluierung werden dann im Hinblick auf ihren praktischen Bezug diskutiert.

Abschließend gibt das siebte Kapitel eine kurze Zusammenfassung dieser Arbeit und ihrer Ergebnisse sowie einen Ausblick für weitere Fragestellungen.

# 2 Heranführung und Grundlagen

Die Zusammenführung oder auch Integration von Workflows und digitalen Informationsobjekten erfordert die Auseinandersetzung mit den zu Grunde liegenden Technologien.

Daten und Informationen können mit Hilfe von digitalen Informationsobjekten verwaltet werden. Dazu erfordern digitale Informationsobjekte eine semantische Beschreibung der mit ihnen verwalteten Daten, um so mit Metadaten einen Mehrwert gegenüber einer einfachen Datenhaltung zu erbringen. Die Technologien des Semantic Web (siehe Abschnitt 2.1) ermöglichen unter anderem eine standardisierte Beschreibung von Metadaten. Arbeiten wie Hitzler u. a. 2008 und Allemang und Hendler 2008 beschäftigen sich mit dem Semantic Web. Lagoze u. a. 2006 zeigt die Verwendung von Technologien des Semantic Web für digitale Objekte.

Die Integration von Softwaresystemen, das heißt auch die Zusammensetzung verschiedener Systeme, kann durch eine service-orientierte Architektur (SOA) erleichtert werden. Dadurch werden auch Konzepte wie beispielsweise lose Kopplung und Wiederverwendbarkeit gefördert. Der praktische Teil dieser Arbeit orientiert sich an den Grundsätzen von SOA um eben die Vorteile dieser Konzepte nutzen zu können. Des Weiteren setzt das Fedora Framework auf SOA und verwendet dazu Web Services. Abschnitt 2.2 gibt eine kurze Einführung in SOA und Abschnitt 2.3 einen Einblick in Web Services, welche eine SOA umsetzen können (Krafzig u. a. 2004, Erl 2005, Masak 2007).

Schließlich wird für die Arbeit ein Verständnis des Begriffs Workflow benötigt. Der Abschnitt 2.4 liefert hierzu einen Einblick in das Thema und zeigt die Verwendung von ereignisgesteuerten Prozessketten (EPK) zur visuellen Darstellung von Workflows (Workflow Management Coalition 1995, Keller u. a. 1992).

Bei der Integration von Workflows und digitalen Informationsobjekten ist offensichtlich auch die Struktur eines digitalen Informationsobjektes von Bedeutung. So wird in Abschnitt 2.5 ein Objektmodell für digitale Informationsobjekte vorgestellt.

## 2.1 Semantic Web

Der Begriff des Semantic Web beschreibt ein Konzept, mit dem die Möglichkeit geschaffen werden soll, Daten mit einer Bedeutung zu annotieren. Dabei sollen nicht nur Menschen erkennen können, was ein bestimmtes Datum darstellt, sondern auch Maschinen sollen ein Datum mit anderen Daten in Zusammenhang bringen können.

Entsprechend sind Standards das Ziel des Semantic Web. Standards sollen dabei die Interoperabilität zwischen Systemen, die Flexibilität und die Erweiterbarkeit von Systemen begünstigen. Sprachen wie das Resource Description Framework (RDF) und die Ontology Web Language (OWL) ermöglichen die semantische Beschreibung von Daten (Hitzler u. a. 2008).

In Verbindung mit dem Semantic Web steht auch das bereits genannte Schlagwort Ontologie. Eine Kurzfassung des Begriffs Ontologie liefert Gruber 1993 mit:

„An ontology is a specification of a conceptualization."

Unter einer Ontologie kann also eine formale Spezifikation verstanden werden, die Konzepte und Beziehungen zwischen Konzepten in Zusammenhang bringt. Dabei bezieht sich eine Ontologie in der Regel auf eine Anwendungsdomäne. Im Gegensatz zu einer einfachen Taxonomie oder einer Klassifikation repräsentiert eine Ontologie einen bestimmten Weltausschnitt, der nicht nur hierarchisch aufgebaut sein muss (Gruber 1993).

Das World Wide Web Consortium (W3C) stellt mit der Semantic Web Activity (Koivunen und Eric Miller 2002) einen Semantic Web Layer vor, der den Aufbau des Semantic Web erkennen lässt. Abb. 2.1 zeigt den Semantic Web Layer. Als Grundlagen dienen Zeichenkodierung Unicode und der Uniform Resource Identifier (URI[4]). Die Extensible Markup Language (XML, W3C 2003) mit Namespaces und der Erweiterung XML Schema bilden die zweite Ebene (Layer). Darauf aufbauend werden RDF und RDF Schema, und Ontologie Vokabular definiert. Die Ebenen Trust, Proof und Logic sind bisher allerdings noch nicht vollständig entwickelt.

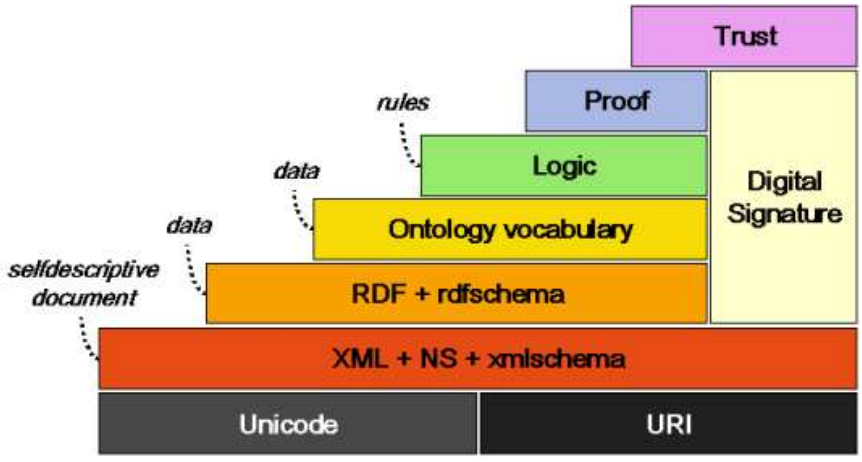


**Abb. 2.1: Semantic Web Layers[5]**

[4] Berners-Lee 1994, Berners-Lee u. a. 2005
[5] Quelle: http://www.w3.org/2001/12/semweb-fin/w3csw (2009-03-18)

Eine Einführung zum Semantic Web wird beispielsweise in Hitzler u. a. 2008 und Allemang und James A. Hendler 2008 geboten. Im Folgenden werden RDF und OWL näher vorgestellt, da sie die Grundlagen für digitale Objekte, und deren Beziehungen untereinander, bilden.

### 2.1.1 Resource Description Framework (RDF)

Das Resource Description Framework stellt eine formale Sprache zur Beschreibung von strukturierten Informationen bereit. Diese Informationen können als Statements formuliert und als gerichtete Graphen veranschaulicht werden. Die einzelnen Teile eines Statements werden dabei eindeutig durch eine URI identifiziert (Manola und Erik Miller 2004).

Ein Statement besteht aus dem zu beschreibenden Objekt (Resource), aus einer Eigenschaft (Property) und einem zugeordneten Wert. Dieser Aufbau gleicht der Grammatik der natürlichen Sprachen mit Subjekt, Prädikat und Objekt.

Im Folgenden wird ein Beispiel für eine Beschreibung von Informationen in RDF gegeben. Dieses Beispiel verwendet dabei die Konventionen des Dublin Core[6] zur Beschreibung von Dokumenten und anderen Objekten.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.example.com/books/12345">
    <dc:title>Beispiel Buch 12345</dc:title>
    <dc:creator>Roland Koppe</dc:creator>
    <dc:date>2008-10-11</dc:date>
  </rdf:Description>
</rdf:RDF>
```

**Listing 2.1: RDF Beispiel: Beschreibung eines Buches**

Abb. 2.2 zeigt das obige Beispiel in Listing 2.1 als Graph visualisiert.

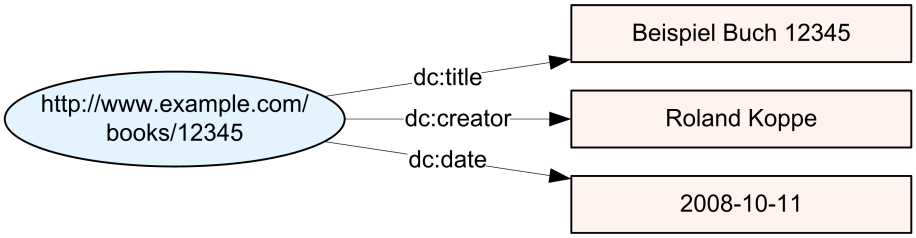


**Abb. 2.2: Als Graph visualisiertes RDF Beispiel[7]**

[6] http://dublincore.org/ (2009-03-18)
[7] Quelle: eigene

Statements können in Form von Triples mit Hilfe von Triple Stores verwaltet werden. Solche Triple Stores ermöglichen über SQL-ähnliche Sprachen, wie beispielsweise SPARQL (Prud'hommeaux und Seaborne 2008), die Anfrage von in RDF beschriebenen Informationen.

Listing 2.2 zeigt die zum obigen Beispiel gehörigen Triples.

```
<http://www.example.com/books/12345> <dc:title> <Beispiel Buch 12345>
<http://www.example.com/books/12345> <dc:creator> <Roland Koppe>
<http://www.example.com/books/12345> <dc:date> <2008-10-11>
```

**Listing 2.2: Triple der Beschreibung eines Buches**

Folglich beschreibt RDF im Wesentlichen Aussagen oder Beziehungen zwischen Metadaten. Mit RDF Schema (RDFS) wird die Möglichkeit gegeben, auch die Struktur dieser Metadaten gezielter zu definieren. Zu diesen Möglichkeiten zählen Konzepte für Klassen, Eigenschaften, Vererbung und Einschränkungen.

Ein Beispiel für eine RDFS Beschreibung zeigt das Listing 2.3. Hier wird ein Buch als Ableitung eines Dokuments gesehen. Die beiden Objekte Dokument und Buch sind dabei jeweils als Klassen definiert.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdf:Description rdf:ID="Dokument">
    <rdf:type
     rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
  </rdf:Description>

  <rdf:Description rdf:ID="Buch">
    <rdf:type
     rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
    <rdfs:subClassOf rdf:resource="#Dokument" />
  </rdf:Description>
</rdf:RDF>
```

**Listing 2.3: RDFS Beispiel: Ein Buch ist ein Dokument**

Eine kurze Einführung mit Beispielen in RDF und RDFS bietet W3Schools[8], einen vertiefenden Einblick in RDF liefert das W3C (Manola und Erik Miller 2004).

Mit dieser kurzen Einführung in RDF ist nun die Grundlage für weiterführende Technologien gegeben. Der folgende Abschnitt benennt die Ontology Web Language auf Basis von RDF. Abschnitt 2.5 befasst sich mit digitalen Informationsobjekten wobei hier mit Hilfe von RDF Beziehungen ausgedrückt werden.

[8] http://www.w3schools.com/rdf/ (2009-03-18)

### 2.1.2 Ontology Web Language (OWL)

Die Ontology Web Language (OWL) kann als eine Erweiterung des zuvor vorgestellten RDF aufgefasst werden. Sie bietet weitergehende Konzepte zur formalen Beschreibung von Ontologien und reduziert dabei die Generizität von RDF, um die maschinelle Interpretierbarkeit zu verbessern (Dean und Schreiber 2004).

OWL erweitert RDF um

- Einschränkungen, wie beispielsweise die des Werte- und Definitionsbereich und Kardinalitäten,
- mengenlogische Operatoren wie Vereinigung, Durchschnitt, Komplement,
- Symmetrie und Transitivität von Eigenschaften und
- Definitionen wie etwa Äquivalenz und Inversen.

OWL teilt sich in drei verschiedene Versionen auf: OWL Lite, OWL DL und OWL Full. OWL Lite ermöglicht bereits die Modellierung von Taxonomien und Ontologien. Dabei kann OWL Lite wegen seiner geringen Ausdrucksstärke effizienter als die beiden anderen verarbeitet werden. OWL DL unterstützt die Beschreibung von Ontologien mit einer entscheidbaren[9] Untermenge der Prädikatenlogik ersten Grades[10]. Des Weiteren bietet OWL Full das volle Spektrum von OWL und hebt Einschränkungen von OWL Lite und OWL DL auf, dadurch wird OWL Full allerdings unentscheidbar.

Die Abbildung von Konzepten und Beziehungen einer Anwendungsdomäne, auch als Weltausschnitt bezeichnet, kann mit OWL vorgenommen werden. Im weiteren Verlauf der Arbeit (vgl. Kapitel 3) wird von Objekttypen die Rede sein, welche durch eine Ontologie näher beschrieben werden können, um damit ihre Semantik darzustellen.

## 2.2 Service-orientierte Architektur

Das Konzept der service-orientierten Architektur definiert Grundprinzipien, die zeigen, auf welche Weise Softwarekomponenten miteinander verbunden werden können. Ein wichtiger Aspekt dabei ist die Integration verschiedener Softwaresysteme. Die

[9] Gibt es zu einem Problem einen Algorithmus, der bei einer beliebigen Eingabe irgendwann anhält und das Problem mit „ja“ oder „nein“ beantworten kann, so wird das Problem als entscheidbar beschrieben, sonst unentscheidbar (Wimmel 2008, S. 3).

[10] Die Prädikatenlogik ist eine Erweiterung der Aussagenlogik und ermöglicht unter anderem das Ausdrücken von Eigenschaften durch „für alle Objekte gilt ...“ oder „es gibt ein Objekt, für das gilt ...“. Siehe auch Rechenberg und Pomberger 2006, S. 52f.

Verbindung der einzelnen Softwarekomponenten wird hierbei durch Dienste (Services) vorgenommen.

Krafzig u. a. 2004, S. 57 geben dazu die folgende Definition einer service-orientierten Architektur:

> „A Service-Oriented Architecture (SOA) is a software architecture that is based on key concepts of an application frontend, service, service repository, and service bus. A service consists of a contract, one or more interfaces, and an implementation."

Service-orientierte Architekturen basieren fundamental auf einigen Grundprinzipien, die bei der Entwicklung beziehungsweise Einführung von SOA beachtet werden sollten. Nach Erl 2005, S. 37 sind diese

- Lose Kopplung: Dienste besitzen untereinander nur minimale Abhängigkeiten. Diensten genügt ein „Wissen" darüber, dass auch noch andere Dienste existieren.

- Dienstvertrag: Ein Dienst verfügt über Beschreibungen zur Verwendung des Dienstes. Solche Beschreibungen können als Kommunikationsvertrag zwischen Diensten betrachtet werden.

- Autonomie: Dienste besitzen die alleinige Kontrolle über die ihnen zugrunde liegende Logik. Anfragen an einen Dienst können also nur über den Dienst Vertrag vorgenommen werden.

- Abstraktion: Die tatsächliche Umsetzung eines Dienstes wird nach außen hin verborgen. Es wird nur der oben genannte Dienst Vertrag veröffentlicht.

- Wiederverwendbarkeit: Um Dienste mehrfach verwenden zu können kann die Logik, die zur Erfüllung einzelner Aufgaben verwendet wird, auf mehrere Dienste aufgeteilt werden.

- Komponierbarkeit: Die Komponierbarkeit von Diensten ermöglicht es, über Koordination und Zusammenfassung von Diensten neue Dienste zur Verfügung zu stellen. Dienste können also auf mehreren anderen Diensten basieren.

- Zustandslosigkeit: Informationen über den Zustand eines Dienstes sollen nicht im Dienst selbst gespeichert sein; die Zustandsinformationen für eine bestimmte Aktivität sollen minimiert werden.

- Auffindbarkeit: Dienste werden nach außen hin so beschrieben, dass sie gefunden und bewertet werden können.

Schließlich sollen die genannten Grundprinzipien es unter Anderem ermöglichen, Software wieder verwenden zu können, die Integration von Software zu erleichtern und damit den Entwicklungsaufwand zu begrenzen.

Diese Prinzipien sollen sich auch im praktischen Teil dieser Arbeit wieder finden. Weiterhin liegt dem Fedora Framework eine service-orientierte Architektur zu Grunde, die mit Hilfe von Web Services umgesetzt wird. Aus diesem Grunde ist ein Einblick in diese Technologien für das Verständnis der vorliegenden Arbeit von Vorteil.

## 2.3 Web Services

Web Services stellen eine Möglichkeit zur Umsetzung der zuvor beschriebenen service-orientierten Architektur dar. Dabei bietet ein Web Service einen bestimmten Dienst in einem Netzwerk an.

Das World Wide Web Consortium (W3C) definiert einen Web Service als Softwaresystem, welches die Interoperabilität zwischen Maschinen über ein Netzwerk ermöglicht. Ein Web Services besitzt dazu eine Beschreibung seiner Schnittstellen in einem maschinenlesbaren Format. Systeme können mit einem Web Service interagieren, indem sie Nachrichten mit dem Web Service austauschen (Booth u. a. 2004).

Weiterhin definiert das W3C Technologien als Standards für Web Services.

- Die Extended Markup Language (XML) dient grundlegend als Sprache zur Beschreibung von Dokumenten.
- Das Simple Object Access Protocol (SOAP, vgl. Box u. a. 2000) ermöglicht als leichtgewichtiges Protokoll auf Basis von XML den Austausch von Nachrichten.
- Zur Beschreibung der Schnittstellen eines Web Services wird die Web Service Description Language (WSDL, vgl. Chinnici u. a. 2007) verwendet.

Um Web Services nutzen zu können, müssen diese zum einem von einem Anbieter zur Verfügung gestellt werden und zum anderen auffindbar sein. Das Auffinden von Web Services kann durch Verzeichnisdienste, ähnlich der „Gelben Seiten“ ermöglicht werden.

Die Organization for the Advancement of Structured Information Standards (OASIS) beschreibt UDDI (Universal Description Discovery & Integration) in Clement u. a.

2004 als einen plattformunabhängigen Verzeichnisdienst für Web Services und andere elektronische als auch nicht elektronische Dienste.

Insgesamt wird damit eine Web Service Architektur beschrieben, die den Ansprüchen der service-orientierten Architektur genügen kann. Abb. 2.3 zeigt diese Architektur mit den drei Akteuren Anbieter (Web Service Server), Konsument (Web Service Client) und Verzeichnis (UDDI Server).

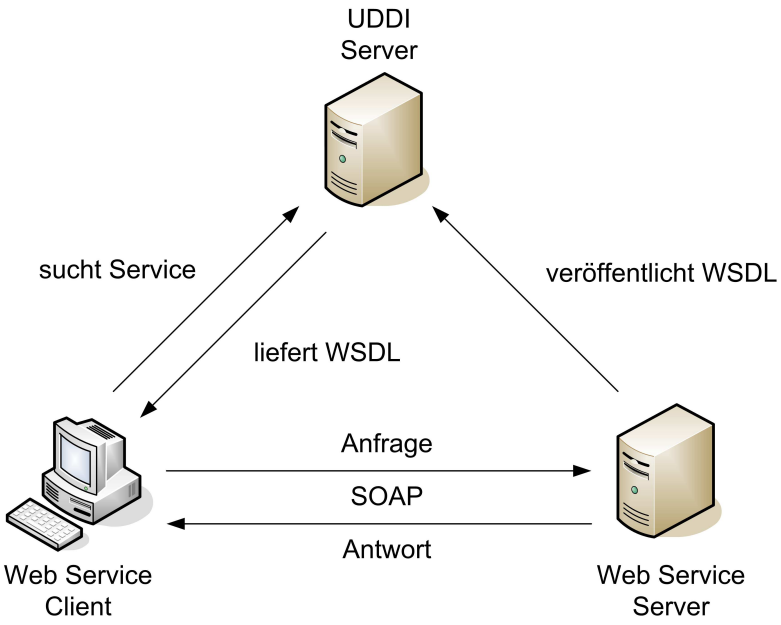


**Abb. 2.3: Web Service Architektur**[11]

Anbieter von Web Services bieten Dienste auf ihrem Web Service Server an. Damit diese Dienste für Konsumenten auffindbar sind, wird eine Beschreibung des Dienstes als WSDL veröffentlicht. Konsumenten können nun in diesem UDDI Verzeichnis nach Diensten suchen und erhalten die Beschreibung eines Services. Schließlich kann die Kommunikation zwischen Konsument (Client) und Anbieter (Server) erfolgen.

## 2.4 Workflows

Diese Arbeit befasst sich mit der Definition und Verwendung von Workflows. Daher gibt der folgende Abschnitt einen Einblick in die Thematik.

Es gibt zahlreiche verschiedene Definitionen für den Begriff Workflow. Das Wort Workflow findet bei einfachen Abläufen bis hin zum Business Process Management Verwendung (vgl. Krafzig u. a. 2004, S. 103 ff).

[11] Quelle: eigene, in Anlehnung an Erl 2005, S. 75

Die Workflow Management Coalition (WfMC), welche die Standardisierung von Workflowmanagement-Systemen zum Ziel hat, definiert einen Workflow (Workflow Management Coalition 1999, S. 8) als

> „The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules."

Ein Workflow dient demnach der Automatisierung eines Geschäftsprozesses. Ein Geschäftsprozess (business process) wiederum besteht aus einer oder mehreren miteinander verknüpften Vorgängen beziehungsweise Aktivitäten, welche zusammen ein geschäftliches Ziel verfolgen. Geschäftsprozesse werden im Kontext einer Organisationsstruktur betrachtet, welche funktionale Rollen und Beziehungen definiert (Workflow Management Coalition 1999, S. 10).

Das Workflowmanagement befasst sich dem Wort nach mit dem Management von Workflows. Dabei bezeichnet Management im Allgemeinen das Organisieren, Planen, Entscheiden, Kontrollieren, Steuern und Führen (Jablonski u. a. 1997, S. 491). Diese Aufgaben sollen durch Workflowmanagement-Systeme unterstützt beziehungsweise ausgeführt werden.

Workflowmanagement-Systeme übernehmen die Aufgabe der Automatisierung von Workflows in einem Geschäftsprozess. Dabei bezeichnen Workflows automatisierbare Aktivitäten eines Geschäftprozesses. Ein Geschäftsprozess selbst kann durchaus weitere manuelle Aktivitäten beinhalten, die nicht durch einen Workflow dargestellt werden.

Schließlich beschäftigt sich das Business Process Management mit der Struktur von Geschäftsprozessen innerhalb einer oder mehrerer Organisationen.

In dieser Arbeit wird ein Workflow als ein automatisierbarer Teil eines Geschäfts-prozesses betrachtet, der eine Folge von Aktivitäten zusammenfasst, um ein bestimmtes Ziel zu erreichen. Ein Workflow betrachtet hier die Seite der Informationstechnologie und nicht die betriebswirtschaftliche Seite.

Für die Definition und Beschreibung von Workflows existieren verschiedene Sprachen, wie beispielsweise BPEL (Business Process Execution Language), ebXML (Electronic Business using XML) oder YAWL (Yet Another Workflow Language). Ebenso vielfältig stehen Workflow Engines zur Verfügung, die es erlauben Workflows auszuführen.

Allen Workflow Sprachen gemein sind bestimmte Muster von Aktivitäten, die mit diesen Sprachen ausgedrückt werden können. In Russell u. a. 2006 werden Muster für Folgen von Aktivitäten vorgestellt und existierende Sprachen auf ihre Fähigkeiten zur Beschreibung von Workflows hin bewertet.

Als grundlegende Muster des Kontrollflusses eines Workflows können genannt werden (vgl. Workflow Management Coalition 1999):

- Sequentielle Ausführung (sequence),
- Parallele Ausführung (parallel split, AND-split),
- Synchronisation (synchronisation, AND-join),
- exklusive Ausführung (exclusive choice, XOR-split) und
- einfaches Zusammenführen (simple merge, XOR-join).

Solche Muster lassen sich in der im folgenden Abschnitt vorgestellten EPK-Notation wieder finden. Weitere Informationen zu Mustern bietet die Workflow Patterns Initiative[12] an. Neben den genannten Mustern sind dort weitere Muster aus Sicht von Daten (data perspective), Ressourcen (resource perspective) und Ausnahmebehandlungen (exception handling perspective) beschrieben. In „Workflow Control-Flow Patterns“ werden weitere Muster detailliert dargestellt (Russell u. a. 2006, Russell u. a. 2005).

### 2.4.1 Ereignisgesteuerte Prozesskette

Im Folgenden wird kurz der Begriff der ereignisgesteuerten Prozesskette (EPK) eingeführt. Mit Hilfe von EPK können Prozesse und die damit zusammenhängenden Abläufe in einer grafischen Notation dargestellt werden (Keller u. a. 1992).

Die Verwendung von EPK ermöglicht die relativ übersichtliche Darstellung von Workflows für den menschlichen Betrachter. Für eine konkrete Modellierung und Implementierung von Workflows mit dem Ziel der automatisierten Ausführung werden Workflow Sprachen verwendet (Krallmann und Trier 2008).

[12] http://www.workflowpatterns.com (2009-03-18)

Abb. 2.4 zeigt ein Beispiel für eine Bestellung in der EPK Notation.

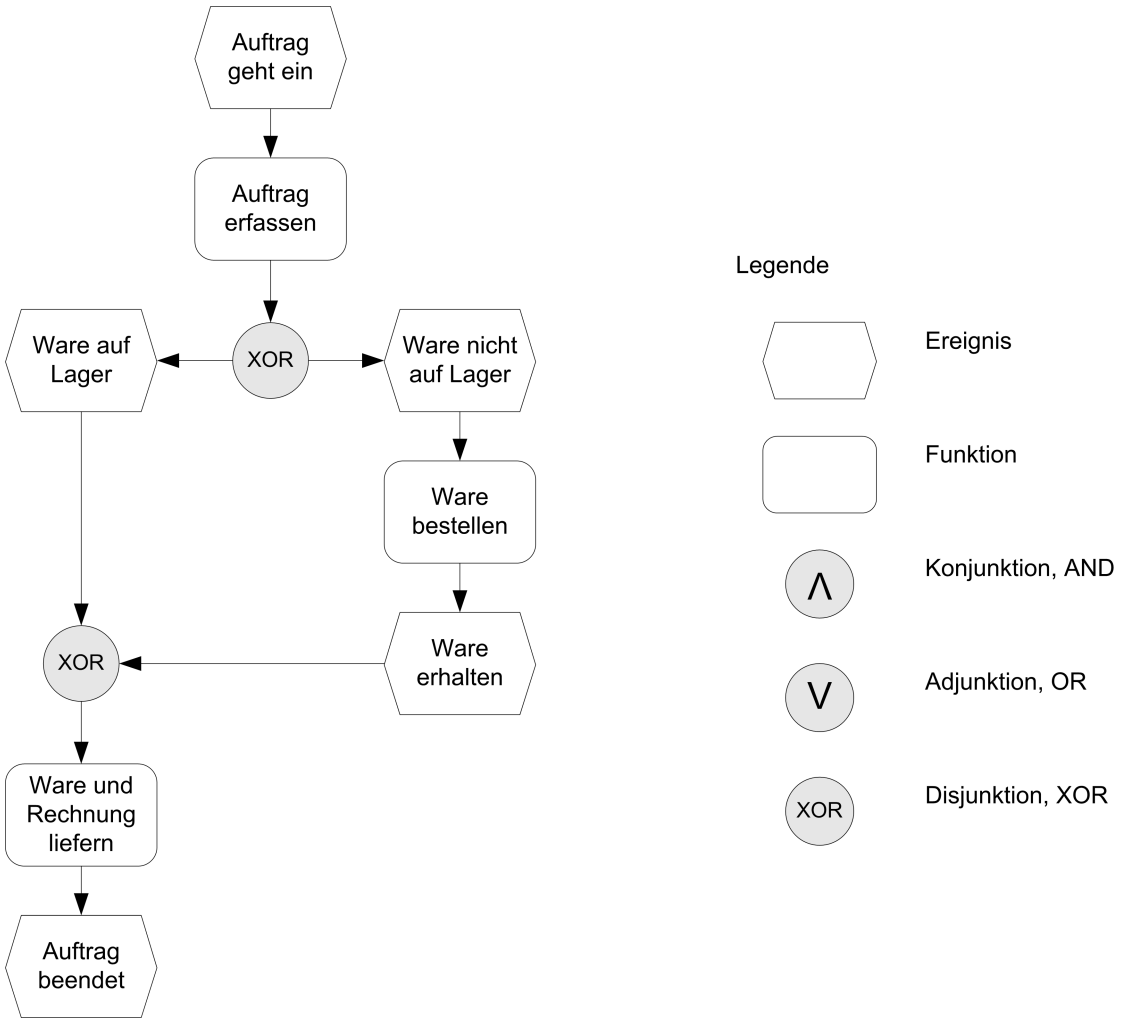


**Abb. 2.4: Beispiel einer Bestellung in EPK Notation**[13]

Die Modellierung von EPK erfolgt über vier Konstrukte:

- Funktionen sind aktive Komponenten, welche bestimmte Aufgaben übernehmen und damit Eingaben zu Ausgaben transformieren,
- Ereignisse sind passive Komponenten, welche Zustände beschreiben, die im Verlauf eines Prozesses auftreten und Auswirkungen auf den Ablauf eines Prozesses haben,
- Verknüpfungsoperatoren (Konjunktion „und“, Disjunktion „entweder oder“ und Adjunktion „und / oder“) zwischen Ereignissen und Funktionen ermöglichen die zusammenhängende Darstellung eines Prozesses und
- Kanten zeigen die Ablaufrichtung eines Prozesses auf.

[13] Quelle: eigene

## 2.5 Digitale Informationsobjekte

Für die Verwaltung von digitalen Informationsobjekten wird in dieser Arbeit das bereits angesprochene Fedora Framework betrachtet. Fedora bietet als Basis ein Modell für digitale Informationsobjekte. Dieses Modell ermöglicht die Aggregation von Daten verschiedenen Typs, die Angabe von Beziehungen von Objekten und auch die Assoziation von Diensten (Web Services) zu bestimmten Daten (Lagoze u. a. 2006).

Das Fedora Framework wird stetig weiterentwickelt. In dieser Arbeit kommt die Version 2.2.3 zur Anwendung. Die Festlegung auf diese Version liegt in der Arbeitsumgebung des Szenarios begründet, welche eben diese Version erfordert. So verwendet Fedora 2.2.3 ein anderes Objektmodell als Fedora 3.

Abb. 2.5 stellt den Aufbau eines digitalen Objektes in Fedora grafisch dar.

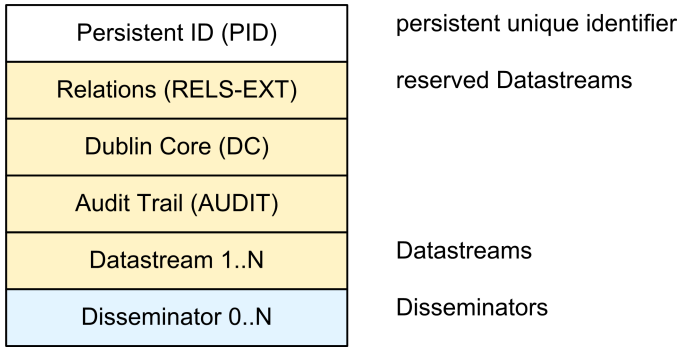


**Abb. 2.5: Fedora digital object model**[14]

Ein Fedora digital object setzt sich aus vier Komponenten zusammen: einer PID (persistent unique identifier), Objekteigenschafen, Datastreams und Disseminators. Die Beschreibung eines digitalen Objekts erfolgt mit Hilfe der auf XML basierenden Sprache Fedora Object XML (FOXML) (Fedora Commons 2008c, Fedora Commons 2008d).

- Die PID und die Objekteigenschaften dienen der Identifikation und Verwaltung eines digitalen Objektes im Fedora Repository.

- Ein digitales Objekt besitzt mindestens einen Datastream. Datastreams verweisen auf beliebige digitale Ressourcen, die mit dem digitalen Objekt assoziiert werden. Datastreams können innerhalb eines digitalen Objektes (Internal XML Metadata), im Fedora Repository (Managed Content) oder außerhalb von Fedora (External / Redirect Referenced Content) über die Angabe einer URL gespeichert werden.

[14] Quelle: eigene, in Anlehung an Fedora Commons 2008c

- Disseminators sind für ein digitales Objekt optional. Sie ermöglichen die Angabe von externen Diensten, die mit dem digitalen Objekt oder mit einzelnen Datastreams verwendet werden können. Somit kann eine erweiterte Funktionalität bereitgestellt werden.

Fedora unterscheidet drei Typen von digitalen Objekten. Data Objects dienen der Verwaltung von Daten, Behavior Definition Objects stellen die Schnittstellenbeschreibung (Interface) für Disseminators bereit, welche durch ein Behavior Mechanism Object implementiert werden. Dabei verweist ein Behavoir Mechanism Object auf einen konkreten Dienst (Web Service), der mit WSDL beschrieben wird.

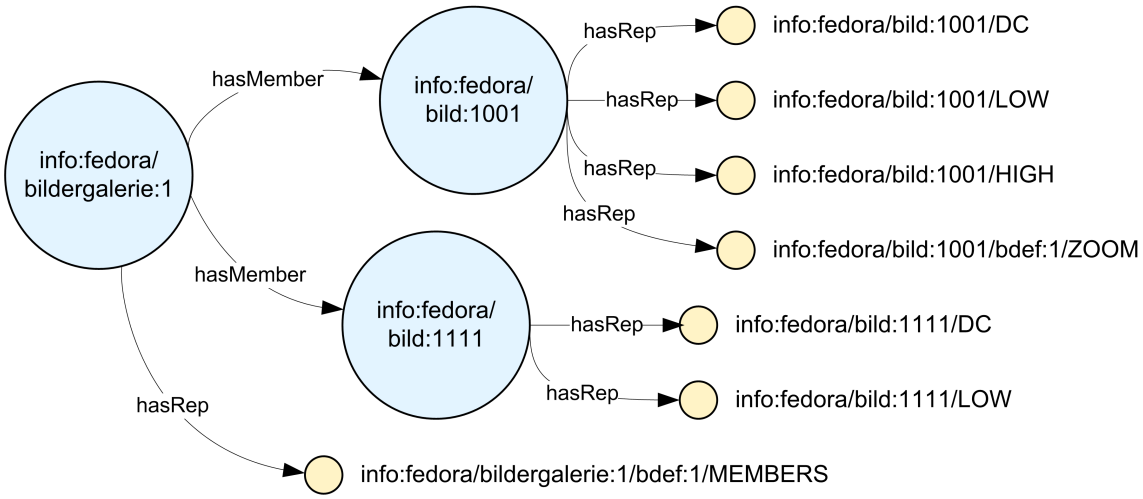


**Abb. 2.6: Fedora digital object Beispiel**[15]

Ein Beispiel für digitale Objekte zeigt Abb. 2.6. Dabei kann dieses Beispiel als Darstellung einer Bildergalerie, zu der einzelne Bilder gehören, betrachtet werden. Hier stellt das digitale Objekt `info:fedora/bildergalerie:1` eine Sammlung weiterer digitaler Objekte dar, die mit der Relation `hasMember` definiert werden. Das digitale Objekt `info:fedora/bild:1001` besitzt vier Repräsentationen. Die ersten drei (DC für Dublin Core, LOW für Vorschau und HIGH für hohe Auflösung) verweisen auf Datastreams und stellen somit konkrete Daten dar. Die vierte Repräsentation verweist auf einen Disseminator ZOOM (einen Web Service), welcher das Zoomen einer Repräsentation ermöglicht.

Wie das Beispiel zeigt, können digitale Objekte grafisch dargestellt werden. Der Aufbau dieser Darstellung erinnert an die in Abschnitt 2.1.1 angesprochene Verwendung von RDF, die in Fedora tatsächlich auch zur Speicherung von digitalen Objekten und ihren Beziehungen Verwendung findet.

[15] Quelle: eigene, nach Lagoze u. a. 2005

Zur Darstellung von Beziehungen zwischen digitalen Objekten bietet Fedora ein grundlegendes Modell an, welches im Fedora-Kontext als Ontologie bezeichnet wird. Dazu gehört auch die bereits in Abb. 2.6 gezeigte Relation `hasMember`. Eine Übersicht der zur Verfügung stehenden Beziehungen kann Abb. 2.7 entnommen werden (Fedora Commons 2008b).

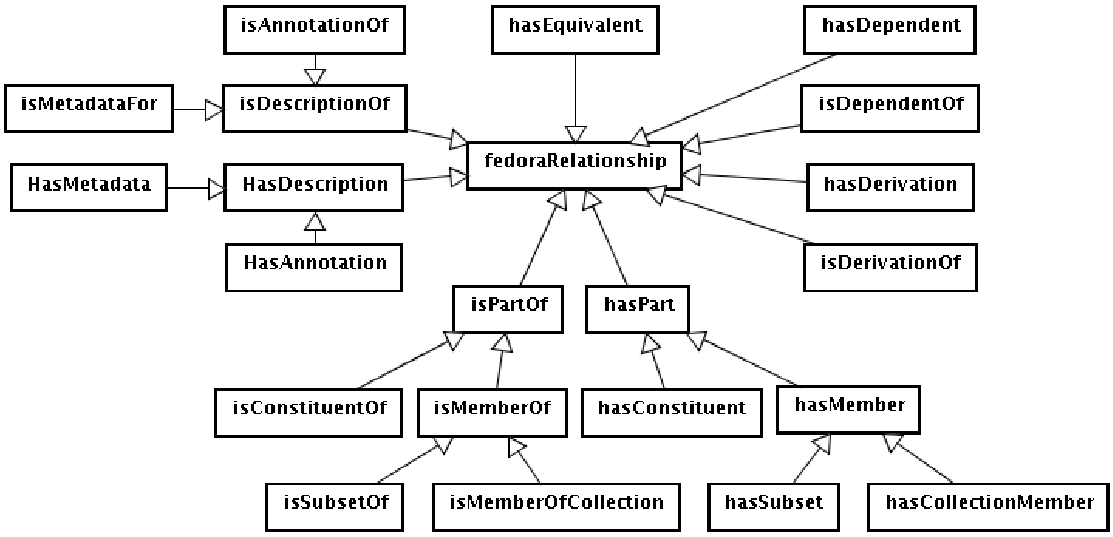


**Abb. 2.7: Fedora Ontologie für Beziehungen**[16]

Der Zugriff auf digitale Informationsobjekte wird mit Fedora durch die Bereitstellung eines Web Service ermöglicht. Fedora bietet dazu verschiedene Schnittstellen an, die je nach Bedarf zum Einsatz kommen können (Fedora Commons 2008e).

- Der Fedora Access Service (API-A) erlaubt die Anfrage von digitalen Objekten inklusive der Anfrage möglicher Disseminators über SOAP.

- Der Fedora Access Service (API-A-Lite) bietet eine geringere Funktionalität als die API-A, erlaubt dafür aber die Anfrage über den Representational State Transfer (REST, Fielding 2000) Architekturstil.

- Der Fedora Management Service (API-M) dient der Bearbeitung von digitalen Objekten, das heißt es werden Methoden zum Erstellen, Editieren und Löschen von digitalen Objekten über SOAP angeboten.

- Der Fedora Management Service (API-M-Lite) soll API-M über REST ermöglichen, bietet aber im Gegensatz zu API-M kaum Funktionalität.

[16] Quelle: eigene, nach http://www.fedora.info/definitions/1/0/fedora-relsext-ontology.rdfs (2009-03-18)

Entsprechend der Web Service Architektur (vgl. Abschnitt 2.3) wird mit den genannten SOAP und REST APIs also die Möglichkeit gegeben, im Sinne einer service-orientierten Architektur mit dem Fedora Repository zu interagieren.

# 3 Integration von Workflows und digitalen Informationsobjekten

Die Einführung und Verwendung von Workflows zur Pflege von digitalen Informationsobjekten im Rahmen des vorgestellten Szenarios (siehe Abschnitt 1.1 und detaillierter 3.5) dient keinem Selbstzweck. Erkenntnisse, die sich aus der Integration von Workflows und digitalen Informationsobjekten speziell im Szenario ergeben, können auf weitere Bereiche abgebildet werden.

In „Workflow-based Integration“ beschreibt Müller 2005 (S. 25ff.) Vorteile und Gründe zur Einführung von Workflows. Als Kriterien für den Einsatz von Workflows sprechen unter anderem

- die Häufigkeit und Regelmäßigkeit eines Prozesses,
- die Anzahl der beteiligten Mitarbeiter und
- die Struktur eines Geschäftsprozesses.

Bereits das hier betrachtete Szenario (ausführlich in Abschnitt 3.5) erfüllt diese genannten Kriterien.

Im Folgenden bezeichnet das Wort Integration den Zusammenhang und das Zusammenbringen von Workflows und digitalen Informationsobjekten in einem System.

## 3.1 Anforderungen an die Integration

Im Kontext der Integration von Workflows und digitalen Informationsobjekten ergeben sich verschiedene Einflussgrößen und Anforderungen. Abb. 3.1 veranschaulicht informell den Zusammenhang zwischen Faktoren und den Einfluss auf Workflows. Entsprechend können diese Anforderungen als eine Vorstufe und Motivation für eine Architektur betrachtet werden, die in Abschnitt 4 allgemein erläutert und in Abschnitt 5, der Implementierung, auf das Muradora Framework angewendet wird, um es diesbezüglich zu evaluieren.

Als Akteure für die Bearbeitung von digitalen Informationsobjekten mit Hilfe von Workflows zeigt Abb. 3.1 einen Administrator und einen Benutzer. Der Administrator hat hier die Möglichkeit Objekttypen, Workflow Definitionen und Rechte zu bearbeiten. Auf der anderen Seite ist es dem Benutzer möglich digitale Informationsobjekte zu bearbeiten und dafür Workflows zu verwenden. Die Funktionen, die ein Benutzer verwenden kann, hängen von den ihm zur Verfügung stehenden Rechten ab.

Auf der linken Seite werden digitale Informationsobjekte (im Folgenden kurz auch Objekte genannt) gezeigt, welche hier als Instanz eines Objekttyps betrachtet werden. Objekttypen geben die Struktur eines Objektes vor und bilden damit ein Grundgerüst für den Aufbau und die Art eines Objektes. Der Aufbau eines Objektes kann sich je nach Anwendungsgebiet anders darstellen. So ist ein Objekt von der Art beziehungsweise dem Typ „Dokument“ möglicherweise verschieden von einem Objekt, welches dem Typ „Bild“ angehört. Dabei ist zu beachten, dass ein Objekttyp nicht zwingend einschränkend ist. Es ist denkbar, dass ein konkretes Objekt weitere Informationen enthält, als vom Objekttyp selbst vorgegeben werden.

Weiterhin können Objekte und Objekttypen grundsätzlich Beziehungen zu anderen Objekten beziehungsweise Objekttypen besitzen.

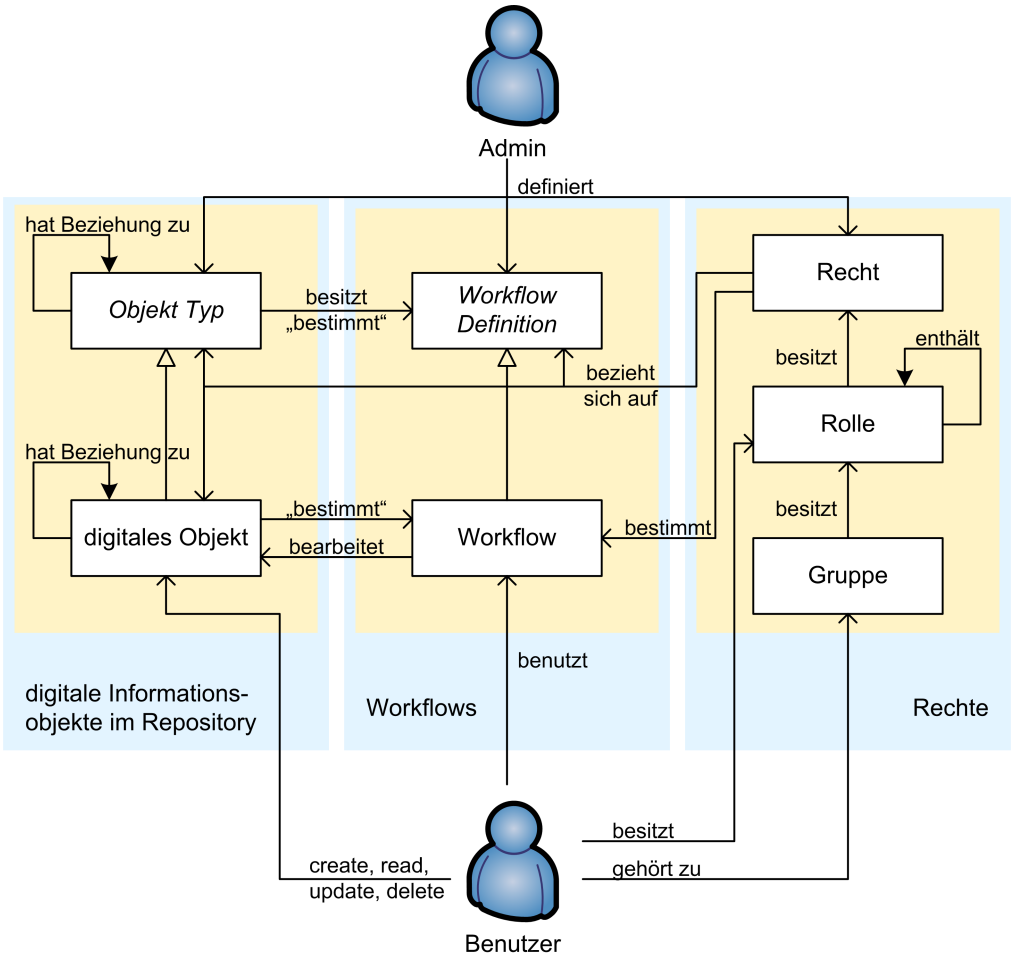


**Abb. 3.1: Zusammenhang zwischen digitalen Informationsobjekten und Workflows**[17]

Im Zentrum der Abbildung steht der Workflow an sich, welcher durch äußere Bedingungen beeinflusst wird. Hier wird zwischen der Definition eines Workflows und dem tatsächlichen Workflow unterschieden. Eine Workflow-Definition beschreibt dabei

[17] Quelle: eigene

die grundlegende Struktur eines Workflows. Zu dieser Struktur zählen neben dem eigentlichen Arbeitsablauf auch Einschränkungen durch Rechte. So können Benutzer möglicherweise nicht alle Schritte eines Workflows ausführen. Ein Workflow wird weiter als Instanz einer Workflow-Definition betrachtet, welche durch konkrete Objekte und Rechte beeinflusst wird.

Auf der rechten Seite sind die Rechte abgebildet. Benutzer besitzen hier Rechte, welche die Einschränkung von Workflows und den Zugriff auf Objekte erlauben oder beschränken können. Die Bereitstellung von Rollen und Gruppen ermöglicht die Zusammenfassung von Rechten für ähnliche Anspruchsgruppen oder für ähnliche Benutzer. So genügt es an dieser Stelle eine Zuordnung von Rollen zu Benutzern oder die Zuordnung von Rollen zu Gruppen und Gruppen zu Benutzern vorzunehmen, um mehrere Benutzer mit gleichen Rechten zu versehen.

## 3.2 Ansatz der Integration

Bei der Zusammenführung von Workflows und digitalen Informationsobjekten sind verschiedene Ansätze für das Workflowmanagement denkbar. Hier seien beispielhaft zwei Ansatze genannt.

- Klassische Workflows beziehen sich auf recht starre Arbeitsabläufe, die nur geringen Spielraum für Dynamik erlauben, dafür aber recht gut strukturiert sind. Ein Workflow bestimmt die Schritte zur Erreichung eines Ziels (siehe auch Abschnitt 2.4).

- Eher datenorientierte Ansätze wie beispielsweise das Case Handling ermöglichen im Gegensatz dazu die Bearbeitung von Objekten in dynamischer Weise, wobei der Benutzer die Art und Weise zur Erreichung eines Ziels bestimmt (van der Aalst u. a. 2005).

Vorteile von klassischen Workflows liegen in der Definition beziehungsweise der Modellierung von Workflows begründet. Zwar bieten Workflows hier nur einen relativ festen Ablauf von Aktivitäten zu Erreichung eines Ziels, doch ist damit natürlich auch die Ausführung eines Workflows bekannt und die Strukturen eines Prozesses sind klar beschrieben.

Dynamische Ansätze, welche die Strukturen von klassischen Workflows kritisieren und flexiblere Methoden für Workflows anbieten, können aber gerade durch ihre Dynamik und Flexibilität eine hohe Komplexität annehmen, welche ebenfalls als problematisch zu bewerten ist (Agostini und De Michelis 2000).

Entsprechend der verschiedenen Einsatzmöglichkeiten und Anforderungen, welche von der Systemumgebung, also der realen und relevanten Umwelt mit ihren Geschäftsprozessen, gestellt werden, bieten sich verschiedene Ansätze des Workflowmanagements beziehungsweise von Workflows an (Agostini und De Michelis 2000, Reijers u. a. 2003, Nurcan und Edme 2005).

Während der Abarbeitung eines Workflows können verschiedene Zusammenhänge zwischen Workflow und digitalen Informationsobjekten unterschieden werden.

- Ein Workflow bezieht sich auf die Bearbeitung eines digitalen Informationsobjektes und orientiert sich dabei an einem Objekttyp, wie in Abschnitt 3.1 angesprochen. Dieses ist beispielsweise der Fall, wenn tatsächlich nur ein digitales Informationsobjekt bearbeitet wird. Im Szenario könnte dieses ein Dokument im Publikationsprozess sein.

- Ein Workflow bezieht sich auf die Bearbeitung von mehreren digitalen Informationsobjekten. Hierbei können innerhalb eines Workflows mehrere digitale Informationsobjekte bearbeitet werden und gegebenenfalls auch Beziehungen untereinander gepflegt werden.

- Digitale Informationsobjekte können über einen konkreten Status oder eine Aggregation von Daten zu einem Status verfügen, welcher den Stand der Bearbeitung eines digitalen Objektes beschreibt. Ein solcher Status kann Auswirkungen auf die nächsten Schritte eines Workflows haben.

Für die Integration von Workflows und digitalen Informationsobjekten soll in dieser Arbeit im Wesentlichen der klassische Ansatz verfolgt werden. Die Auswahl des klassischen Ansatzes gründet in der klaren Strukturiertheit von Prozessen. Um Informationen über digitale Informationsobjekte und Informationen über Rechte, wie in Abschnitt 3.1 angesprochen, beachten zu können, muss in einer Workflow Architektur neben dem Kontrollfluss auch die Datensicht Beachtung finden.

## 3.3 Anforderungen an die Definition von Workflows

Workflows sollen hier im Sinne der service-orientierten Architektur (Abschnitt 2.2) definiert werden können. Entsprechend dieses Ansatzpunktes können einige minimale Anforderungen an die Definition und an die Ausführung eines Workflows gestellt werden.

- Die Definition eines Workflows soll unabhängig von den verwendeten Systemen ermöglicht werden. Hier bietet sich eine Beschreibung von Workflows im XML Format an, welches unter anderem flexibel, systemunabhängig und maschinenlesbar ist.

- Die Ausführung eines Workflows erfolgt mit so genannten Workflow-Engines. Eine Workflow-Engine muss die Definition eines Workflows abarbeiten können und in die Arbeitsumgebung integrierbar sein.

- Sowohl Workflow Sprache als auch Workflow-Engine müssen die Integration mit digitalen Informationsobjekten, also Daten, erlauben.

- Workflow Sprache und Workflow-Engine müssen die Beachtung von Rechten für ganze Workflows oder einzelne Schritte ermöglichen.

Weitere Anforderungen an ein System zur Integration von Workflows und digitalen Informationsobjekten ist dem Anhang A zu entnehmen.

## 3.4 Sprache zur Definition von Workflows

In Abschnitt 2.4 wurde bereits kurz angesprochen, dass zahlreiche Sprachen für Definition von Workflows existieren und dass es ebenso eine Vielfalt von Workflow-Engines für entsprechende Sprachen gibt.

Die Auswahl einer konkreten Sprache mit dazugehöriger Engine erfordert die gezielte Auseinandersetzung mit den hier vorgestellten Anforderungen und der Leistungsfähigkeit existierender Sprachen und Engines für die Anwendungsumgebung. Die angesprochene Vielzahl erschwert den Überblick über existierende Sprachen. Weiterhin kann davon ausgegangen werden, dass die Durchführung von Analysen und Machbarkeitsstudien für Sprachen und Engines einen erheblichen Aufwand bedeuten würden.

Zwar sind solche Analysen Gegenstand aktueller Forschung, doch wird hier der Blickpunkt eher auf die generellen Möglichkeiten, beispielsweise die Unterstützung von Workflow Mustern, gelegt (Wohed u. a. 2008). Die Betrachtung von Einsatzmöglichkeiten oder Machbarkeitsstudien im Speziellen, wie hier für die Verwendung mit digitalen Informationsobjekten, wird in solchen Studien verständlicher Weise nicht vorgenommen.

Um den Rahmen dieser Arbeit nicht durch die Analyse vorhandener Sprachen und Engines zu sprengen, wird eine eigene prototypische Sprache vorgestellt, welche die

genannten Anforderungen erfüllt. Damit wird auf die besprochene Problemstellung eingegangen und die Lösung problemadäquat gehalten. Im Gegensatz dazu bieten viele Workflow-Lösungen unter anderem ein erhebliches Überangebot an Funktionalität an. Die Anforderungen mit einer entsprechenden Evaluierung sind in Kapitel 6 zu finden.

Die Workflow Management Coalition (WfMC) definiert mit der XML Process Definition Language (XPDL) eine Sprache zur Beschreibung von Workflows. Die aktuelle Version XPDL 2.1 wurde als finale Spezifikation Ende 2008 veröffentlicht[18] (Workflow Management Coalition 2008).

In Anlehnung an den XPDL Standard wird die hier verwendete Sprache definiert. Die Sprache stellt dabei nur einen Ausschnitt der mit XPDL möglichen Beschreibungen dar und konzentriert sich auf die für diese Arbeit relevanten Anforderungen. Mögliche Abweichungen der Sprache von XPDL können dabei durch Transformationen, beispielsweise mit Extensible Stylesheet Language Transformation (XSLT), behoben werden und werden wegen dem einfacherem Aufbau der Sprache verwendet.

Im Anhang B ist das für die vorliegende Arbeit entwickelte XML Schema zur Beschreibung von Workflows zu finden. Die Inhalte des Schemas werden im Folgenden vorgestellt. Abb. 3.2 zeigt das wesentliche Schema für Workflows in Form eines Klassendiagramms.

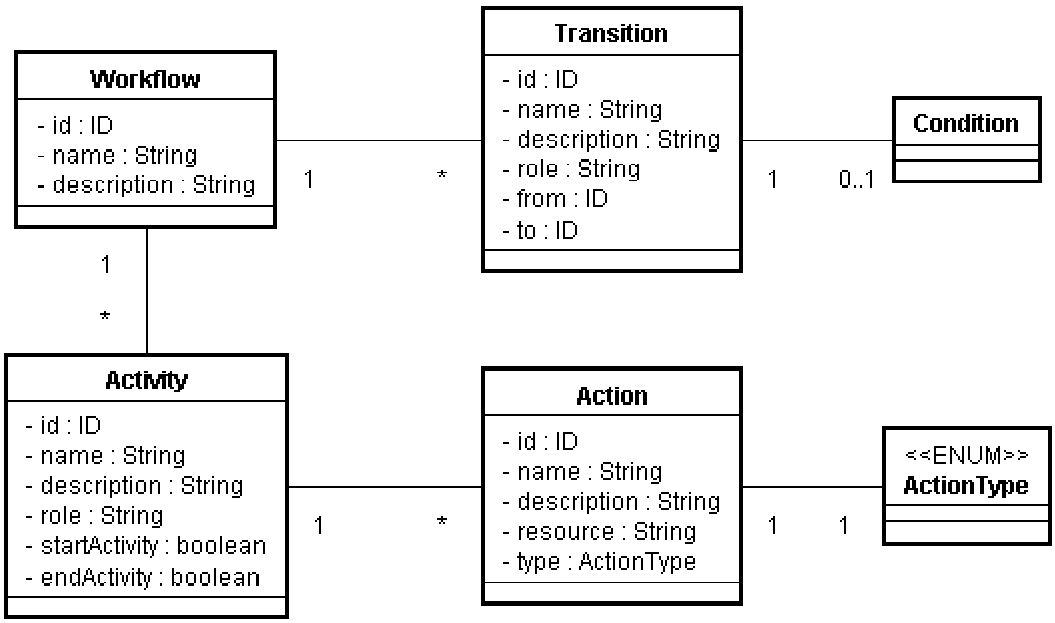


**Abb. 3.2: Schema eines Workflows als informales Klassendiagramm**

[18] auf den Seiten der WfMC (http://www.wfmc.org/ [2009-03-18]) durch Anmeldung erhältlich

Eine Workflow-Definition (`Workflow`) wird als XML Dokument formuliert. Dieses Dokument enthält als Wurzel ein Workflow-Element, welches eine Zahl von Aktivitäten (`Activity`) und Transitionen (`Transition`) besitzt.

Aktivitäten können als Aufgaben betrachtet werden und entsprechen in der EPK-Notation (vgl. Abschnitt 2.4.1) einer Funktion. Eine Aktivität wiederum kann eine Reihe von kleineren Aktionen (`Action`) beinhalten, welche auf Ressourcen (Implementierungen) verweisen. Zu solchen Aktionen zählen hier zunächst verschiedene Typen (`ActionType`), das heißt Verhaltensweisen einer Aktion, die sich auf die Implementierung beziehen:

- `Class`. Eine Klasse repräsentiert eine Java-Klasse, welche ausgeführt werden soll.
- `Method`. Eine Methode repräsentiert eine bestimmte Methode einer Klasse, die ausgeführt werden soll.
- `Jsp`. Eine JavaServer Page verweist auf eine bestimmte Seite, die dem Benutzer zur Interaktion mit dem Workflow angeboten werden soll.
- `Form`. Eine Form bezeichnet ein Formular, welches beispielsweise aus XML Schemata erzeugt und dem Benutzer zur Interaktion mit dem Workflow angeboten werden soll.
- `WebService`. Ein Web Service verweist auf einen Dienst, der durch die Aktion aufgerufen werden soll.

Die letzten beiden Typen werden durch den in dieser Arbeit entwickelten Prototypen noch nicht unterstützt, da sie nicht Teil der Anforderungen sind. Sie können als komfortable Erweiterungen betrachtet werden, die in weiteren Entwicklungen Beachtung finden sollten.

Aktivitäten werden durch Transitionen miteinander verbunden. Dabei können Transitionen als einfache Verbindung oder auch als bedingte Verbindung betrachtet werden. So können Bedingungen (`Condition`) formuliert werden, welche die Fortsetzung eines Workflows nur erlauben, wenn bestimmte Bedingungen wahr sind. Dabei können sich solche Bedingungen auf digitale Informationsobjekte beziehen. Des Weiteren erlaubt die Workflow-Definition die Einschränkung von Aktivitäten und Transitionen für bestimmte Rollen. Es besteht also die Möglichkeit, dass einzelne Schritte nur bestimmten Benutzern vorbehalten bleiben.

Im Folgenden wird das Szenario und in Abschnitt 3.6 der Workflow für das Szenario vorgestellt und die zuvor beschriebene Workflow-Sprache dabei praktisch angewandt.

## 3.5 Szenario

Im Rahmen der Anwendungsumgebung werden am AWI laufend von unterschiedlichen Personen zahlreiche neue Dokumente verschiedenen Typs erstellt, welche einem Veröffentlichungsprozess am AWI unterliegen. Das Szenario betrachtet dabei nun Veröffentlichungen, welche über das Epic System (electronic Publication Information Center) des AWI verwaltet werden.

Die Kriterien zur Einführung von Workflows, die kurz in Abschnitt 3 angesprochen wurden, kommen schon hier Anwendung. Das Szenario tritt zum einen häufig auf und bezieht die Mitarbeiter des AWI (ca. 780[19]) sowie externe Personen mit ein. Weiterhin gibt es einen Prozess zur Veröffentlichung von Dokumenten.

Das betrachtete Szenario zeigt den Prozess zur Veröffentlichung von Dokumenten. Abb. 3.3 stellt diesen Prozess als ereignisgesteuerte Prozesskette dar. Dabei sind die Schritte im oberen Bereich einem Benutzer und die Schritte im unteren Bereich einem Kurator zugeordnet.

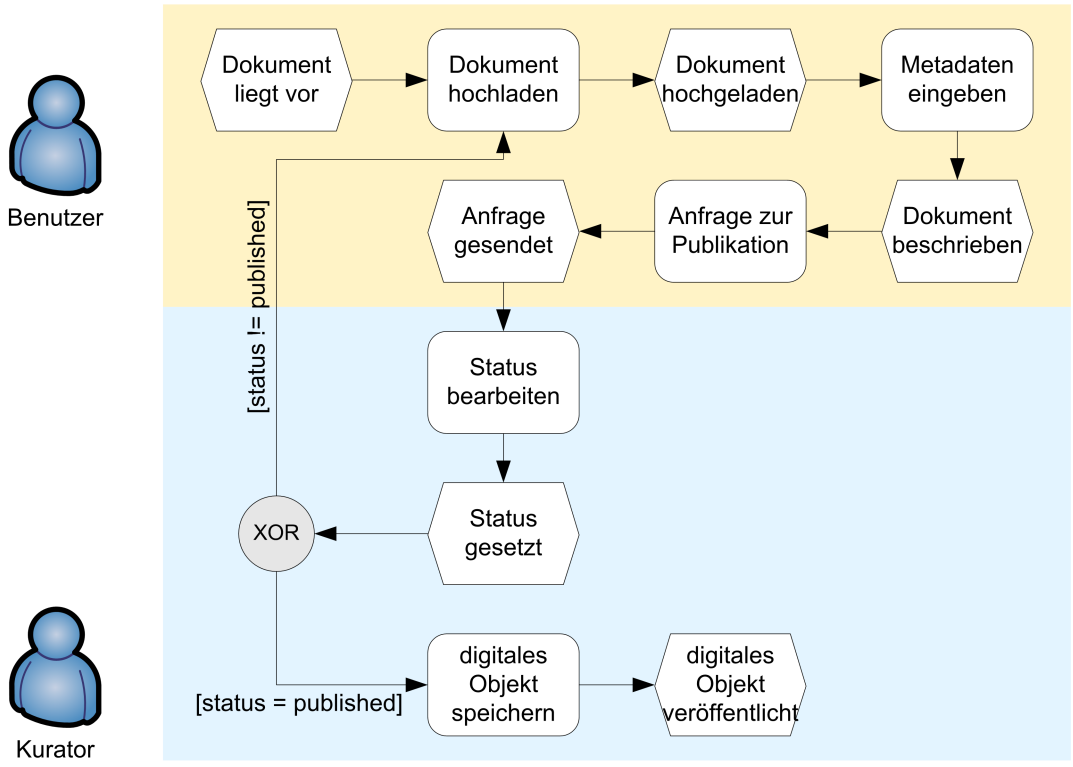


**Abb. 3.3: Prozess zur Veröffentlichung eines Dokuments im Szenario[20]**

Der Prozess beginnt mit der wissenschaftlichen Forschung und der Formulierung von Ergebnissen. Diese Ergebnisse werden von Benutzern in Dokumenten verfasst.

[19] http://www.awi.de/de/institut/ (2009-03-18)
[20] Quelle: eigene

Für diese Arbeit sind die folgenden Schritte von Interesse: das Archivieren von Dokumenten und die Annotation dieser Dokumente mit Metadaten, kurz die Pflege digitaler Informationsobjekte.

Nachdem ein Benutzer ein Dokument verfasst hat, wird dieses Dokument hochgeladen und der hier betrachtete Workflow beginnt. Neben dem eigentlichen Dokument gibt der Benutzer Metadaten zu diesem Dokument an. Zu diesen Metadaten zählen Informationen wie Titel, Autor, Jahr und Beschreibung, die beispielsweise durch Dublin Core[21] abgebildet werden.

Im Anschluss an die Angabe der Metadaten wird eine Anfrage zur Veröffentlichung des Dokuments an einen Kurator gesendet. Der Kurator ist für die Überprüfung beziehungsweise die Überarbeitung des Dokuments und der Metadaten zuständig und zeichnet sich entsprechend dafür verantwortlich.

Schließlich setzt der Kurator den Status des Dokuments. Dabei werden mehrere Zustände unterschieden:

- `create`: Das Dokument befindet sich in der Bearbeitung des Benutzers.
- `request`: Der Benutzer hat eine Anfrage an den Kurator für die Veröffentlichung des Dokuments gesendet.
- `review`: Der Kurator hat den Status geändert und gibt damit an, dass das Dokument und seine Metadaten überarbeitet werden müssen. Dementsprechend wird der Prozess wieder zum Benutzer geleitet.
- `published`: Der Kurator hat das Dokument veröffentlicht.

Mit dem Setzen des Status `published` wird ein Dokument als veröffentlicht bezeichnet. Weiterhin werden anschließend das Dokument und seine Metadaten als digitales Informationsobjekt im Repository abgelegt. Der aufgezeigte Prozess ist demnach nicht starr, sondern bezieht den aktuellen Status eines Dokuments mit ein.

Die Bezeichnung „Dokument“ ist hier als abstrakter Begriff zu verstehen. Ein Dokument kann verschiedene konkrete Ausprägungen besitzen und dem Aufbau eines bestimmten Objekttypen folgen. Für dieses Szenario sind beispielsweise Objekttypen wie „Article“, „Book“, „InBook“ oder auch „Thesis“ denkbar, die alle dem gezeigten Prozess zur Veröffentlichung eines Dokuments unterliegen.

[21] http://dublincore.org/schemas/xmls/simpledc20021212.xsd (2009-03-18)

## 3.6 Vom Prozess des Szenarios zum Workflow

Aus dem Prozess zum Szenario, welches im vorherigen Abschnitt vorgestellt wurde, lässt sich nun auf relativ einfache Weise ein Workflow ableiten. Im Folgenden wird ein Ausschnitt des XML Dokuments des entsprechenden Workflows vorgestellt. Das komplette XML Dokument kann im Anhang C eingesehen werden.

Die Struktur des Dokuments richtet sich nach dem in Anhang B bezeichneten XML Schema und ist analog zur Abbildung des Prozesses in Abb. 3.3 zu lesen.

```
<?xml version="1.0" encoding="UTF-8"?>
<wf:Workflow id="epicPublication" name="" description=""
   xmlns:wf="http://www.object_workflow.de/2008/Workflow"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.object_workflow.de/2008/XMLSchema
   workflow.xsd ">

   <wf:Activities>
      ...

      <!-- Aktivität: Anfrage zur Publikation, setze Status auf
         "request" und speichere digitales Objekt -->
      <wf:Activity id="request" name="" role="" description=""
         startActivity="false" endActivity="false">
         <wf:Action id="request" name="" type="Method" description=""
            resource="workflows.epic.Workflow:request" />
         <wf:Action id="requestjsp" name="" type="Jsp" description=""
            resource="epic/request.jsp" />
      </wf:Activity>

      <!-- Ereignis: Anfrage gesendet, digitales Objekt gespeichert -->

      <!-- Aktivität: Status bearbeiten, setze Status auf "review"
         oder "published" -->
      <wf:Activity id="status" name="" role="curator" description=""
         startActivity="false" endActivity="false">
         <wf:Action id="statusjsp" name="" type="Jsp" description=""
            resource="epic/status.jsp;epic.StatusBean" />
      </wf:Activity>

      <!-- Ereignis: Status gesetzt -->
      ...
   </wf:Activities>

   <wf:Transitions>
      ...

      <wf:Transition id="t3a" name="" role="curator" description=""
         from="request" to="status">
      </wf:Transition>
      ...
```

```
    <wf:Transition id="t4a" name="" role="curator" description=""
      from="status" to="published">
      <wf:Condition>
        epic.MetadataBean:epic.recordStatus.name.toLowerCase ==
          "published"
      </wf:Condition>
    </wf:Transition>

    <wf:Transition id="t4b" name="" role="curator" description=""
      from="status" to="upload">
    </wf:Transition>
  </wf:Transitions>
</wf:Workflow>
```

**Listing 3.1: Workflow zur Veröffentlichung eines Dokuments im Szenario**

Das Listing 3.1 zeigt mit den `Activity`-Elementen Funktionen, die in Abb. 3.3 als gerundete Kästchen dargestellt sind. Ereignisse werden in der Workflow-Definition nicht explizit betrachtet. Ereignisse stellen hier die Ausgabe einer Aktivität dar, welche wiederum als Eingabe der folgenden Aktivität betrachtet werden kann. In der dargestellten Form werden keine Vor- und Nachbedingungen innerhalb der Workflow-Definition beachtet. Vielmehr müssen in der vorgestellten Form nachfolgende Aktionen für die Gültigkeit ihrer Ausführung Sorge tragen. Transitionen verbinden die einzelnen Aktivitäten, wie aus Abb. 3.3 zu erkennen ist.

Die Aktivität mit der Id `request` besitzt keine Einschränkung der Rolle, somit kann jeder Benutzer diese Aktivität ausführen. Die Aktivität beinhaltet zwei atomare Aktionen. Zunächst wird eine Methode aufgerufen, welche die *Anfrage zur Publikation* durchführt. Die zweite Aktion präsentiert dem Benutzer des Systems eine JSP Seite zur Information.

Im nächsten Schritt kommt die Transition mit der Id `t3a` zur Ausführung, wenn der Benutzer die Rolle `curator` besitzt. Der Benutzer wird dann zur Aktivität `status` geführt. Sollte der Benutzer die angegebene Rolle nicht besitzen, wird der Workflow hier zurück an `upload` (siehe Anhang C) geleitet. Diese Umleitung ist der Übersicht halber in Abb. 3.3 nicht zu sehen und dient im Workflow der Bearbeitung eines digitalen Objektes.

Die Aktivität `status` ermöglicht nun einem Kurator die Bearbeitung eines Status, der sich auf das digitale Objekt bezieht. Setzt der Kurator den Status des digitalen Objektes auf `published`, so wird die Transition `t4a` aktiviert. Die Bedingung (`condition`) verweist auf den `recordStatus` des digitalen Objektes mit dem Namen „epic“. Der Aufbau des digitalen Objektes wird in Abschnitt 5.2.1 genauer betrachtet. Alternativ zu dieser Transition kann die Transition `t4b` aktiviert sein – immer dann, wenn der Status

nicht `published` ist. Somit bilden diese beiden Transitionen ein logisches exklusives Oder.

## 3.7 Ontologie für digitale Informationsobjekte

Die Anforderungen in Abschnitt 3.1 zeigen, dass digitale Informationsobjekte Beziehungen untereinander aufweisen können. Solche Beziehungen können vielfältiger Natur sein und verschiedenen Bedingungen unterliegen. Um entsprechende Beziehungen und die Struktur eines Anwendungsbereichs ausdrücken zu können, wird hier OWL verwendet. Im Grundlagenabschnitt 2.1.2 wurde OWL als Sprache für Ontologien vorgestellt.

OWL ermöglicht es, die Struktur von digitalen Objekten und die Beziehungen zwischen digitalen Objekten darzustellen. Mit Bezug zu Abschnitt 3.1 existieren Objekttypen, die als eine Art Prototyp für digitale Informationsobjekte betrachtet werden können.

Für verschiedene Arten von Objekten sind auch unterschiedliche Workflows denkbar. Objekttypen besitzen also nicht nur mögliche Beziehungen untereinander sondern auch verschiedene Workflows, die mit ihnen assoziiert werden.

Der Prozess des Szenarios in Abschnitt 3.5 ermöglicht die Veröffentlichung von Dokumenten. Dokumente können dabei verschiedener Art sein. Beispiele für Dokumente sind hier *Book* und *InBook*.

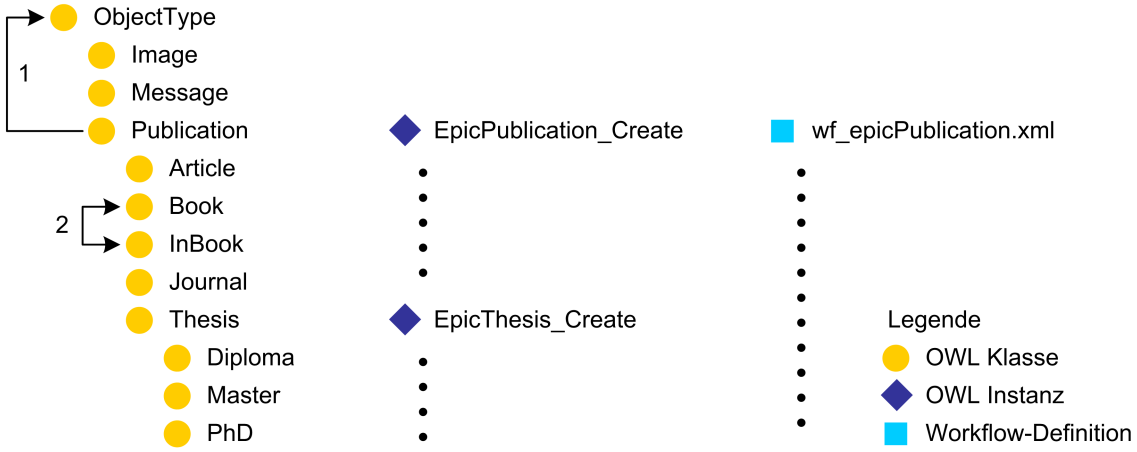


**Abb. 3.4: Ontologie für Objekttypen**[22]

[22] Quelle: eigene

Abb. 3.4 zeigt den Aufbau einer Ontologie, die als Beschreibung des Anwendungsbereichs betrachtet werden kann. Die Ontologie kann durchaus für weitere Anwendungen und Objekttypen erweitert werden.

Auf der linken Seite der Abbildung sind die OWL Klassen zu erkennen. Im mittleren Bereich sind zwei Instanzen dargestellt, die sich auf die Klassen in gleicher Höhe beziehen. Schließlich wird rechts der Verweis auf einen Workflow gezeigt.

Die Instanzierung einer Klasse in OWL ermöglicht die Beschreibung von Individuen für eine solche Klasse. Für die angesprochene Ontologie beschreiben Instanzen eines Objekttyps verschiedene weitergehende Eigenschaften eines Objekttyps. Eine Eigenschaft ist beispielsweise der Verweis auf einen Workflow. Entsprechend werden Workflows nach der Instanz eines Objekttyps ausgewählt.

Weitere Eigenschaften, die durch Instanzen abgebildet werden können, sind die Einschränkung eines Objekttyps auf bestimmte Rollen von Benutzern oder beispielsweise auch die Angabe von XML Schemata zur Definition der Struktur eines digitalen Objektes. Die Definition der Struktur eines digitalen Objektes könnte hierbei die Struktur der Metadaten des digitalen Objektes beschreiben. Der hier entwickelte Prototyp beschränkt sich allerdings ausschließlich auf die Beachtung von Workflows, die mit Objekttypen assoziiert werden können.

Die gezeigte Klassenhierarchie gilt als Erweiterung auch für die Instanzen und Workflows. Deren Nutzen liegt in der Möglichkeit begründet, Instanzen und Workflows an untergeordnete Objekttypen zu vererben (in der Abbildung durch die Punkte angedeutet). Somit genügt die generische Definition einer Instanz beziehungsweise eines Workflows für eine Reihe von untergeordneten Objekttypen.

Die vorliegende Ontologie zeigt, mit dem Szenario als Anwendungsbereich, folglich

- die einfache hierarchische Strukturierung von Objekttypen,
- die beispielhafte Definition einer inversen Eigenschaft (2) „inBook“ und „hasInBook“ an den Klassen *Book* und *InBook* und
- Beziehungen die durch eine Eigenschaft (1) „references“ abgebildet werden. So kann ein digitales Objekt vom Typ *Publication* auf alle möglichen digitalen Objekte eines beliebigen Objekttyps verweisen.

Der Anwendungsbereich des Szenarios zeigt bereits einige Möglichkeiten, die mit OWL dargestellt werden können. Die Unterstützung von OWL durch Software, wie

beispielsweise durch das Modellierungswerkzeug Protégé[23], und die Potentiale der Beschreibungslogik von OWL ermöglichen die Erweiterung einer Ontologie für Objekttypen um weiterführende Anwendungen und Konzepte.

Das folgende Kapitel verbindet die zuvor vorgestellten und entwickelten Ansätze mit einer Architektur für Workflows und digitale Informationsobjekte.

[23] http://protege.stanford.edu/ (2009-03-18)

# 4 Architektur des Systems

Dieses Kapitel erläutert die hier entwickelte Architektur. Die Architektur selbst ist so generisch wie möglich gehalten und kann daher auch bei Änderungen von Teilsystemen Wiederverwendung finden beziehungsweise als Bezugspunkt genutzt werden.

Im Weiteren wird auf die Architektur des untersuchten Frameworks Muradora eingegangen.

## 4.1 Architektur

Abb. 4.1 zeigt die angesprochene Architektur. Die eingezeichneten Pfeile weisen auf den Informationsfluss zwischen den dargestellten Komponenten hin. Grundlegend ist die strikte Trennung der einzelnen Teilsysteme zu erkennen. Als solche Teilsysteme werden hier

- die Anbindung an das Repository für digitale Informationsobjekte (siehe Abschnitt 4.1.1),
- das Workflow-System (siehe Abschnitt 4.1.2),
- die Verwaltung und Anfrage von Objekttypen (siehe Abschnitt 4.1.3),
- die Speicherung, sowie die Verwaltung von Rechten und Rollen (siehe Abschnitt 4.1.4) und
- die Präsentation von grafischen Benutzungsschnittstellen, mit der Anbindung an Frontends wie beispielsweise Muradora (siehe Abschnitt 4.1.5) betrachtet.

Diese Teilsysteme sollen eine lose Kopplung aufweisen, um zu gewährleisten, dass auf relativ einfache Weise Erweiterungen dieser Systeme vorgenommen werden können. Neben Erweiterungen ist durch eine lose Kopplung und die Definition von Schnittstellen auch der Austausch von Teilsystemen denkbar. Dabei sollten weitere Grundprinzipien wie Autonomie und Abstraktion bei der Entwicklung von Teilsystemen beachtet werden.

Weiterhin sei für eine Implementierung auf die Grundprinzipien der service-orientierten Architektur verwiesen. Vergleiche dazu Abschnitt 2.2.

Im Folgenden werden die einzelnen Teilsysteme und ihre Funktion erläutert.

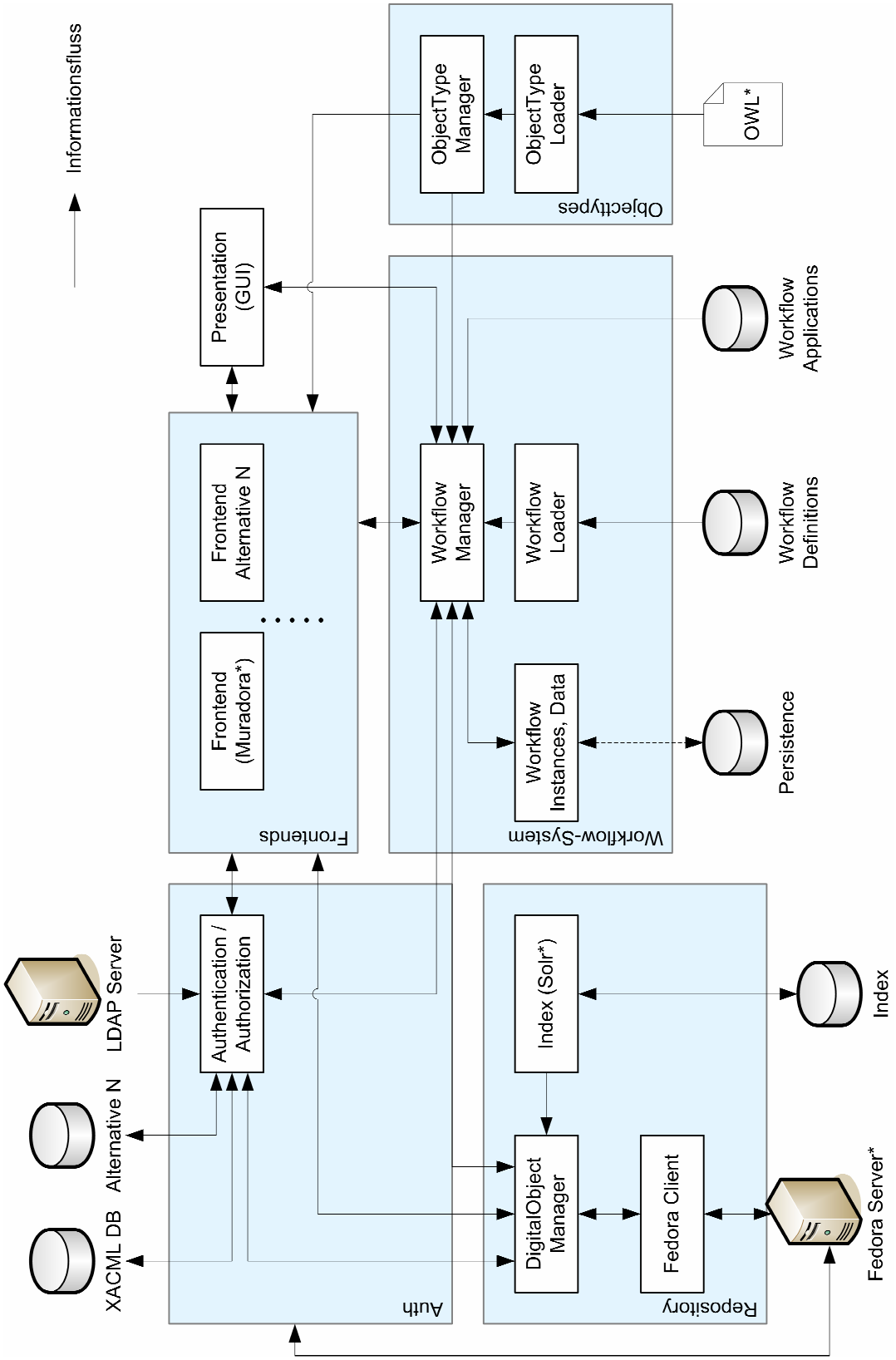


**Abb. 4.1: Architektur für Workflows und digitale Informationsobjekte**[24]

[24] Quelle: eigene

### 4.1.1 Repository für digitale Informationsobjekte

Das *Repository* Teilsystem bildet die Schnittstelle zum eigentlichen Repository und bietet eine Reihe von Zugriffsmöglichkeiten auf ein Repository. Im Allgemeinen kann eine beliebige Implementierung eines Repositories zum Einsatz kommen. Im Speziellen ist in dieser Arbeit das Fedora Framework von Interesse.

Der *DigitalObjectManager* stellt für die anderen dargestellten Systeme die Schnittstelle auf das Repository dar. Zum einen nutzt der *DigitalObjectManager* einen *FedoraClient* um mit dem Fedora Server zu kommunizieren. Auf der anderen Seite kann der *DigitalObjectManager* weitergehende Dienste anbieten.

Eine mögliche Erweitung ist die Nutzung von Werkzeugen zur *Indexierung* von digitalen Informationsobjekten. Eine geeignete Indexierung von digitalen Informationsobjekten ermöglicht beispielsweise die performante Anfrage von indexierten Metadaten zur Darstellung in beziehungsweise durch Frontends. Zum Beispiel kann hier der Search Server Solr[25] aus dem Apache Projekt Verwendung finden.

Die Verbindung zu dem Repository Server, hier dem Fedora Server, erfolgt durch den Einsatz von Web Services. Ebenso sollten auch weitergehende Dienste, wie die Indexierung über Web Services bereitgestellt werden. Dieses Vorgehen ermöglicht eine Architektur im Sinne der SOA (vgl. Abschnitt 2.2).

Die Architektur zeigt Verbindungen zu den Teilsystemen *Auth*, *Frontends* und dem *Workflow-System*. Der Zusammenhang zwischen diesen Systemen ist offensichtlich: Digitale Informationsobjekte können Einschränkungen unterliegen, so dass *Frontends* verschiedene Sichten auf digitale Informationsobjekte für unterschiedliche Benutzer bereitstellen sollen. Die Verbindung zum *Workflow-System* liegt in der Arbeit von Benutzern mit Workflows begründet: Workflows arbeiten auf oder bearbeiten digitale Informationsobjekte.

### 4.1.2 Workflow-System

Das *Workflow-System* besitzt Beziehungen zu allen in der Architektur aufgezeigten Teilsystemen. Wie bereits im vorherigen Abschnitt angesprochen, arbeiten Benutzer

[25] http://lucene.apache.org/solr/ (2009-03-18)

mittels Workflows mit digitalen Informationsobjekten. Daher existiert bereits eine Verbindung zum *Repository* Teilsystem und dem *Auth* Teilsystem.

Weiterhin können *Frontends* wie Muradora das *Workflow-System* nutzen. Das *Workflow-System* selbst produziert aus den aktuellen Schritten eines Workflows Sichten zur Interaktion mit dem Benutzer eines Workflows. Somit existiert auch eine Verbindung zur Präsentation von grafischen Benutzungsschnittstellen.

Die Beziehung zu dem *Objecttypes* Teilsystem stellt Funktionen für Objekttypen zur Verfügung. Objekttypen können die Struktur von digitalen Objekten beschreiben, so dass diese auch für Workflows genutzt werden können (vergleiche dazu Abschnitt 3.7).

Das *Workflow-System* selbst teilt sich in weitere Komponenten auf. Dabei kommt dem *WorkflowManager* eine wesentliche Bedeutung zu. Der *WorkflowManager* übernimmt alle Aufgaben zur Organisation und Ausführung von Workflows.

Workflows werden in dieser Arbeit durch die in Abschnitt 3.4 vorgestellte Sprache definiert. Der *WorkflowLoader* erzeugt aus den vorliegenden XML Dokumenten zur Definition von Workflows eine Repräsentation, die durch den *WorkflowManager* programmtechnisch verwendet werden kann.

Der *WorkflowManager* verwendet die durch den *WorkflowLoader* bereitgestellte Struktur zur Ausführung von Workflows. Dabei kommen für einen Workflow spezifische Anwendungen zum Einsatz, die in der Architektur als *Workflow Applications* angedeutet sind.

Schließlich erfordert die Verwaltung von Workflows auch die Beachtung von *Workflow-Instanzen*. *Workflow-Instanzen* speichern den aktuellen Zustand eines Workflows eines konkreten Arbeitsablaufs.

Als Erweiterung dieser grundlegenden Workflow-Funktionen ist es denkbar, Workflow-Instanzen persistent abzulegen. Durch Persistenz können beispielsweise unterbrochene Workflow-Instanzen gespeichert oder auch eine Versionierung von Workflows unterstützt werden. Dieses sind allerdings weitergehende Funktionen, die über den Rahmen dieser Arbeit hinausgehen.

### 4.1.3 Verwaltung von Objekttypen

Das Teilsystem für Objekttypen ermöglicht die Verwaltung von Objekttypen als Struktur für digitale Informationsobjekte. Neben dem Laden von Objekttypen mit dem *ObjectTypeLoader* kann der *ObjectTypeManager* den Zugriff auf Objekttypen erlauben.

Der *ObjectTypeManager* kann als Erweiterung die Inferenz, also das logische Schließen, von Objekttypen gestatten und weitere Möglichkeiten anbieten, die durch eine Ontologie zur Abbildung von Objekttypen bereitgestellt werden können. Ein Beispiel ist hier die Auflösung von Beziehungen zwischen verschiedenen Objekttypen (vergleiche Abschnitt 2.1 und 3.7).

### 4.1.4 Rechte und Rollen

Das Teilsystem *Auth* ist als zentrale Instanz für alle Fragen zu Rechten zu verstehen. Neben der Verwaltung von Rechten für digitale Informationsobjekte sind auch Rollen für die Verwendung von Workflows interessant.

Die Informationen für Rechte und Rollen können in verschiedener Form vorliegen und müssen geeignet für die anderen Teilsysteme zur Verfügung gestellt werden. So können Benutzerinformationen und Rollen von Benutzern beispielsweise auf einem Light-weight Directory Access Protocol (LDAP) Server zur Verfügung stehen, die Rechte für digitale Informationsobjekte aber in anderen Datenbanken.

Das *Auth* Teilsystem soll nun im Wesentlichen die notwendigen Informationen integrieren und für andere Teilsysteme abstrakt zur Verfügung stellen, ohne dass diese sich um die Implementierung kümmern müssen.

Die Verbindungen zu den anderen Teilsystemen wurden bereits zuvor angesprochen und sind in der Architektur durch eingezeichnete Pfeile zu erkennen.

### 4.1.5 Grafische Benutzungsschnittstelle und Frontends

*Frontends* bilden in der Architektur die Möglichkeit der grafischen Präsentation von Inhalten und die Bereitstellung weiterer anwendungsspezifischer Funktionen, die durch ein Gesamtsystem zur Verfügung gestellt werden.

In einem grundlegenden Szenario ermöglichen *Frontends* mit Hilfe der anderen Teilsysteme das Anmelden am System (*Auth*) und die Anfrage und Darstellung von digitalen Informationsobjekten (*Repository*). Sollen im Weiteren digitale Informations-

objekte erstellt oder bearbeitet werden, so kann die Verbindung zum *Workflow-System* aufgebaut werden, welches eben solche Aufgaben mit Workflows unterstützen kann.

Des Weiteren soll es auch *Frontends* möglich sein, Beziehungen zwischen Objekttypen darzustellen oder auch nur eine Übersicht von Objekttypen zu liefern, die im System zur Verfügung stehen.

Ein Beispiel für ein Frontend zeigt der in dieser Arbeit entwickelte Prototyp oder auch das in dieser Arbeit weiter betrachtete Muradora Framework.

## 4.2 Architektur und Aufbau von Muradora

Die im vorherigen Abschnitt vorgestellte Architektur stellt ein ideales Bild für den Aufbau eines Systems dar, welches durch eine lose Kopplung und die Abstraktion von Schnittstellen unter anderem Flexibilität und Erweiterbarkeit ermöglichen kann. Auch die Wiederverwendbarkeit von Teilsystemen kann damit unterstützt werden.

In diesem Abschnitt soll die Architektur beziehungsweise der Aufbau des Muradora Frameworks vorgestellt werden. Leider steht für Muradora zur Zeit der Bearbeitung dieser Arbeit nur eine beschränkte Dokumentation zur Verfügung. Daher wird im Folgenden die Architektur im Wesentlichen auf Basis eigener Untersuchungen des Frameworks vorgestellt.

Abb. 4.2 zeigt den grundlegenden Aufbau von Muradora. Der Aufbau von Muradora kann mit der Architektur, wie sie in Abschnitt 4.1 vorgestellt wurde, verglichen werden.

Muradora verfügt neben der eigentlichen Web-Anwendung auch über Möglichkeiten zum Zugriff auf den Fedora Server und einen Index-Server. Somit gehört das als *Repository* bezeichnete Teilsystem mit zu Muradora und ist nicht durch einen DigitalObjectManager getrennt.

Unabhängig von Muradora ist im oberen Teil der Abbildung die Verwaltung von Rechten für digitale Informationsobjekte angesiedelt. Mit *MelcoePDP*[26] wird ein Web Service zur Verfügung gestellt, der den Zugriff auf eine Datenbank zur Speicherung von Rechten mit der eXtensible Access Control Markup Language (XACML) ermöglicht.

[26] Muradora und die Verwendung von MelcoePDP wird in Muradora 2008 angesprochen.

Für weiterführende Informationen zum Thema XACML sei auf die XACML Spezifikation in Moses 2005 verwiesen. Die Verwendung von Rechten in Muradora wird in Nguyen 2008 veranschaulicht.

Muradora als Web-Anwendung erfordert den Zugriff auf das Fedora Repository und weitere Repository-Dienste, wie beispielsweise Solr. Dabei weist Muradora im Gegensatz zu dem im Abschnitt 4.1 vorgestellten DigitalObjectManager keine zentrale Schnittstelle zum Zugriff auf das Fedora Repository auf. Die Funktionen *Browse/Search* beziehungsweise *View/Edit/Add* abstrahieren an dieser Stelle nicht von der Implementierung und den Zugriff auf digitale Objekte.

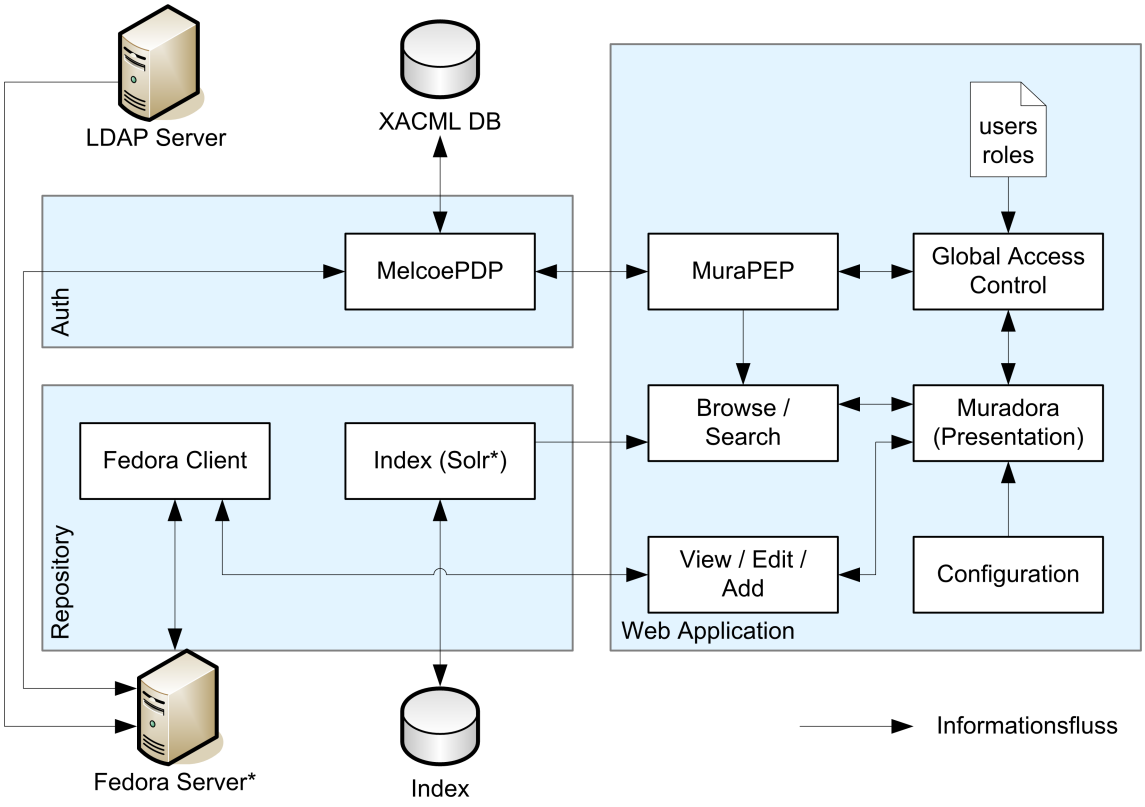


**Abb. 4.2: Aufbau von Muradora**[27]

Der Teil des *Global Access Control* ermöglicht die Bearbeitung von Rechten für Benutzer und Rollen, die dann per XACML über *MuraPEP* in der Datenbank zur Verwaltung von Rechten abgelegt werden können. Dabei werden Benutzernamen und Namen von Rollen in Muradora als Datei hinterlegt. Andere Informationen über Benutzer oder Rollen werden an dieser Stelle von Muradora nicht mit einbezogen.

[27] Quelle: eigene

Weiterhin fehlen im Vergleich der entwickelten Architektur mit dem Aufbau von Muradora ein Konzept beziehungsweise Teilsysteme, welche mit Objekttypen und Workflows umgehen könnten.

Im folgenden Abschnitt wird der wesentliche Entwurf für das in dieser Arbeit entwickelte System vorgestellt. Der Aufbau von Muradora wird bei der Evaluierung in Kapitel 6 erneut angesprochen werden.

# 5 Entwurf und Implementierung

Dieses Kapitel geht auf den Entwurf und die Implementierung des Systems ein. Voraussetzung für den Entwurf ist die im vorherigen Kapitel vorgestellte Architektur.

Zunächst werden die Technologien, die im System Verwendung finden und die durch die Anwendungsumgebung und das Muradora Framework vorgegeben werden genannt. Dann folgen der Entwurf und die Umsetzung des Systems.

## 5.1 Verwendete Technologien

Das entwickelte System wie auch die betrachteten Frameworks Fedora und Muradora sind betriebssystemunabhängige Anwendungen und sind in der Sprache Java implementiert. Zum Einsatz kommt daher hier das Java Development Kit[28] in der Version 1.6.0 Update 10, welches kompatibel zu den genannten Frameworks ist.

Das Fedora[29] Framework wird in der Version 2.2.3 verwendet. Diese Version wird von Muradora[30], welches in der Version 1.3 zum Einsatz kommt, benutzt.

Als weitere Technologie wird ein Tomcat[31] Server für Web-Anwendungen aus dem Apache Projekt eingesetzt. Der Apache Tomcat wird in der Version 6.0.16 verwendet.

Apache Struts2[32] bietet mit dem Model-View-Controller (MVC) Konzept ein Framework für Web-Anwendungen. Struts2 wird von Muradora eingesetzt. Analog dazu erfolgt die Umsetzung der eigenen Entwicklung auch mit Struts2.

Zur Verwaltung einer Ontologie für Objekttypen wird das Jena[33] Framework in der Version 2.5.7 verwendet.

Eine Übersicht dieser Technologien und weiterer, notwendiger beziehungsweise eingesetzter Technologien können dem Anhang D entnommen werden.

Im nächsten Abschnitt wird der Entwurf für das System vorgestellt.

[28] http://java.sun.com/javase/6/ (2009-03-18)
[29] http://www.fedora-commons.org/ (2009-03-18)
[30] http://www.muradora.org/ (2009-03-18)
[31] http://tomcat.apache.org/ (2009-03-18)
[32] http://struts.apache.org/2.x/ (2009-03-18)
[33] http://jena.sourceforge.net/ (2009-03-18)

## 5.2 Entwurf

Dieser Abschnitt beschreibt die wesentlichen Teile des Entwurfs für das System zur Integration von digitalen Informationsobjekten und Workflows. Dabei unterteilt sich dieser Abschnitt in weitere Unterpunkte, die der Architektur aus Abb. 4.1 entsprechen.

Mit einem generellen Blick auf die Architektur kann für den Entwurf und die Implementierung die Verwendung von Web Services für die Trennung von Repository, Auth und den anderen Systemen festgehalten werden.

Eine Trennung der Frontends von dem Workflow-System beziehungsweise dem System für Objekttypen mit Hilfe von Web Services scheint für den entwickelten Prototypen zunächst nicht praktikabel. Auf Grund der engen Beziehung zwischen der Benutzungsschnittstelle des Workflow-Systems und des Frontends wird für den Prototypen auf eine Trennung durch Web Services verzichtet. Eine solche Trennung ist im Rahmen der Anwendungsumgebung und der Anforderungen im Weiteren auch nicht explizit vorgesehen.

Die Trennung der Systeme erfolgt hier daher in Form von definierten Schnittstellen. Der Einsatz von Web Services für die Trennung der Systeme wird durch ihre Struktur und der Ausrichtung nach den Grundsätzen der SOA durch den Prototypen dennoch gestattet. Als über diese Arbeit hinausgehende, weiterführende Frage könnte der Einsatz von Web Services für die Verwendung eines Workflow-Systems mit mehreren Frontends diskutiert werden.

### 5.2.1 Anbindung an das Repository

In der gezeigten Architektur übernimmt der DigitalObjectManager die Anfrage des Repository Servers, hier Fedora, und ermöglicht mit weitergehenden Diensten wie beispielsweise der Indexierung von Metadaten die performante Anfrage von digitalen Objekten. Somit abstrahiert der DigitalObjectManager von der tatsächlichen Implementierung des Zugriffs auf das Repository oder andere Dienste.

Abb. 5.1 zeigt als Klassendiagramm den Entwurf zum DigitalObjectManager. Das Paket `digitalobject` enthält dabei im Wesentlichen die Klasse `DigitalObject-Manager`, welche zum Auffinden und Speichern von digitalen Objekten verwendet wird. Die Klasse `FedoraClient` ermöglicht es den Fedora Web Service aufzurufen und digitale Objekte zu laden. Des Weiteren stellt diese Klasse den Zugriff auf den Solr Web Service bereit und erlaubt damit die Anfrage von indexierten Metadaten, die in den Klassen `SearchResults` und `SearchResult` gehalten werden.

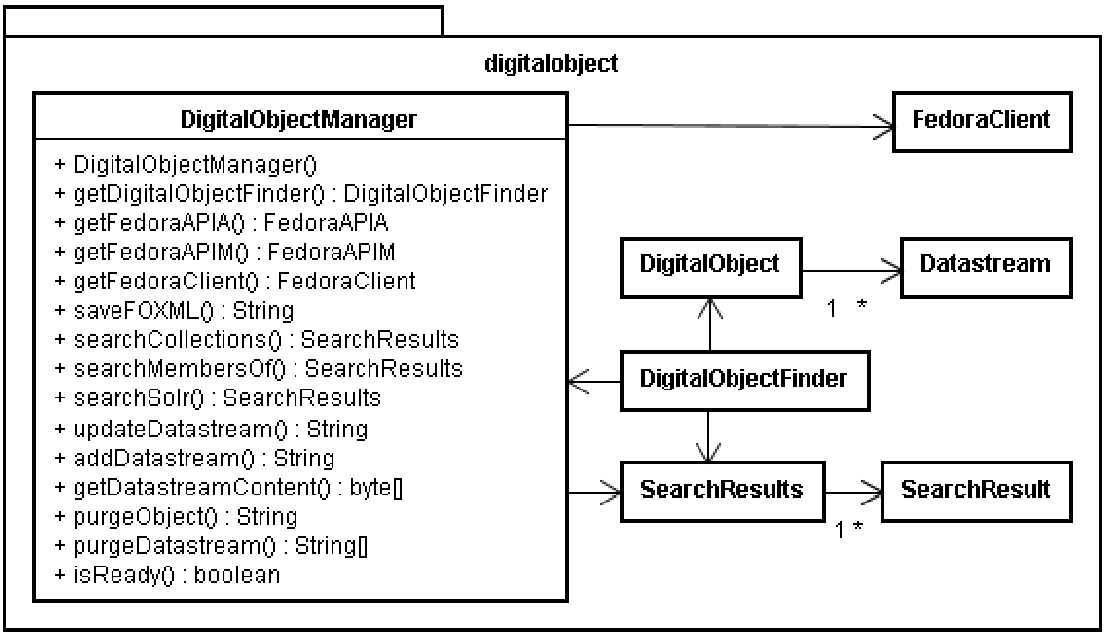


**Abb. 5.1: Klassendiagramm des Pakets digitalobject**[34]

Die weiteren Klassen kapseln `DigitalObjects` und `Datastreams` oder bieten eine Zusammenstellung der Suchergebnisse mit dem `DigitalObjectFinder` an.

Im Folgenden wird mit Abb. 5.2 der Aufbau von digitalen Objekten gezeigt, wie er im Szenario dieser Arbeit zum Einsatz kommt. Ein digitales Objekt enthält dabei eine Reihe von Metadaten zur Beschreibung von Dokumenten im Kontext des AWI und weitere Informationen, die für Objekttypen und Workflows von Interesse sind.

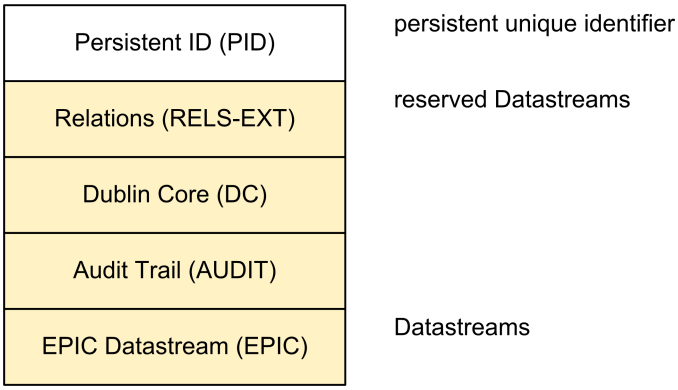


**Abb. 5.2: Aufbau eines digitalen Objektes im Szenario**[35]

Im Vergleich zum Aufbau von digitalen Objekten im Fedora Modell (siehe Abschnitt 2.5) ist ein neuer Datastream mit dem Namen *EPIC* eingeführt worden. Dieser Datastream enthält die Metadaten eines digitalen Objektes mit Bezug zur Anwendungs-

[34] Quelle: eigene
[35] Quelle: eigene

umgebung des AWI[36]. Dabei wird eine Versionierung des neuen Datastreams unterstützt, so dass beispielsweise auch die Bearbeitung eines digitalen Objektes nachvollzogen werden kann.

Listing 5.1 zeigt die Zuordnung eines digitalen Objektes mit der Id `epic:500059` zu einer Collection `epic:1`.

```
<foxml:digitalObject PID="epic:500059" ...>
  ...
  <foxml:datastream CONTROL_GROUP="X" ID="RELS-EXT" STATE="A"
    VERSIONABLE="true">
    ...
    <rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:rel="info:fedora/fedora-system:def/relations-external#">

      <rdf:Description rdf:about="info:fedora/epic:500059">
        <rel:isMemberOf rdf:resource="info:fedora/epic:1"/>
      </rdf:Description>
    </rdf:RDF>
    ...
  </foxml:datastream>
  ...
</foxml:digitalObject>
```

**Listing 5.1: RELS-EXT Datastream für Beziehungen zwischen digitalen Objekten**

Der RELS-EXT Datastream enthält die Möglichkeit zur Angabe von Collections und Member. Collections bilden dabei eine obere Ordnungseinheit für Publikationen, Member werden als Mitglied einer solchen Ordnungseinheit betrachtet. Als Ordnungseinheit sind fiktive Einordnungen oder auch Einordnungen im Sinne einer Organisation, wie beispielsweise einer Abteilung, denkbar.

Ein digitales Objekt zeichnet sich in der Anwendungsumgebung weiterhin durch den zuvor schon angesprochenen EPIC Datastream aus. Dieser Datastream enthält eine XML Repräsentation zur Speicherung von Metadaten eines digitalen Objektes. Als Erweiterung enthält dieser Datastream außerdem Informationen über den Status eines digitalen Objektes, wie er im Workflow verwendet wird. Im Folgenden wird ein beispielhafter, nicht vollständiger Ausschnitt dieses Datastreams vorgestellt.

Das Listing 5.2 zeigt einen EPIC Datastream, welches ein XML Dokument mit verschiedenen Elementen enthält. Das XML Dokument richtet sich nach dem XML Schema, das für die Anwendungsumgebung entwickelt wurde.

[36] Diese Metadaten entsprechen dem Epic XML Schema unter http://fedora.awi.de/schema/epic.xsd (2009-03-18) in der Version 2008-12-12.

Das Element `citation` ermöglicht die Angabe von Metadaten zu einem Dokument, welches mit dem digitalen Objekt assoziiert wird. Dabei enthält das Element über den Dublin Core Standard hinausgehende und organisationsspezifische Daten.

```
<foxml:digitalObject PID="epic:500059" ...>
  ...
  <foxml:datastream CONTROL_GROUP="X" ID="EPIC" STATE="A"
    VERSIONABLE="true"> ...
    <epic:EPIC-PublicationObject
      xmlns:epic="http://fedora.awi.de/schema/EPIC" ...>
      <epic:citation>
        <epic:creator>
          <epic:role>author</epic:role>
          <epic:creatorName>Koppe, Roland</epic:creatorName>
          <epic:affiliation>Universität Oldenburg</epic:affiliation>
        </epic:creator>
        <epic:year>2009</epic:year>
        <epic:title>Evaluierung eines Framesworks für das
          Workflowmanagement zur Pflege digitaler
          Informationsobjekte</epic:title>
        <epic:genre>text, diploma</epic:genre>
        <epic:pages>ca. 100</epic:pages>
        ...
      </epic:citation>
      <epic:record>
        <epic:recordAffiliation>Uni Oldb.</epic:recordAffiliation>
        <epic:recordStatus>request</epic:recordStatus>
        <epic:recordContactStaff>roko</epic:recordContactStaff>
        <epic:recordContentModel>Diploma</epic:recordContentModel>
      </epic:record>
      <epic:orgUnit>
        <epic:AwiUnit>
          <epic:AwiUnitID>awi-2005-0504</epic:AwiUnitID>
          <epic:AwiUnitName>Computing and Data Centre,
            Rechenzentrum und Datenbanken</epic:AwiUnitName>
        </epic:AwiUnit>
      </epic:orgUnit>
      ...
    </epic:EPIC-PublicationObject> ...
  </foxml:datastream>
  ...
</foxml:digitalObject>
```

**Listing 5.2: Metadaten eines digitalen Objektes im Szenario**

Das Element `record` erlaubt die Beschreibung von Informationen über das digitale Objekt. Im Wesentlichen ist der `recordStatus` und das `recordContentModel` für diese Arbeit interessant. Der `recordStatus` beschreibt den Status des digitalen Objektes in einem Workflow (vgl. Abschnitt 3.6). Das `recordContentModel` des digitalen Objektes verweist auf den dazugehörigen Objekttypen (vgl. Abschnitt 3.7).

Als weiteres Element ist `orgUnit` von Interesse. Dieses Element ermöglicht die Angabe von möglicherweise mehreren Organisationseinheiten, welche mit dem

digitalen Objekt assoziiert werden. Als Beispiel kann ein Artikel in einer bestimmten Abteilung veröffentlicht werden. In Bezug auf den Workflow des Szenarios ist die Organisationseinheit in soweit relevant, als dass ein Kurator dieser Organisationseinheit angehören muss, um ein digitales Objekt in den Status `published` zu versetzen.

### 5.2.2 Workflow-System

Zum Workflow-System zählen der Architektur nach Workflow-Manager, Workflow-Loader und Workflow-Instanzen. Diese Bestandteile finden sich gleichsam im Entwurf wieder.

Abb. 5.3 zeigt den Aufbau des Pakets `workflow` mit der Klasse `WorkflowManager`. Der `WorkflowManager` dient als zentrale Instanz zur Verwaltung von Workflows und Instanzen dieser Workflows. Das Singleton[37] Muster stellt sicher, dass der `WorkflowManager` nur einmal instanziert werden kann. Durch dieses Vorgehen steht der `WorkflowManager` in der Anwendung einmal und einzigartig zur Verfügung und kann damit die Steuerung von Workflow-Instanzen übernehmen.

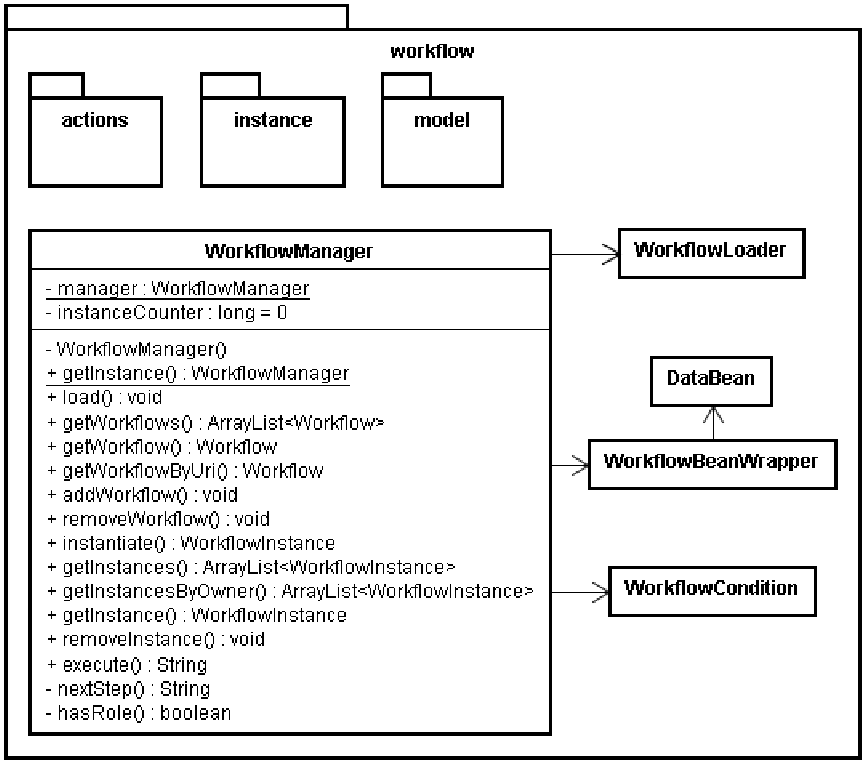


**Abb. 5.3: Klassendiagramm des Pakets workflow[38]**

[37] vgl. Gamma u. a. 1995
[38] Quelle: eigene

Um Daten zu verarbeiten, die während eines Workflows anfallen und verwendet werden sollen, werden hier `DataBeans` angeboten. Solche `DataBeans` stehen jeweils für eine Workflow-Instanz zur Verfügung und können beliebige Daten speichern.

Weiterhin werden die Anforderungen aus Abschnitt 3.3, wie beispielsweise die Möglichkeit zur Einschränkung von Transitionen für bestimmte Bedingungen, durch die Klasse `WorkflowCondition` umgesetzt.

Um den Aufbau beziehungsweise den Ablauf im Workflow-System besser verstehen zu können, zeigt Abb. 5.4 ein Sequenzdiagramm dazu.

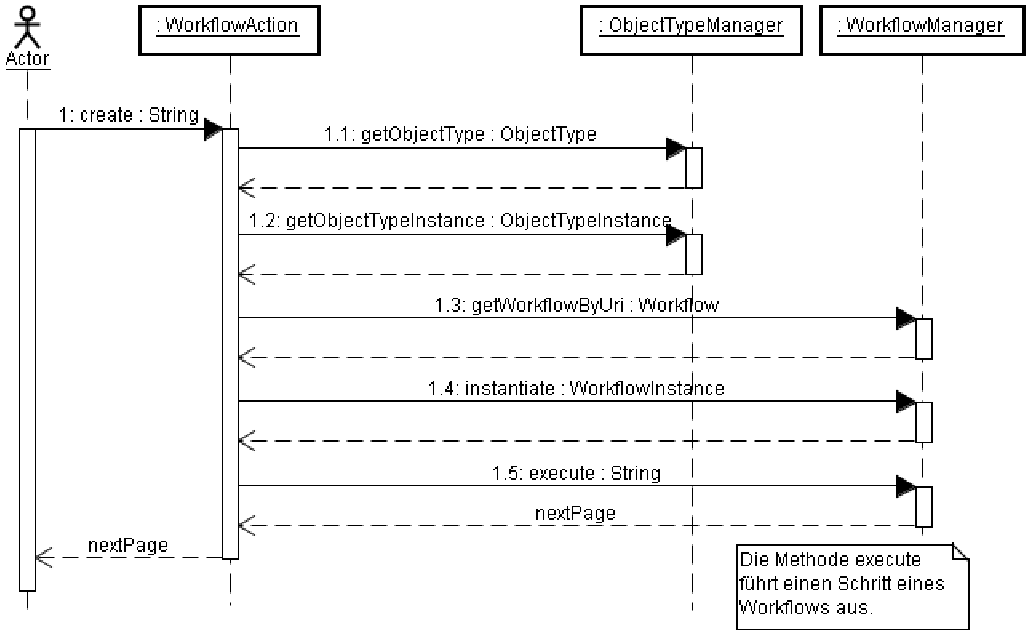


**Abb. 5.4: Sequenzdiagramm WorkflowAction**[39]

Ein Workflow wird gestartet, indem ein Akteur die Anweisung zum `create` eines Workflows gibt. Dabei ist es zunächst unerheblich, ob der Akteur einen menschlichen Benutzer oder ein anderes System darstellt.

Die `WorkflowAction` übernimmt hier die Koordination des Erstellens einer neuen Workflow-Instanz. Dabei verwendet die `WorkflowAction`, als *Action* im Sinne des Struts2 Frameworks, Informationen aus der Session[40] eines Benutzers. Hier bestimmt die `WorkflowAction` aus der Session den vom Benutzer gewählten Objekttypen und eine Instanz eines Objekttypen sowie die Rollen eines Benutzers.

[39] Quelle: eigene
[40] Eine Session ermöglicht einer Web-Anwendung die Verwaltung von Daten über eine Sitzung eines Benutzers.

Der Objekttyp ist entsprechend Abschnitt 3.7 eine Klasse der Ontologie für digitale Informationsobjekte. Eine Instanz des Objekttyps besitzt nun einen Verweis auf den mit einem Objekttyp verbundenen Workflow als URI. Schließlich wird die Workflow-Definition mit der Methode `getWorkflowByUri` geladen und dann mit den Benutzerinformationen instanziert. Sollte die Workflow-Definition noch nicht im `WorkflowManager` vorliegen, so wird der `WorkflowLoader` (vgl. 4.1.2) diese Definition laden. Der nächste Schritt führt die Instanz des Workflows aus, bis eine Benutzerinteraktion notwendig wird. Dabei werden der `execute` Methode alle wichtigen Informationen zu Session und Parametern der Benutzerinteraktion mitgegeben. Klassische Parameter sind hier beispielsweise Formulardaten. Die Notwendigkeit einer Interaktion wird über die Rückgabe von `nextPage` realisiert, welche den Namen der nächsten Seite zur Benutzerinteraktion bestimmt, die dem Benutzer präsentiert werden soll.

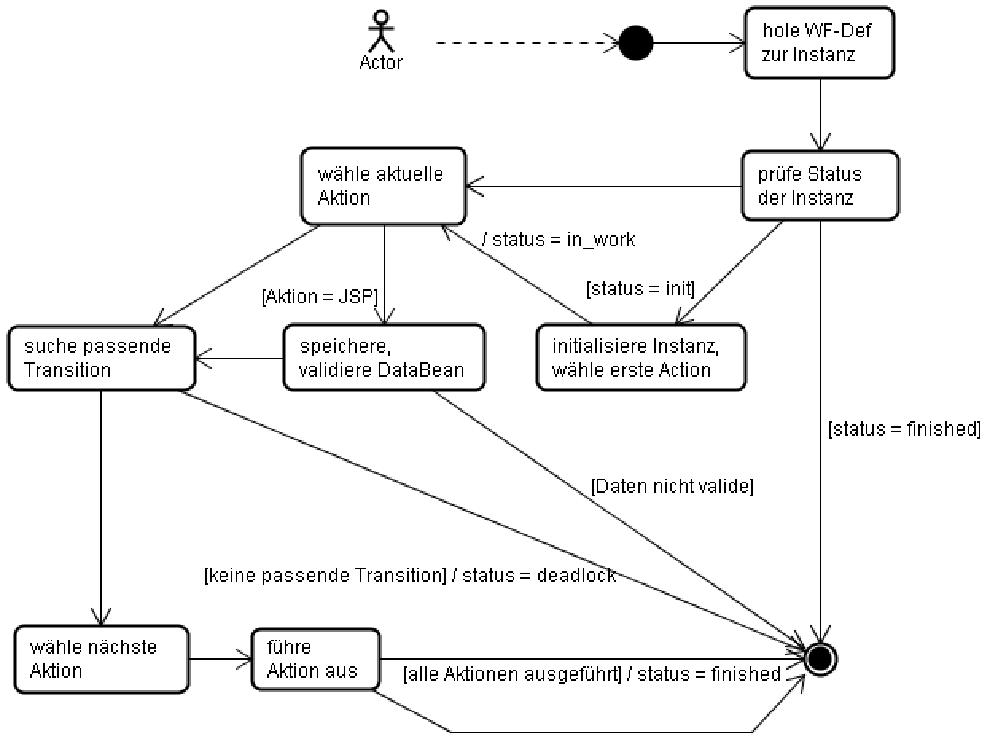


**Abb. 5.5: informelles Aktivitätsdiagramm zur Ausführung eines Workflows**[41]

Die Fortsetzung einer Workflow-Instanz erfolgt recht analog zu dem zuvor genannten Ablauf zur Erstellung einer Workflow-Instanz. Die `WorkflowAction` erhält dazu im Wesentlichen einen Verweis auf die Workflow-Instanz des Benutzers und führt dann wieder die `execute` Methode des WorkflowManagers aus.

[41] Quelle: eigene

Die Logik zum Ablauf einer Workflow-Instanz ist im `WorkflowManager` untergebracht. Die angesprochene `execute` Methode erhält als Argumente die Workflow-Instanz, Benutzerinformationen, Daten aus der Session und weitere Parameter. Im Folgenden wird die Logik informell als Aktivitätsdiagramm dargestellt.

Das Aktivitätsdiagramm in Abb. 5.5 zeigt verschiedene Aktivitäten, die während eines Schrittes eines Workflows ausgeführt werden. Ein Akteur, hier die `WorkflowAction`, leitet auf den Startzustand. Zunächst wird zu einer Workflow-Instanz die Workflow-Definition geholt, um mit dieser den Ablauf des Workflows bestimmen zu können.

Sind die entsprechenden Aktivitäten dieser Workflow-Logik durchgeführt, so befindet sich das Aktivitätsdiagramm im Endzustand. Dieser Endzustand kann gleichsam als Startzustand betrachtet werden und kennzeichnet die nächste Iteration dieser Logik, solange der Status der Workflow-Instanz nicht als `finished` markiert ist. Entspricht während eines Laufes die letzte auszuführende Aktion einer Aktion, welche die Benutzerinteraktion erfordert, so wird die Ausführung des Workflow unterbrochen und dem Benutzer eine Seite zur Interaktion angeboten. Dieses entspricht dem oben angegebenen `nextPage`.

Das Aktivitätsdiagramm zeigt verschiedene Zustände. Der Status `deadlock` gibt an, dass der Benutzer mit seinen Rechten beziehungsweise mit denen ihm zugewiesenen Rollen den aktuellen Workflow nicht weiter bearbeiten kann. Zu diesem Zeitpunkt können aber zuständige Benutzer mit mächtigeren Rollen den Workflow fortsetzen. Ein Beispiel dafür zeigt das Szenario aus Abschnitt 3.5 mit den Rollen „creator“ und „curator“.

```
<struts>
  <package name="workflow" extends="struts-default">

    <!-- WorkflowAction -->
    <action name="obwo*"
      class="de.object_workflow.workflow.actions.WorkflowAction">
      <result name="success">${page}</result>
      <result name="error">workflow/workflow_error.jsp</result>
      <result name="*">workflow/workflow_error.jsp</result>
    </action>

    <!-- Actions der Web-Anwendung... -->
  </package>
</struts>
```

**Listing 5.3: Einbindung des Workflow-Systems in Struts2**

Die Einbindung des Workflow-Systems in eine Web-Anwendung kann auf recht einfache Weise erfolgen. Auf Grund der in der Anwendungsumgebung zu verwendenden Technologien (vgl. Abschnitt 5.1) wird hier beispielhaft die Einbindung

in das Struts2 Framework gezeigt. Entsprechend dem Listing 5.3 erfolgt die Einbindung in das von Struts2 vorgesehene XML Dokument struts.xml, welches die Informationen zu Actions beinhaltet. Das Listing zeigt ein Mapping der Aktion `obwo` auf die `WorkflowAction`. Die `WorkflowAction` selbst gibt bei einem erfolgreichen Schritt des Workflows `success` zurück, welches dann auf eine Seite verweist, die mit dem Schritt des Workflow assoziiert wird.

Die Verwendung des Workflow-Systems ist auch für Web-Anwendungen möglich, in denen nicht Struts2 zum Einsatz kommt. So muss bei Bedarf lediglich die `WorkflowAction` geeignet ersetzt und ein entsprechendes Mapping vorgenommen werden, wenn mit anderen Frameworks wie beispielsweise JavaServer Faces[42] (JSF) gearbeitet werden soll.

### 5.2.3 Verwaltung von Objekttypen

Objekttypen werden in dieser Arbeit als Prototypen für digitale Informationsobjekte betrachtet. Auf Grundlage einer Ontologie können Abhängigkeiten und Beziehungen zwischen Objekttypen abgebildet werden (vgl. Abschnitt 3.7).

Als wesentlich bei der Verbindung von Objekttypen und Workflows stellt sich die Hierarchie von Objekttypen heraus. Damit diese Hierarchie für Workflows genutzt werden kann, wird sie hier durch den `ObjectTypeManager` bereitgestellt. Der `ObjectTypeLoader` ermöglicht das Laden einer Ontologie aus einem OWL Dokument.

Abb. 5.6 zeigt die Klassen für Objekttypen. Dabei dient der `ObjectTypeManager` nach dem Singleton Muster als zentrale Instanz zur Verwaltung von Objekttypen. Neben der Bereitstellung einer Hierarchie von Objekttypen kann der `ObjectType-Manager` die Anfrage von Objekttypen und die Inferenz auf Objekttypen über das OWL Model (`getOntModel`) erlauben.

Als Eigenschaften für eine Instanz eines Objekttyps sind hier zunächst Id, Name und Workflow vorgesehen. Erweiterungen dieses Prototyps könnten weitere Informationen zu Metadaten und zu der Gestaltung von Formularen für Metadaten beinhalten, um diese dem Benutzer geeignet darzustellen.

[42] http://java.sun.com/javaee/javaserverfaces/ (2009-03-18)

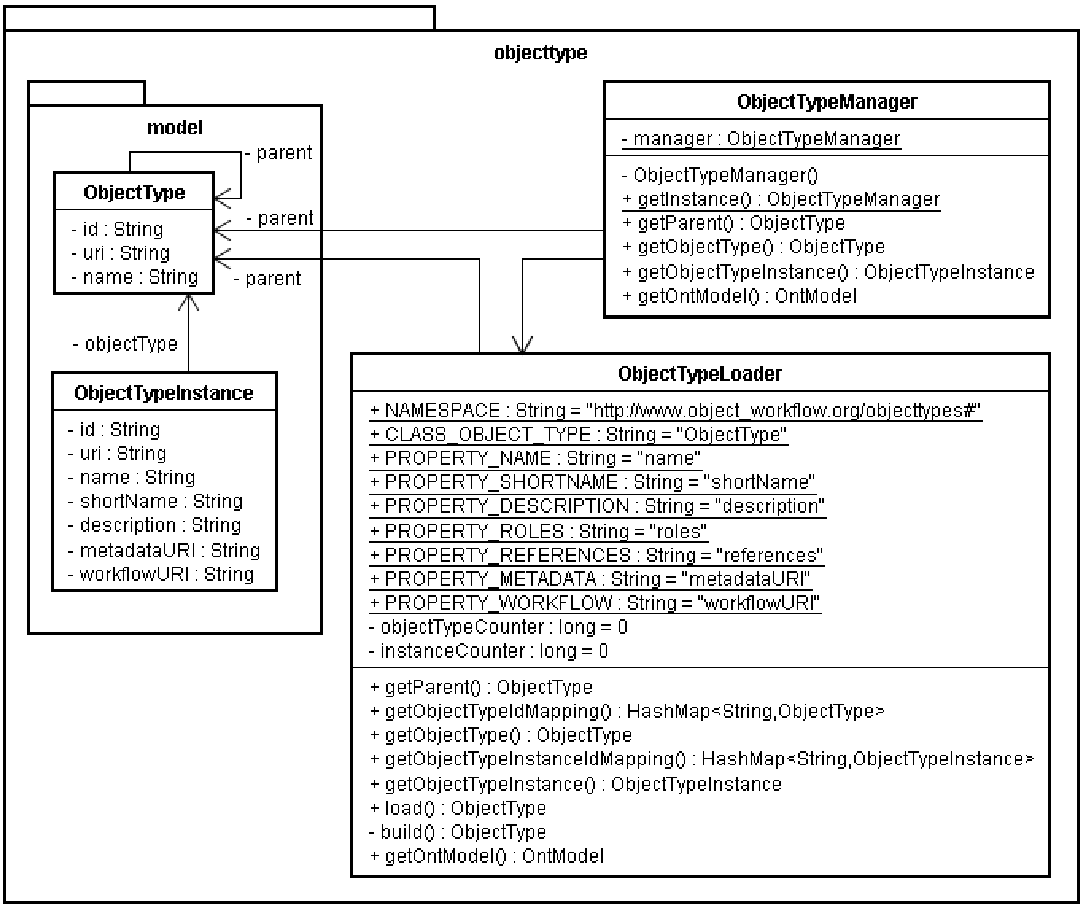


**Abb. 5.6: Klassendiagramm Objekttypen**[43]

### 5.2.4 Rechte und Rollen

Der Bereich der Rechte und Rollen, der Entwurf zum Teilsystem Auth, erfordert die Auseinandersetzung mit den gestellten Anforderungen. In Abschnitt 3.1 wurde bereits eine Idee entwickelt, welche Abhängigkeiten zwischen Rechten beziehungsweise Rollen und digitalen Objekten oder Workflows existieren können.

Mit Blick auf die Anwendungsumgebung wurde ein Schema entwickelt, welches die Definition von Rollen für eine Anwendungsumgebung, Organisationseinheiten und Objekttypen ermöglicht. Dabei wird eine Rolle einem Benutzer als Attribut zugeordnet. Die entsprechende Zuordnung von Benutzern und Rollen wird in der Anwendungsumgebung in einem LDAP Server zur Verwaltung von Benutzern hinterlegt.

[43] Quelle: eigene

Die Verwendung eines Uniform Resource Name (URN) erlaubt die dauerhafte und identifizierende Beschreibung einer solchen Rolle. Der generelle Aufbau einer URN wird in Moats 1997 beschrieben.

Abb. 5.7 zeigt den Aufbau einer Rolle mit einem Beispiel.

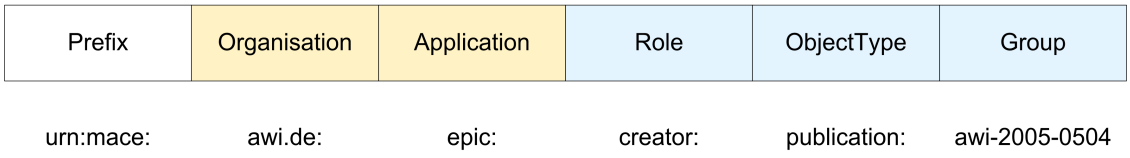


**Abb. 5.7: Schema für Rollen**

Eine Rolle setzt sich demnach zunächst aus einem Präfix, einer Organisation und einer Anwendung zusammen. Der Präfix und die Bezeichnung der Organisation ermöglichen die eindeutige Zuordnung einer Rolle zu Organisationen. Dazu bietet das Middleware Architecture Committee for Education (MACE) ein Register[44] für Organisationen an. Die Definition einer Anwendung ermöglicht es, eine Rolle innerhalb einer Organisation mit einem bestimmten Anwendungsgebiet festzulegen.

Als weitere Bestandteile der URN kommen die eigentliche Rolle, der Objekttyp und die Gruppe zum Tragen. Die Rolle identifiziert eine Rolle in der Anwendung Epic, wie beispielsweise „creator“, „curator“ und „admin“. Der Objekttyp bezieht sich auf die in Abschnitt 3.7 angesprochenen Objekttypen. Schließlich gibt die Gruppe die Organisationseinheit an, für welche die Kombination aus Rolle und Objekttyp gilt.

Als spezielles Schlüsselwort für den Objekttyp und die Organisationseinheit wird der Name „all“ eingeführt. Bei der Verwendung dieses Schlüsselwortes bezieht sich die Rolle auf alle Objekttypen beziehungsweise auf alle Organisationseinheiten.

Das in Abb. 5.7 dargstellte Beispiel einer Rolle ermöglicht es demnach einem Benutzer innerhalb des AWI (*Organisation*) und der Anwendung Epic (*Application*) als creator (*Role*) zu agieren. Dabei darf der Benutzer nur auf dem Objekttyp *publication* arbeiten. Digitale Informationsobjekte dieses Objekttyps dürfen von dem Benutzer nur in der Abteilung (*Group*) *awi-2005-0504* erstellt werden.

Mit dem beschriebenen Schema wird ein mächtiges Konzept für die Einschränkung von Benutzerrechten durch Rollen angeboten, welches auch über die Grenzen einer Anwendung hinweg eingesetzt werden kann. Die Verwendung speziellerer Rechte kann innerhalb einer Anwendung realisiert werden, bei der ein Benutzer mit der Rolle „creator“ beispielsweise lediglich digitale Objekte erstellen aber nicht löschen darf.

[44] http://middleware.internet2.edu/urn-mace/urn-mace.html (2009-03-18)

## 5.3 Implementierung

Dieser Abschnitt zeigt die Umsetzung des für den Prototypen entwickelten Frontends. Auf einen Entwurf wird an dieser Stelle verzichtet, da das Frontend in dieser Arbeit nicht im Mittelpunkt steht. Der Blick wird zunächst abstrakt auf die notwendigen Funktionen gerichtet. Dann wird die Sicht auf die Benutzungsschnittstelle gelenkt, um die praktische Umsetzung dieser Funktionen zu veranschaulichen. Dabei zeigen die dargestellten Screenshots nur einen Ausschnitt des entwickelten Frontends.

### 5.3.1 Frontend für Funktionen

Als Web-Anwendung soll der Prototyp auch ein Frontend für Funktionen anbieten. Unter Funktionen sind hier solche Möglichkeiten zu verstehen, die dem Benutzer des Systems zur Verfügung stehen sollten, um mit digitalen Objekten arbeiten zu können.

Daher können grundlegende Funktionen identifiziert werden, zu denen

- die Möglichkeit zum Anmelden am System (login) und das Laden von Benutzerinformationen,
- das Schmökern (browse) durch die im Repository liegenden digitalen Objekte,
- das Ansehen (view) von digitalen Objekten und
- die Suche (search) nach digitalen Objekten im Repository gehören.

Diesen Funktionen ist bereits am Namen anzusehen, zu welchen Systemen der Architektur sie einen Bezug besitzen. Die erste Funktion besitzt eine Anbindung an das Rechte-System (Auth) und die weiteren Funktionen ermöglichen den Zugriff auf digitale Objekte durch das Teilsystem Repository.

Abb. 5.8 zeigt einen Ausschnitt einer Liste von digitalen Objekten, welche in der „ePIC Collection“ liegen, mit der Funktion browse. Die Funktion search liefert ebenso eine Liste von digitalen Objekten, welche den Suchkriterien wie beispielsweise Titel oder Autor entsprechen. Dabei greifen beide Funktionen auf den DigitalObjectManager und im Speziellen auf einen Dienst zur Indexierung zurück.

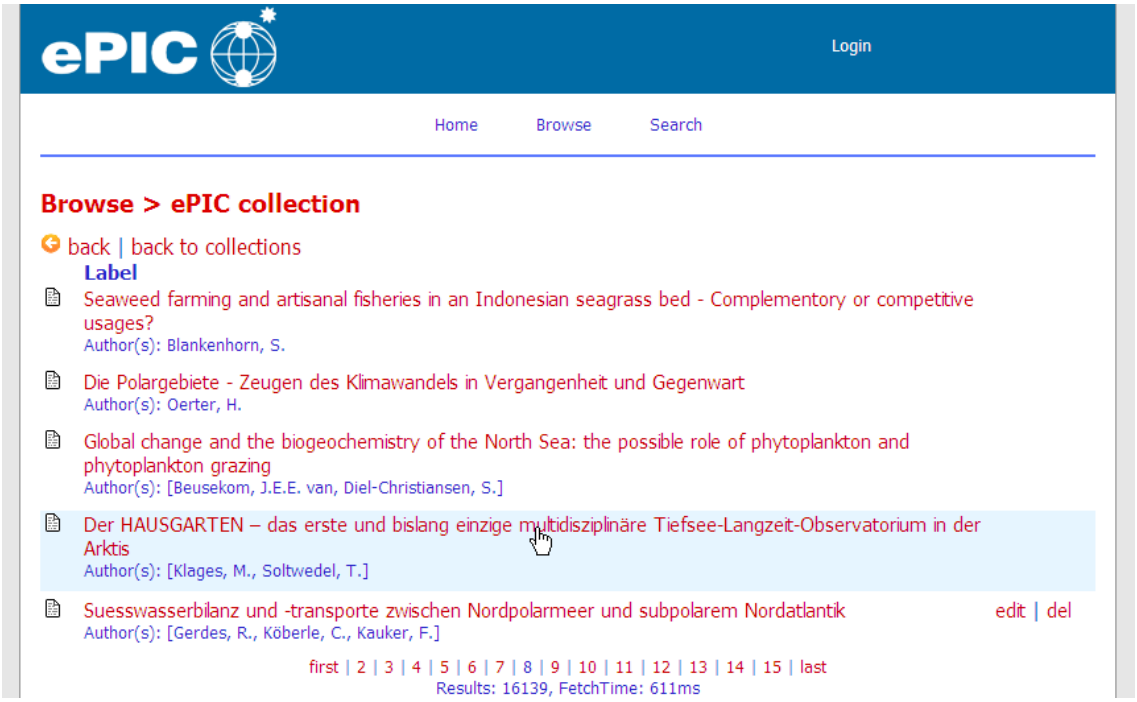


**Abb. 5.8: Screenshot der Funktion Browse**[45]

Abb. 5.9 zeigt einen Ausschnitt der Sicht auf ein digitales Informationsobjekt mit der view Funktion. Informationen zum digitalen Objekt werden in Form der PID und der Version des Objektes angezeigt. Benutzer können auf Dokumente zugreifen, die mit dem digitalen Objekt assoziiert sind.

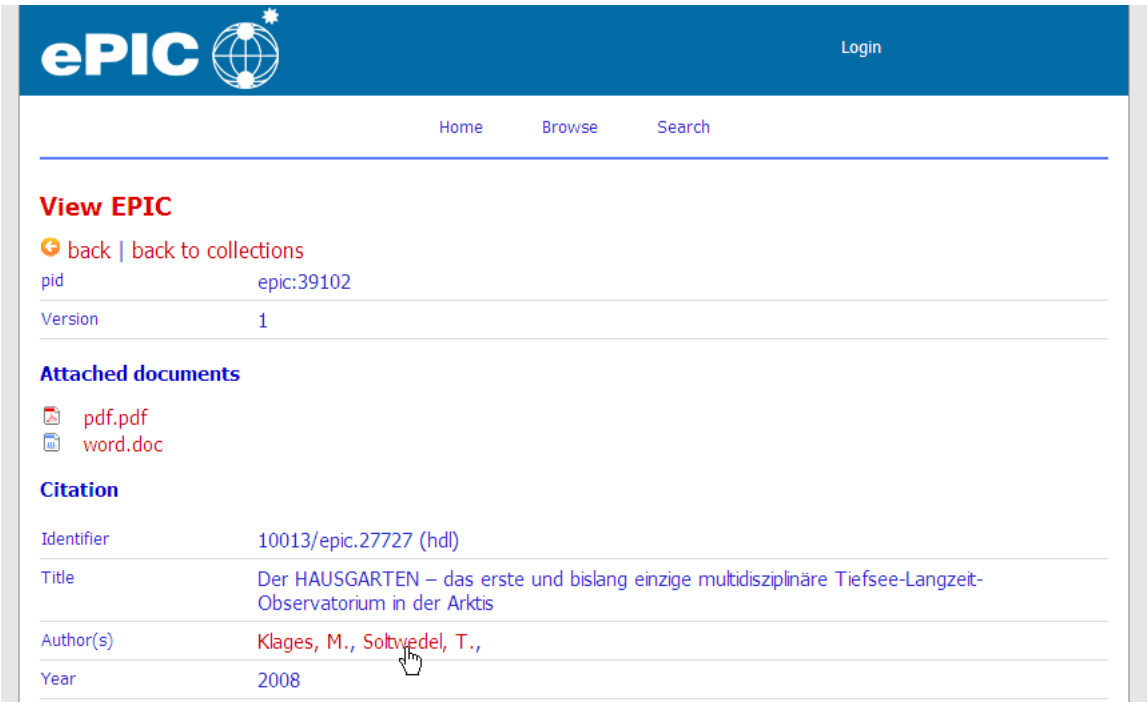


**Abb. 5.9: Screenshot der Funktion view**[46]

[45] Quelle: eigene

Bei der detaillierten Ansicht wird über den DigitalObjectManager auf das digitale Objekt im Fedora Repository zugegriffen. Der Zugriff auf das Repository liefert weitere Metadaten über das digitale Objekt, welche im Epic Datastream als XML Dokument zur Verfügung stehen.

Für die Benutzerfreundlichkeit erlaubt die Ansicht das direkte Suchen nach digitalen Objekten, die mit den angegebenen Autoren assoziiert werden.

### 5.3.2 Frontend für erweiterte Funktionen

Erweiterte Funktionen verstehen sich als Funktionen, die einen Einfluss auf Workflows besitzen oder die Steuerung von Workflows ermöglichen. Damit Benutzer diese Funktionen erreichen können, müssen sie über entsprechende Rollen verfügen (vgl. Abschnitt 5.2.4).

In den Bereich der erweiterten Funktionen fallen somit:

- Submit ermöglicht das Erstellen eines digitalen Objektes unter Verwendung eines Workflows, sowie die Beobachtung eigener Workflows,
- die Funktion publish gibt Kuratoren die Möglichkeit, diejenigen Workflows zu verfolgen und zu bearbeiten, für die sie zuständig sind, und
- Admin stellt eine Sicht für Administratoren zur Verfügung um Objekttypen und Workflows zu beobachten.

Die Erstellung eines digitalen Objektes erfolgt über die Auswahl eines Objekttyps, welcher mit einem entsprechenden Workflow verbunden ist. Abb. 5.10 zeigt die Auswahl von Objekttypen in analoger Hierarchie wie in Abschnitt 3.7 gezeigt. Dabei werden Objekttypen, für die der Benutzer keine Rechte besitzt, ausgeblendet. Dementsprechend kann ein Benutzer auch keinen Workflow starten, wenn er nicht die Rechte für den dazugehörigen Objekttyp besitzt.

Mit dem Start eines Workflows über den Objekttypen gelangt der Benutzer direkt zum ersten Schritt des Workflows. Dabei entsprechen hier die Schritte eines Workflows den Funktionen des Prozesses zur Veröffentlichung eines digitalen Objektes im Szenario. Vergleiche dazu Abb. 3.3.

[46] Quelle: eigene

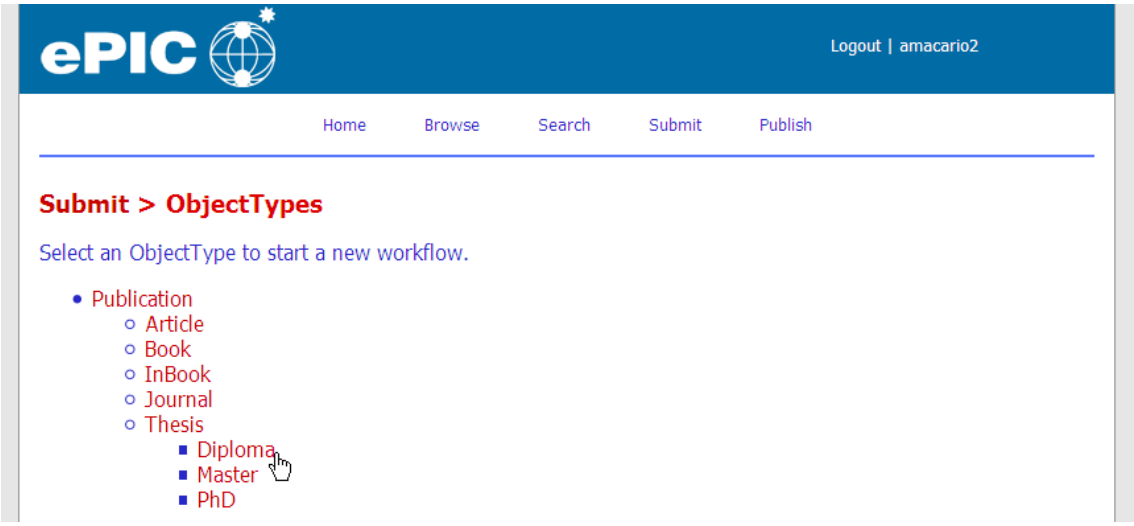


**Abb. 5.10: Screenshot Start eines Workflows über den Objekttyp**[47]

Der Prozess beginnt mit dem Hochladen von Dokumenten beliebigen Typs, die mit dem digitalen Objekt assoziiert werden sollen. Abb. 5.11 zeigt diesen Schritt.


**Abb. 5.11: Screenshot Workflow Schritt 1, Hochladen von Dokumenten**[48]

Abb. 5.12 zeigt Ausschnitte des zweiten Schritts des Workflows zur Eingabe von Metadaten. Durch den Einsatz von JavaScript können hier Felder hinzugefügt oder entfernt werden, um Daten eingeben zu können, die als multi-value ausgezeichnet sind. Multi-value Felder sind wie bei *Author* durch „+“ beziehungsweise „–“ gekennzeichnet.

Als Erweiterung der Metadateneingabe wird die Auswahl einer Organisationseinheit ermöglicht, der das digitale Objekt zugeordnet wird. Es ist zu beachten, dass ein Kurator nur digitale Objekte veröffentlichen kann, wenn er Kurator der entsprechenden Organisationseinheit ist.

[47] Quelle: eigene
[48] Quelle: eigene

Nach der Eingabe der Metadaten kann der Benutzer diese speichern. Damit wird gleichzeitig eine Anfrage zur Veröffentlichung an den zuständigen Kurator gesendet und der Workflow für den Kurator zur Verfügung gestellt.

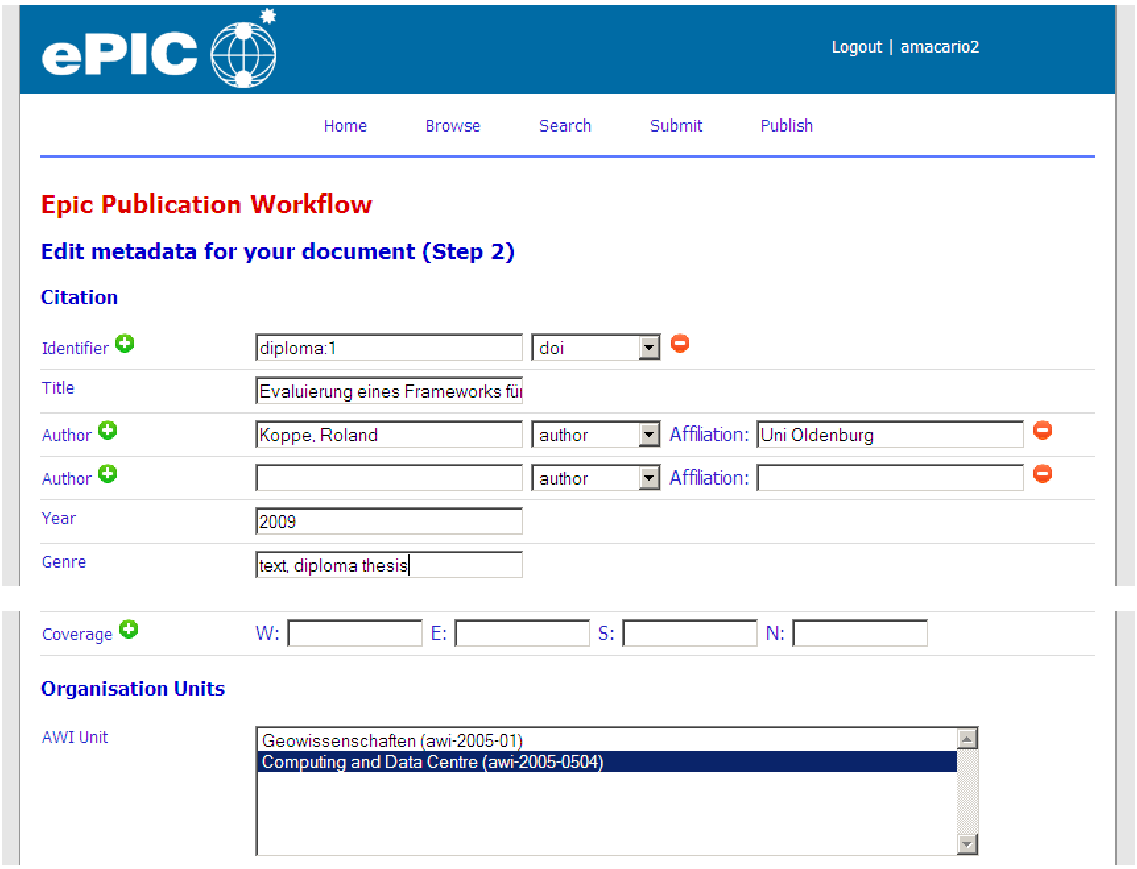


**Abb. 5.12: Screenshot Workflow Schritt 2, Eingabe von Metadaten**[49]

Der Kurator erhält über die publish Funktion Einsicht in Workflows, auf die er Zugriff hat. Abb. 5.13 zeigt die Übersicht mit den für den Kurator sichtbaren Workflows.

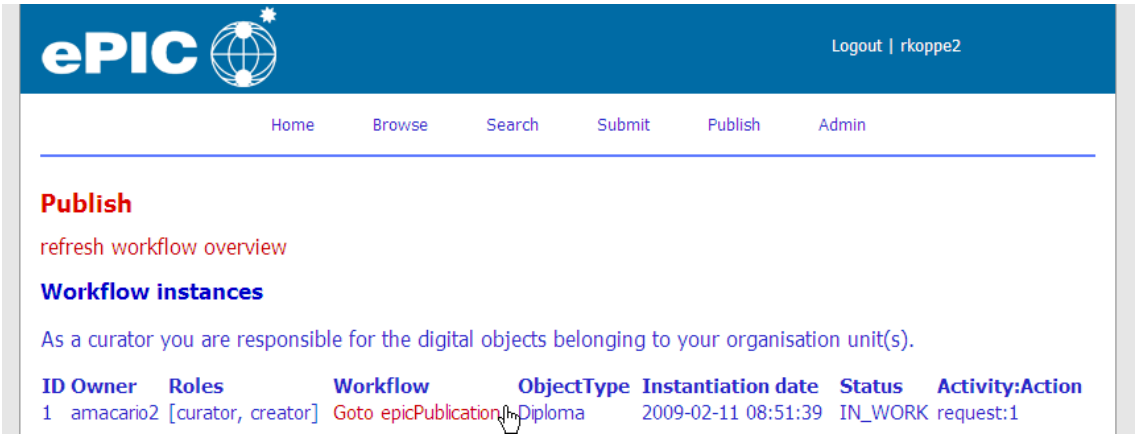


**Abb. 5.13: Screenshot der Funktion publish für Kuratoren**[50]

[49] Quelle: eigene
[50] Quelle: eigene

Befindet sich der ausgewählte Workflow im Schritt zur Bestimmung des Status, so kann der Kurator diesen bearbeiten, wie in Abb. 5.14 gezeigt wird.

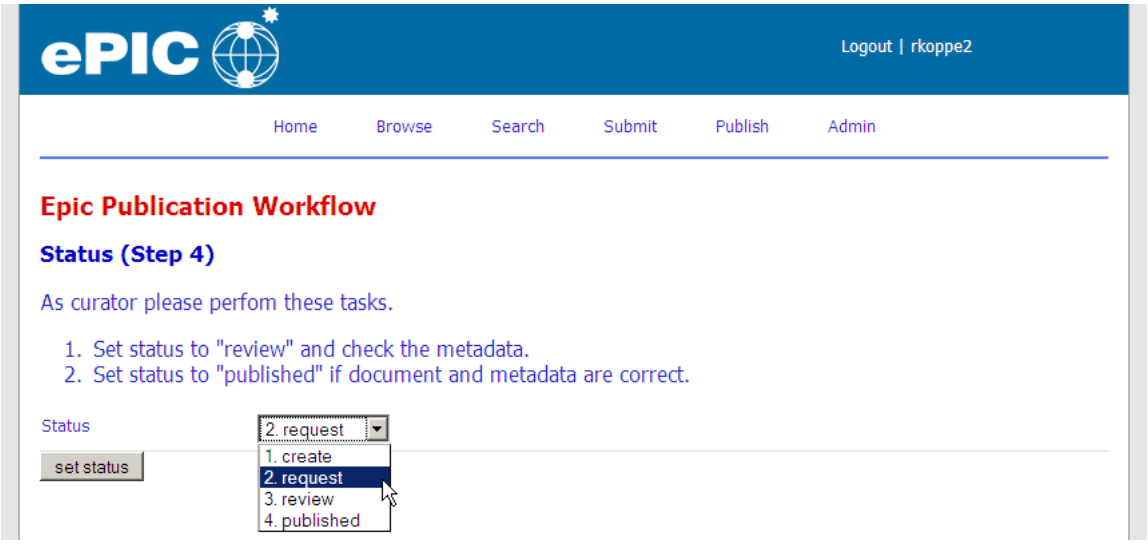


**Abb. 5.14: Screenshot Workflow Schritt 4, Setzen des Status**[51]

Mit dem Setzen des Status *published* wird das digitale Objekt, analog zur Abbildung des Prozesses in Abb. 3.3, veröffentlicht und der Workflow ist damit abgeschlossen.

[51] Quelle: eigene

# 6 Evaluierung

Der Titel der Arbeit nennt die Evaluierung eines Frameworks für das Workflowmanagement zur Pflege digitaler Informationsobjekte. In den vorherigen Kapiteln wurde eine Möglichkeit zur Integration von digitalen Objekten und Workflows aufgezeigt und eine Architektur als Grundlage für einen Entwurf und die Implementierung eines Prototypen entwickelt.

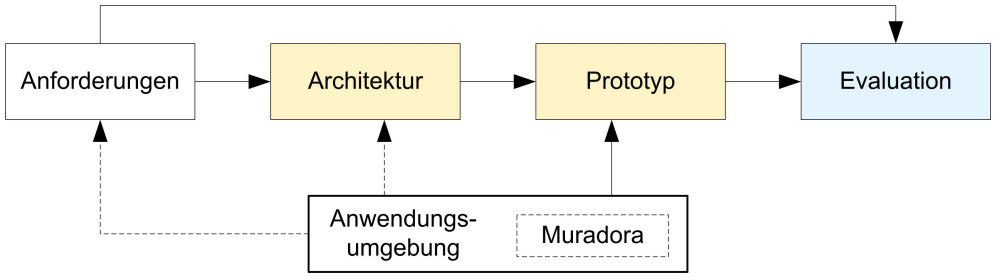


**Abb. 6.1: Prozess der Evaluierung**[52]

Abb. 6.1 zeigt das Vorgehen in dieser Arbeit. Zunächst werden die Anforderungen an ein System für die Zusammenführung von Workflows und digitalen Informationsobjekten bestimmt. Grundlegend dafür sind die Anforderungen, die sich schon aus der mit einer Integration verbundenen Problemstellung ergeben. Weitere Anforderungen werden durch Interview und Expertenbefragung – auch in Zusammenhang mit der Arbeitsumgebung – ermittelt. Die Anforderungen sind in einem Anforderungskatalog im Anhang A gesammelt.

Die nächsten beiden Schritte zeigen die Entwicklung einer Architektur und den daraus entstehenden Prototypen. Dabei haben die Anforderungen sowohl einen entsprechenden Einfluss auf die Architektur als auch auf den angeschlossenen Prototypen. Des Weiteren zeigt sich ein direkter Einfluss der Arbeitsumgebung, mit dem dort betrachteten Muradora, auf die Architektur und den Prototypen. Im Speziellen sind dieses Einflüsse, welche durch technologische Grundvoraussetzungen entstehen.

Schließlich steht zum Ende die Bewertung des Ganzen an – die Evaluation. Hier fließen die Ergebnisse der Architektur und des Prototypen ein. Weiterhin sind die Anwendungsumgebung mit Muradora sowie ihr Einfluss auf den Prototypen von Interesse. Die Ergebnisse werden in der Evaluation mit den Anforderungen verglichen und bewertet. Insgesamt ergibt sich so eine Evaluierung des Frameworks Muradora mit den hier entwickelten Ergebnissen für eine Anwendung zur Pflege digitaler Informationsobjekte mit Hilfe von Workflows.

[52] Quelle: eigene

## 6.1 Einordnung der Evaluierung

In Heinrich 2002 (S. 437ff.) werden Evaluierungsmethoden, also das Vorgehen bei einer Evaluierung beschrieben. Demnach ergeben sich drei Hauptmerkmale einer Evaluation: die Evaluierungsobjekte, das Evaluierungsziel und die Evaluierungskriterien.

Das Evaluierungsobjekt in dieser Arbeit stellt die in Abb. 6.1 gezeigte Architektur und der entwickelte Prototyp in Hinblick auf das Muradora Framework dar. Das Evaluierungsziel ist die Beantwortung der Frage, ob Workflows für die Pflege digitaler Informationsobjekte im Rahmen der Anwendungsumgebung verwendet werden können. Die Evaluierungskriterien können dabei dem aufgestellten Anforderungskatalog entnommen werden, der in Anhang A zu finden ist.

Im Folgenden werden nun die Anforderungen auf ihre Erreichung hin untersucht. Danach werden die in der Einleitung angesprochenen Fragestellungen beantwortet.

## 6.2 Evaluierung der Anforderungen

Dieser Abschnitt zeigt die an das System gestellten Anforderungen zur Pflege digitaler Informationsobjekte mit Hilfe von Workflows. Dabei beziehen sich die genannten Ergebnisse auf die in den vorherigen Kapiteln entwickelten Konzepte, die Architektur und den Prototypen mit Blick auf das Muradora Framework.

In tabellarischer Form werden im Weiteren die Anforderungen genannt und die Erreichung beschrieben. Auf der rechten Seite der Tabelle befindet sich eine Spalte, welche den Grad der Erreichung der Anforderung markiert. Das „+“ meint dabei erreicht, ein „o“ meint, dass für diese Anforderung noch weitere Arbeit investiert werden sollte, um weitere Potentiale zu ermöglichen. Ein Minus „-“ kennzeichnet einen Mangel in der Erfüllung der beschriebenen Anforderung.

Die Nennung der Anforderungen und der erzielten Ergebnisse wird durch eine weitergehende Diskussion abgerundet, welche gegebenenfalls zwischen die tabellarische Übersicht der Anforderungen und Ergebnisse gesetzt wurde.

Abschließend gibt Abschnitt 6.4 noch eine zusammenfassende Bewertung der gestellten Anforderungen und Ergebnisse.

| | | |
|---|---|---|
| 1 | Anforderungen an Workflows | |
| 1.1 | Workflows sollen in XML definiert werden | o+ |
| | In Abschnitt 3.4 und Anhang B wird eine Sprache angegeben, die es ermöglicht Workflows in XML zu beschreiben. Diese Sprache wird in dieser Arbeit von der Workflow-Engine geladen und ermöglicht die Abarbeitung von Workflows, wie der Prototyp zeigt. | |
| 1.2 | Workflows sollen die Definition von Rollen ermöglichen | + |
| | Das entwickelte XML Schema zeigt für die Workflow-Elemente Aktivität und Transition ein Attribut `role`, welches es ermöglicht eine oder mehrere, durch Kommata getrennte, Rollen zu definieren. Vergleiche dazu das Schema im Anhang B. Eine solche Rolle kann beispielsweise die Bezeichnung „curator“ sein. | |
| 1.3 | Die Ausführung eines Workflows soll Bedingungen enthalten können, die in Zusammenhang mit digitalen Informationsobjekten stehen | + |
| | In dieser Arbeit wird im Szenario und im Prototypen der Zusammenhang von Workflows und dem Status eines digitalen Objektes betrachtet. Vergleiche Abschnitte 3.5 und 3.6. Mit Hilfe von Bedingungen können Übergänge von Aktivitäten, also Transitionen, eingeschränkt werden. Dabei können sich Bedingungen auf digitale Objekte beziehen. | |
| 1.4 | Workflows sollen durch Benutzer des Systems gestartet werden können | + |
| | Damit Benutzer Workflows starten können wird ein geeignetes Frontend benötigt. Der Prototyp zeigt ein solches Frontend mit der Submit Funktion (Abschnitt 5.3.2, Abb. 5.10) zur Erstellung von digitalen Objekten. Bei der Erweiterung um thematisch andere Workflows müsste ein entsprechend geeignetes Frontend angeboten werden. Mit Blick auf die Benutzerfreundlichkeit versteckt der Prototyp Workflows, so dass dem Benutzer ein Frontend in gewohnter Weise (existierende Systeme in der Anwendungsumgebung wurden betrachtet) präsentiert werden kann. | |
| 1.5 | Workflows sollen durch Administratoren beobachtet werden können | + |
| | Das Frontend bietet auf Basis des Workflow-Systems eine Übersicht aller existierender Workflows und deren Instanzen an. Administratoren wird damit ein Überblick über Workflows und Instanzen verschafft. Dabei ist auch der Stand der Ausführung einer Workflow-Instanz einsehbar. Diese Sicht ist mit der Publish Funktion (Abb. 5.13) für Kuratoren vergleichbar, enthält jedoch noch weitere Informationen. | |

**Tab. 6.1: Anforderungen an Workflows**

Die obigen allgemeinen Anforderungen an Workflows konnten durch den Prototypen demonstriert werden. Die Definition von Workflows erfolgt über XML Dokumente. Das Workflow-System bietet in Verbindung mit dem Frontend Benutzern die Möglichkeit zur Interaktion mit Workflows. Dabei können auch Rollen von Benutzern beachtet werden und somit Einschränkungen für bestimmte Aktivitäten oder Transitionen definiert werden.

Die Sprache zur Definition von Workflows enthält in dem hier betrachteten prototypischen Status keine Möglichkeit zur Definition von Vor- und Nachbedingungen von Aktivitäten. Solche Vor- und Nachbedingungen könnten es ermöglichen, auf Definitionsebene eines Workflows weitere Einschränkungen für die Ausführung zu treffen. Sind beispielsweise bestimmte Bedingungen einer Aktivität nicht erfüllt, könnte diese unterbrochen und erst mit der Erfüllung der Bedingungen fortgesetzt werden. Für dieses Problem sollte für weitere Arbeiten eine Kosten-Nutzen-Rechnung erfolgen. Derzeit kann der Prototyp Vor- und Nachbedingungen innerhalb von Aktivitäten sicherstellen, indem diese explizit in einer Aktivität beziehungsweise einer Aktion überprüft werden.

| | | |
|---|---|---|
| 2 | Anforderungen an die Workflow-Engine | |
| 2.1 | Die Workflow-Engine soll Workflows laden können | + |
| | Für das Laden von Workflows im Workflow-System ist der hier so bezeichnete WorkflowLoader zuständig (vgl. Abschnitte 4.1.2 und 5.2.2). Der WorkflowLoader ermöglicht das Einlesen von XML Dokumenten des definierten XML Schemas zur Beschreibung von Workflows. Dabei wird eine Workflow-Definition in eine programmtechnische Struktur übersetzt, welche vom Workflow-System verarbeitet werden kann. | |
| 2.2 | Die Workflow-Engine soll Workflows instanzieren können | + |
| | Die Instanzierung von Workflows erfolgt, sobald ein Workflow gestartet wird. Diese Funktion zeigt sich im Prototypen dann, wenn Benutzer ein digitales Objekt erstellen wollen (vgl. Abb. 5.10). Eine Instanz enthält dabei Informationen zum Zustand der Ausführung eines Workflows. | |
| 2.3 | Die Workflow-Engine soll Workflows abarbeiten können | + |
| | Das entwickelte Workflow-System ermöglicht die Abarbeitung von Workflows, welche mit Hilfe der in dieser Arbeit vorgestellten Sprache definiert worden sind. Dazu wurde hier ein Szenario zur Veröffentlichung von Dokumenten (siehe Abschnitt 3.5) entwickelt, welches einen Workflow verwendet. Die Angabe des XML Schemata für die Definition von Workflows stellt sicher, dass Workflows geladen, dann instanziert und schließlich abgearbeitet werden können. Das Workflow-System ermöglicht | |

| | | |
|---|---|---|
| | die generische Definition und Abarbeitung von Workflows und ist nicht auf spezielle Workflow-Szenarien ausgelegt. Entsprechend ist zu beachten, dass der Entwickler eines Workflows für die Korrektheit von Aktivitäten und Transitionen Sorge tragen muss. | |
| 2.4 | Die Workflow-Engine soll die Interaktion von Schritten eines Workflows mit einem Benutzer ermöglichen | + |
| | Der Aufbau des entwickelten Workflow-Systems ermöglicht nicht nur die Abarbeitung von einzelnen Aktivitäten, sondern auch die Präsentation, Auswertung und Abarbeitung von Seiten zur Interaktion mit Benutzern. Der vorgestellte Entwurf (Abschnitt 5.2.2) und die Darstellungen der erweiterten Funktionen (Abschnitt 5.3.2) zeigen die Möglichkeit der Benutzerinteraktion über JSP-Seiten der Web-Anwendung. | |
| 2.5 | Technologien wie sie im „Web 2.0“ verwendet werden können, sollen in der Benutzungsschnittstelle von Workflows benutzt werden können | + |
| | Die Interaktion von Benutzern mit dem Workflow-System bei der Abarbeitung eines Workflows erfolgt im Prototypen über JSP-Seiten. Hier können auf bekannte Weise JavaScript Programme eingesetzt werden und damit auch Technologien des Web 2.0, wie beispielsweise Asynchronous JavaScript and XML (Ajax). Das Szenario zeigt bei der Eingabe von Metadaten die Verwendung von JavaScript für die Darstellung von multi-value Feldern beispielsweise zur Benennung mehrerer Autoren. | |

**Tab. 6.2: Anforderungen an die Workflow-Engine**

Die Workflow-Engine beziehungsweise das Workflow-System ermöglicht also das Laden und die Ausführung von Workflows, die mit der in dieser Arbeit vorgestellten Sprache zur Beschreibung von Workflows formuliert werden. Die Interaktion von Benutzern mit Instanzen von Workflows und einzelnen Aktivitäten wird durch das Workflow-System ermöglicht und durch den Prototypen mit der Anwendung des Szenarios demonstriert.

Die Entwicklung einer eigenen Workflow-Engine und Sprache reduzierte für diese Arbeit den Aufwand zur Analyse von bestehenden Workflow-Systemen, welche den in der Anwendungsumgebung gestellten Anforderungen entsprechen. Abschnitt 3.4 deutete die Komplexität des Themenfeld Workflow-Engines an. Außerdem sprechen auch Experten, wie die Fedora Gemeinschaft oder die Projektleitung von Muradora, für den Einsatz eines eigenen Workflow-Systems. Gründe hierfür sind in der Problemangemessenheit der Workflow-Lösung und der Nähe zu den von Fedora und Muradora gestellten Anforderungen zu erkennen.

Der Prototyp zeigt die Einbindung von Workflows in eine Web-Anwendung. Unter Nutzung eines Decoration Frameworks kann die grafische Benutzungsschnittstelle von Workflows sowie von einzelnen Aktivitäten ohne Stilbruch in die Web-Anwendung integriert werden. Diese Möglichkeit wird an den Screenshots zum Workflow des Szenarios in Abschnitt 5.3.2 veranschaulicht. Dabei kann die Verwendung eines Decoration Frameworks nicht nur als kosmetische Korrektur sondern vielmehr positiv in Richtung der Corporate Identity und der Benutzerfreundlichkeit (Usability[53]) bewertet werden.

| | | |
|---|---|---|
| 3 | Anforderungen an digitale Objekte und Objekttypen | |
| 3.1 | Objekttypen sollen als ein Prototyp für digitale Informationsobjekte verwendbar sein | o+ |
| | Objekttypen werden in dieser Arbeit als eine Art Prototyp betrachtet. Durch die erweiterbare Ontologie (siehe Abschnitt 3.7) wird die Möglichkeit gegeben Instanzen von Objekttypen unter anderem eine Struktur für Metadaten zuzuordnen, die als Prototyp betrachtet werden kann. Der entwickelte Prototyp beachtet diese Struktur allerdings nicht. | |
| 3.2 | Objekttypen sollen mit anderen Objekttypen in Beziehung stehen können | + |
| | Auf Basis der Möglichkeiten, die durch eine Ontologie, welche hier in OWL beschrieben wird, gegeben werden, wird ein standardisierter Zusammenhang zwischen Klassen (Objekttypen) angeboten. Neben einer einfachen Hierarchisierung sind auch weitere Eigenschaften wie beispielsweise inverse Abbildungen denkbar. Vergleiche dazu Abschnitt 3.7. | |
| 3.3 | Die Abfrage von Beziehungen zwischen Objekttypen soll ermöglicht werden | + |
| | Die Standardisierung von OWL und ein Angebot von Inferenzmaschinen beziehungsweise Anfragesprachen ermöglicht es, Beziehungen zwischen Objekttypen und damit auch zwischen digitalen Informationsobjekten abzufragen und in einer Anwendung weiter zu verwenden. | |
| 3.4 | Objekttypen sollen mit Workflows assoziiert werden können | + |
| | Instanzen von Objekttypen können hier verschiedene Eigenschaften besitzen. Neben der oben bereits angesprochenen Struktur von Metadaten kann eine Instanz ebenso einen Verweis auf einen Workflow enthalten. Somit können Objekttypen mit Workflows in Beziehung gesetzt werden. | |

[53] weitere Informationen zu Usability bzw. Usability Engineering bietet beispielsweise Faulkner 2000

| | | |
|---|---|---|
| 3.5 | Digitale Objekte sollen mit einem Benutzer in Zusammenhang gebracht werden können | + |
| | Die Zusammenbringung von digitalen Objekten und Benutzern erfolgt hier über die Bereitstellung von Rollen für Benutzer. Rollen ermöglichen somit die Bearbeitung von digitalen Objekten. Ein konkretes digitales Informationsobjekt kann mit Hilfe von Audit Datastreams in seiner Bearbeitung beobachtet werden. Benutzer, die ein digitales Objekt erstellen, werden als Besitzer (owner) mit dem digitalen Objekt assoziiert. | |

**Tab. 6.3: Anforderungen an digitale Objekte und Objekttypen**

Objekttypen werden hier als Prototypen für konkrete digitale Informationsobjekte betrachtet. Mit Hilfe einer Ontologie kann der Zusammenhang zwischen Objekttypen und damit auch der abstrakte Zusammenhang zwischen digitalen Objekten beschrieben werden.

Die in dieser Arbeit aufgezeigte Ontologie stellt einen Ausschnitt der Anwendungsumgebung im Szenario dar. Für eine Standardisierung einer Ontologie sollten weitere bereits existierende Ontologien[54] im Bereich der Veröffentlichung von Dokumenten beachtet werden und wenn nötig entsprechende Anpassungen der Ontologie beziehungsweise auch der Organisation vorgenommen werden. Die Verwendung eines Standards könnte dabei die Flexibilität, Austauschbarkeit und Interoperabilität von Ontologien und damit auch von Objekttypen und Objekten, über Organisationen hinweg, ermöglichen. Es ist klar festzuhalten, dass die in dieser Arbeit vorgestellte Ontologie nur als Fallbeispiel zu betrachten ist und nicht alle denkbaren Möglichkeiten von OWL im Szenario für Objekttypen ausgeschöpft werden.

Die Struktur von Objekttypen kann über Instanzen von OWL Klassen definiert werden. Angesprochene Eigenschaften sind die Angabe der Struktur von Metadaten und der Verweis auf Workflow-Definitionen. Die Struktur von Metadaten kann durch XML Schemata beschrieben werden. Im Weiteren könnten auch Definitionen über den Aufbau von Formularen zur Benutzerinteraktion mit einem Objekttyp assoziiert werden. Ein Beispiel dafür wäre der Einsatz von XForms (Boyer 2007).

Ebenso sind weitergehende Eigenschaften denkbar, welche sich auf verschiedene Schemata zur Beschreibung der Metadaten beziehen. Als Beispiele können EPIC[55], Dublin Core[56] oder auch Darwin Core[57] genannt werden. Interessant wäre an dieser

[54] bspw. in SchemaWeb http://www.schemaweb.info/schema/SchemaInfo.aspx?id=109 (2009-03-18)
[55] http://fedora.awi.de/schema/epic.xsd (2009-03-18)
[56] http://dublincore.org/ (2009-03-18)
[57] http://www.tdwg.org/activities/darwincore/ (2009-03-18)

Stelle auch ein Matching zwischen diesen Formaten, um Metadaten in ein jeweils anderes Format, beispielsweise mit XSLT, zu überführen.

OWL bietet als Sprache zur Beschreibung der hier gewünschten Beziehungen weitere Möglichkeiten zur Inferenz und Auflösung von Abhängigkeiten zwischen Objekttypen und digitalen Objekten. Konkrete Beziehungen zwischen digitalen Objekten können über den RELS-EXT Datastream eines Fedora digital object dargestellt werden. Um weitere Beziehungen abbilden zu können, ist an dieser Stelle auch die Einführung von anwendungsspezifischen Beziehungen in RDF oder OWL denkbar. Für den hier betrachten Prototypen und die genannten Anforderungen genügt derzeit in der Anwendungsumgebung allerdings RDF zur Abbildung von Collections und Members (vgl. Abschnitte 2.1, 5.2.1).

| | | |
|---|---|---|
| 4 | Anforderungen an Rechte und Rollen | |
| 4.1 | Benutzer sollen Rollen besitzen können | + |
| | In Abschnitt 5.2.4 wurde im Entwurf mit der Verwendung von URN eine Möglichkeit vorgestellt, Rollen in beliebiger Granularität für Benutzer zu definieren. Damit Benutzern solche Rollen zugeordnet werden können, bedarf es einer Möglichkeit zur Speicherung und Verwaltung solcher Rollen für Benutzer. | |
| 4.2 | Rollen sollen die Möglichkeiten zur Interaktion mit dem System einschränken | + |
| | Rollen werden in dem entwickelten Entwurf und Prototypen sowohl für das Frontend als auch für Workflows verwendet. Damit können durch Rollen die Möglichkeiten zur Interaktion mit den Teilen des Systems oder dem gesamten System eingeschränkt werden. | |
| 4.3 | Der Zugriff auf bestimmte Objekttypen soll eingeschränkt werden können | o+ |
| | Das angesprochene URN Schema erlaubt die Benennung von allgemeinen Rollen und auch die Benennung von Rollen für bestimmte Objekttypen. Somit können entsprechende Einschränkungen auch auf Ebene von Objekttypen vorgenommen werden. Weiteres im Anschluss an die Tabelle. | |
| 4.4 | Benutzerdaten und Rollen sollen in das System geladen werden können | + |
| | Im Allgemeinen ist es notwendig, Rollen und Benutzerdaten laden zu können. Der Prototyp ermöglicht das Laden von Rollen und Benutzern aus einem LDAP Server oder alternativ aus einem XML Dokument. | |

| | | |
|---|---|---|
| 4.5 | Informationen über Rollen sollen aus einem LDAP Server entnommen werden können | + |
| | Der Prototyp bietet eine Schnittstelle für LDAP Server an. Über diese Schnittstelle wird der Zugriff auf im LDAP Server abgelegte Benutzerinformationen ermöglicht. Rollen werden dabei Benutzern in Form von Attributen zugeordnet. | |
| 4.6 | Digitale Objekte sollen in Zusammenhang mit Gruppen stehen können | + |
| | Der Zusammenhang von digitalen Objekten mit Gruppen (Organisationseinheiten) wird im Epic Schema eines digitalen Objektes gespeichert. Somit kann ein konkretes digitales Objekt einer oder mehreren Gruppen zugeordnet werden. Des Weiteren erlaubt das URN Schema die Definition von Rollen für eine Gruppe. So kann der Zugriff von Benutzern auf digitale Objekte auf bestimmte Organisationseinheiten eingeschränkt werden. | |

**Tab. 6.4: Anforderungen an Rechte und Rollen**

Im Entwurf wurde ein URN Schema für die Verwaltung von Rollen für Objekttypen und Organisationseinheiten vorgestellt. Dieses URN Schema erlaubt ebenfalls die Erweiterung um weitergehende Einschränkungen. Damit liegt eine mächtige Möglichkeit vor, Rollen in bestimmter Granularität definieren zu können.

Ein Manko der Verwendung von URN Schemata liegt allerdings in der nicht normalisierten Form des Schemas. So würde nach Abb. 5.7 ein Benutzer mit der Rolle „creator“ für Publikationen für jede unterschiedliche Organisationseinheit Einträge mit dem selben Infix „creator:publication“ enthalten. Dennoch wird hier diese Einschränkung in Kauf genommen. Damit wird der Anforderung zur Anfrage von Rollen von einem LDAP Server in performanter Weise nachgekommen. Die nicht normalisierte Form erspart dabei die Auflösung oder ein Join[58] von Einzeldaten.

| | | |
|---|---|---|
| 5 | Anforderungen an die Implementierung | |
| 5.1 | Das entwickelte System soll auf das Muradora Framework anwendbar sein | + |
| | Muradora ist im Grunde eine Web-Anwendung, welche mit Hilfe von Struts das MVC Muster einsetzt. Der aus der vorgestellten Architektur und dem Entwurf hervorgehende Prototyp ist ebenfalls als Web-Anwendung mit dem Struts Framework implementiert worden. Somit ist eine gemeinsame Grundlage geschaffen, die zeigt, dass der Prototyp entsprechend auf Muradora anwendbar ist. Weiteres in Abschnitt 6.2.1. | |

[58] SQL: Ein join liefert das kartesische Produkt zweier Mengen; einen Verbund.

| | | |
|---|---|---|
| 5.2 | Die Implementierung des Systems soll in Java erfolgen | + |
| | Im Abschnitt 5.1 wurden die für diese Arbeit verwendeten Technologien genannt. Dabei bildet das Java Development Kit (JDK) die Grundlage für die Entwicklung. Die Anforderung ist somit erfüllt. | |
| 5.3 | Die Grundprinzipien der SOA sollten beachtet werden | o+ |
| | Sowohl die hier entwickelte Architektur, als auch die Architektur von Muradora, setzt auf den Einsatz von Web Services. Web Services ermöglichen, den Grundlagen nach, die Bildung einer service-orientierten Architektur. Weiter können die einzelnen Systeme des Prototyps in einer Weise zerlegt werden, welche den Grundprinzipien der SOA nachkommen.<br><br>Die Workflow-Definition sieht für einzelne Aktivitäten beziehungsweise Aktionen eine Schnittstelle für die Verwendung von Web Services innerhalb eines Workflows vor, so dass SOA auch innerhalb von Workflows verwendet werden könnte. | |
| 5.4 | Die Anwendung soll plattformunabhängig sein | o+ |
| | Der Einsatz von Java als virtuelle Maschine und Programmumgebung gestattet den Einsatz des entwickelten Systems auf vielen Computersystemen. Die Voraussetzung ist hier eben der Einsatz einer kompatiblen virtuellen Maschine für Java. Für einen produktiven Einsatz sollten entsprechende Test bezüglich der Plattformunabhängigkeit und verwendeter Systeme vorgenommen werden. | |
| 5.5 | Die Anwendung soll speziell auf einem Ubuntu Linux Server lauffähig sein | + |
| | Der Prototyp wurde auf verschiedenen Systemen entwickelt und sein Einsatz getestet. Zu diesen Systemen zählen Windows XP Professional, Linux Ubuntu und Linux Ubuntu Server, jeweils in den Versionen 8.04 und 8.10. Auch unterschiedliche Hardwarekonfigurationen (insbesondere CPU und RAM) wurden bei den Tests verwendet. Somit kann diese Anforderung positiv bewertet werden. | |
| 5.6 | Das System soll als Web-Anwendung realisiert werden | + |
| | Bereits die Anforderung 5.1 mit der Bedingung, dass das entwickelte System auf Muradora anwendbar sein soll, empfahl die Entwicklung des Prototypen als Web-Anwendung und wurde daher so umgesetzt. Die wesentlichen Systeme (Auth, Repository, Workflow-System und Objecttypes) bleiben dennoch unabhängig und erfordern nicht den Einsatz einer Web-Anwendung. | |

| | | |
|---|---|---|
| 5.7 | Als Web-Anwendung soll das System Struts2 unterstützen | + |
| | Entsprechend der vorgenannten Anforderung kommt im Prototypen Struts2 zum Einsatz, da auch Muradora Struts2 verwendet. Auch wenn für den Prototypen konkret Struts2 als MVC Muster eingesetzt wurde, so kann doch ebenso ein anderes MVC Framework, wie beispielsweise JSF verwendet werden. | |
| 5.8 | Es soll eine Möglichkeit zur Internationalisierung gegeben werden | o+ |
| | Als Web-Anwendung mit Struts2 ermöglicht der Prototyp die Internationalisierung der Web-Anwendung. Struts2 bietet dazu mehrere Möglichkeiten zur „Localization“[59]. | |

**Tab. 6.5: Anforderungen an die Implementierung**

Für die Anforderungen an die Implementierung bleibt insgesamt festzuhalten, dass diese durch den Prototypen erfüllt werden. Dabei stellt der Prototyp selbst weitergehende Möglichkeiten zur Verfügung: Die aus der Architektur hervorgehenden Systeme sind unabhängig von dem Einsatz des Prototypen als Web-Anwendung und basieren auch nicht auf einem konkreten Framework wie Struts oder JSF. Somit können die Systeme in andere Anwendungen integriert beziehungsweise mit anderen Frontends verwendet werden.

| | | |
|---|---|---|
| 6 | Anforderungen an das Szenario | |
| 6.1 | Das gewählte Szenario zeigt einen Workflow zur Erstellung eines digitalen Objektes | + |
| | Diese Anforderung wird bereits in Abschnitt 3.5 erfüllt. Schließlich wird im Verlauf dieser Arbeit aus einem Prozess zur Veröffentlichung eines digitalen Objektes ein Workflow abgeleitet. Dieser Workflow ermöglicht mit dem hier entwickelten Prototypen die Erstellung eines digitalen Objektes in einem Veröffentlichungsprozess. | |
| 6.2 | Digitale Objekte sollen ein Schema für Metadaten erhalten | + |
| | Im Workflow des Szenarios werden Metadaten von Benutzern zu einem digitalen Objekt angegeben. Diese Metadaten werden als Epic Datastream im digitalen Objekt gespeichert. Das Schema für diese Metadaten steht am AWI[60] zur Verfügung. | |

**Tab. 6.6: Anforderungen an das Szenario**

[59] http://struts.apache.org/2.x/docs/localization.html (2009-03-18)
[60] http://fedora.awi.de/schema/epic.xsd (2009-03-18)

Die Ausführungen in diesem Abschnitt zeigten im Wesentlichen die Erreichung der gestellten Anforderungen an ein System zur Pflege digitaler Informationsobjekte mit Hilfe von Workflows. Im Folgenden wird nun konkret auf Muradora und die Anwendungsumgebung eingegangen und diese bewertet.

### 6.2.1 Bewertung von Muradora und der Anwendungsumgebung

Die in dieser Arbeit entwickelte Architektur kann als generell betrachtet werden und zeigt eine Trennung von Systemen in verschiedene Zuständigkeiten. Die analysierte Architektur beziehungsweise der Aufbau von Muradora lässt sich auf geeignete Weise in die entwickelte Architektur als Frontend einbinden. Allerdings hat Muradora selbst keine solche scharfe Trennung und besitzt damit, im Detail betrachtet, leider eine recht unübersichtliche Struktur.

Ein weiteres Problem in der Struktur von Muradora liegt in der Tatsache, dass sich das Framework während des Entstehens dieser Arbeit noch in der Entwicklung befindet. Zwar existiert eine Dokumentation für den Einsatz von Muradora in Form eines Wiki[61], doch wird dort nicht weiter auf den Aufbau und den Zusammenhang von Bausteinen eingegangen. Insgesamt kann so von einem erheblichen Arbeitsaufwand für eine Einarbeitung in die Struktur und die Funktionsweise von Muradora gesprochen werden.

Da die Struktur von Muradora kein dokumentiertes Schnittstellenkonzept besitzt, werden Anpassungen (Customisation) des Muradora Frameworks bei einem Update des Frameworks hinfällig, beziehungsweise es wird eine recht aufwendige Migration notwendig. Dabei sind in der Anwendungsumgebung Updates natürlich von Interesse, um in den Genuss von Verbesserungen in Hinblick auf Fehlerverhalten und Performance zu kommen. Ein weiterer Schwachpunkt an dieser Stelle zeigt sich in der Vielzahl von Klassen, welche die Web-Anwendung beinhaltet. Einige Klassen scheinen veraltet und werden nicht mehr verwendet, andere sind von ihrer Funktion her nicht in die Abläufe der Web-Anwendung eingebunden. Hier ist dringend ein Refactoring zu empfehlen.

Muradora unterstützt Workflows nur in einer harten Form. Dabei müssen einzelne Aktionen eines Workflows und damit die Abbildung eines Geschäftsprozesses mit Hilfe von hart vernetzten und unflexiblen Methoden in die Web-Anwendung Muradora eingebunden werden. Workflows, wie sie im Laufe dieser Arbeit beschrieben wurden, können nicht in Muradora genutzt werden. Dabei existiert derzeit weder eine implizite

[61] Ein Wiki ermöglicht es Benutzern des Systems Inhalte (zum Beispiel Texte) zu veröffentlichen, zu lesen, zu bearbeiten und gemeinsam auszutauschen. Das Wiki von Muradora wird von den Entwicklern bereitgestellt und ist nicht mit dem bekannten Wikipedia zu verwechseln.

noch eine explizite Bindung zwischen Objekttypen und Workflows. Der in dieser Arbeit entwickelte Prototyp mit dem Workflow-System ermöglicht demgegenüber eine neue Flexibilität, welche durch Workflow-Definitionen strukturiert wird. Durch die in Abschnitt 4 vorgestellte Architektur wird die Möglichkeit gegeben, die hier entwickelten Systeme in Muradora zu integrieren.

Muradora bietet als Schwerpunkt eine sehr aufwändige und abstrakte Rechteverwaltung mit Hilfe von XACML an. Diese Rechteverwaltung kann sehr feingranular aber auch grob eingesetzt werden und ermöglicht damit die Einschränkung von Zugriffen auf digitale Objekte. So werden hier unter anderem auch Konzepte der Vererbung unterstützt. Dabei kommt das in Abschnitt 4.2 angesprochene MelcoePDP und MuraPEP zum Einsatz, um entsprechende Rechte über das Global Access Control zu verwalten. Rechte können für digitale Objekte und sowohl für Benutzer als auch Rollen definiert werden. Eine weitere Möglichkeit ist die Formulierung von Ausdrücken, welche den Zugriff auf bestimmte digitale Objekte, beispielsweise auf digitale Objekte mit bestimmten Ids, einschränken können.

Die große Flexibilität, die mit XACML in Muradora angeboten wird geht auf der anderen Seite zu Lasten der Performance. Einzelne Rechte, die für das Frontend von Muradora von Interesse sind, müssen über die Web Service Schnittstelle von Melcoe-PDP und MuraPEP abgefragt werden, wobei MelcoePDP dann auf eine Datenbank zur Verwaltung von Rechten zurückgreift. Bei einer lokalen Anordnung der Komponenten, also der Installation von Datenbank für XACML Dokumente, MelcoePDP und Muradora auf einem System, beziehungsweise in einem Netzwerk mit hoher Band-breite, ist die Wartezeit für die Auflösung von Rechten noch akzeptabel. Im Gegensatz dazu kommt bei einer Aufteilung der einzelnen Komponenten auf verschiedene (entfernte) Systeme ein Flaschenhals zum Vorschein. So verwendet Muradora mehrere Rechte für jeweils ein Objekt (beispielsweise create, read, publish, admin). Jedes dieser Rechte wird für jedes digitale Objekt einzeln in Form von XML Dokumenten an MelcoePDP angefragt, welches wiederum mit einem XML Dokument antwortet.

Bei der Darstellung von Listen von digitalen Objekten ergibt sich somit ein verhältnismäßig großes Volumen von Request / Response Nachrichten, welche die Wartezeit für Benutzer unakzeptabel machen. Dabei kann für die Darstellung von lediglich zehn digitalen Objekten eine Wartezeit von mehr als 10 Sekunden auftreten, in der ein Benutzer kein Feedback erhält. Man beachte dazu Grundsätze des Usability Engineering zum Beispiel bei Faulkner 2000 und Jakob Nielsen 1993. Es stellt sich hier die Frage, ob die mögliche Flexibilität von Rechten für digitale Objekte in der Anwendungsumgebung tatsächlich erforderlich ist, oder ob die Möglichkeiten über die

Grenzen der Anforderungen und Performance hinausgehen. Dieses gilt es über diese Arbeit hinaus zu untersuchen und gezielt zu bewerten.

Die Verwaltung von Rechten und Rollen verstreut sich in Fedora und Muradora selbst über verschiedene Dateien, welche auch noch in unterschiedlichen Verzeichnisstrukturen zu finden sind und nicht automatisch miteinander synchronisiert werden. Muradora bietet den Zugriff auf einen LDAP Server an, entnimmt diesem aber keine weiteren relevanten Benutzerinformationen. Des Weiteren werden die oben angesprochenen Rechte nicht mit Benutzern im LDAP Server abgeglichen, so dass Fragen offen bleiben, wie neue Benutzer in Muradora eingebunden werden können oder was mit Rechten für digitale Objekte passiert, für die kein Benutzer mehr existiert. Für die effiziente Wartung ist hier eine einheitliche Möglichkeit zur Bearbeitung von Rechten dringend anzuraten. Die mehrfache Datenhaltung sollte weitestgehend eingeschränkt werden, wenn die Vorzüge einer Replikation nicht gegeben sind.

Im Folgenden werden die Anforderungen an Muradora noch einmal tabellarisch notiert und kurz auf das Ergebnis der Bewertung angesprochen.

| | | |
|---|---|---|
| 7 | Anforderungen an Muradora | |
| 7.1 | Dokumentation: Handbuch für Muradora | o |
| | Das Muradora Projekt bietet auf seinen Internetseiten einige Dokumentation zur Handhabung und Einsatz von Muradora. Darunter fallen allerdings im Wesentlichen nur Beispiele und oberflächliche Informationen. Der Gesamtzusammenhang bleibt dabei eher undurchsichtig. | |
| 7.2 | Dokumentation: Architektur und Aufbau | - |
| | Leider bietet Muradora keine Dokumentation zur Architektur und Aufbau des Systems. Einzig die Architektur zur Beachtung von Rechten wird in Nguyen und Dalziel 2008 kurz besprochen. | |
| 7.3 | Dokumentation und Bereitstellung von Schnittstellen | - |
| | Muradora bietet keine dokumentierten Schnittstellen, die das System betreffen. Dokumentationen liegen im Wesentlichen nur für die in Muradora verwendeten oder assoziierten Softwarekomponenten vor, wie Fedora und Solr. | |
| 7.4 | Anbindung an das Fedora Repository | o+ |
| | Muradora ermöglicht als Frontend den Zugriff auf das Fedora Repository. Dabei bietet Muradora Funktionen wie browse und search über einen Indexierungsdienst an. Das Frontend enthält dabei allerdings einige Darstellungsfehler, die der Benutzerfreundlichkeit nicht zuträglich sind. | |

| | | |
|---|---|---|
| 7.5 | Benutzerfreundlichkeit des Frontends | -o |
| | Es wurde bereits angesprochen, dass das Frontend einige Fehler enthält. Der Benutzer wird durch solche Fehler bestenfalls irritiert. Ausführliche oder verständliche Fehlermeldungen werden nicht angeboten. Das Frontend bietet eine Reihe weitergehender Funktionen, wie beispielsweise der Export nach BibTeX, die für einen normalen Benutzer allerdings nicht intuitiv zu bedienen sind. Außerdem ist die Wartezeit (mehr als 10 Sekunden) für einzelne Funktionen, die auf digitalen Objekten und Rechten arbeiten, für Benutzer nicht akzeptabel. | |
| 7.6 | Verwendbarkeit von Rechten und Rollen | o+ |
| | Die Stärke von Muradora liegt in der Möglichkeit Rechte feingranular für digitale Objekte zu definieren. Dazu wird XACML verwendet um Rechte zu verwalten. Allerdings ist je nach Anwendungsumgebung der Einsatz dieser Möglichkeiten gegen die Performance und die Problemadäquatheit abzuwägen. Für die Bedürfnisse der Anwendungsumgebung scheinen die Möglichkeiten von XACML überproportioniert. | |
| 7.7 | Anbindung an LDAP Server für Rechte und Rollen | -o |
| | Eine Anbindung an einen LDAP Server wird von Muradora bereitgestellt. Informationen über Benutzer oder Rollen werden aber nicht entnommen und auch nicht für die Verwaltung von Rechten und Rollen verwendet. | |
| 7.8 | Administration von Rechten und Rollen | o |
| | Muradora bietet für die Verwaltung von Benutzern, Rechten und Rollen eine Funktion namens Global Access Control an. Diese Funktion bearbeitet XACML Policies und ermöglicht damit die feingranulare Definition von Rechten. Das Frontend dazu ist allerdings nicht für umfangreichere Anzahlen von Benutzern oder Rollen ausgelegt, so dass nur kleinere Zahlen effektiv bearbeitet werden können. Des Weiteren werden Informationen über Benutzer und Rollen über mehrere Dateien in verschiedenen Verzeichnisstrukturen abgelegt, was die Wartung für Administratoren erschwert. | |
| 7.9 | Eingabe und Bearbeitung von Metadaten | -o |
| | Muradora stellt für die Eingabe von Metadaten einen abstrakten Metadata Editor bereit, der je nach Anwendungsumgebung um verschiedene Metadatenformate, beispielsweise DC und Darwin Core, erweitert werden kann. Dem Benutzer wird dabei die Auswahl des Formats der Metadaten überlassen, was einen normalen Benutzer an dieser Stelle überfordern könnte. | |

| | | |
|---|---|---|
| | Muradora stellt einen, dort so bezeichneten, „Workflow“ bereit, der allerdings fest mit der Anwendung verdrahtet ist. Dieser Workflow unterscheidet dabei nicht zwischen Objekttypen oder Rollen. Damit wird nur ein Anwendungsfall, die Eingabe von Metadaten, unterstützt.<br><br>Für die Eingabe und Bearbeitung der Metadaten sieht Muradora XForms vor. XForms sollen es ermöglichen aus XML Schemata Formulare zu generieren und diese zu validieren, damit die Daten schließlich als digitales Objekt gespeichert werden können. Allerdings funktioniert die Validierung innerhalb von Muradora aus ungeklärten Gründen nicht und Formulare werden teilweise nicht korrekt präsentiert. | |

**Tab. 6.7: Anforderungen an Muradora**

Um die Anwendbarkeit des Prototyps beziehungsweise der einzelnen Systeme auf Muradora bestätigen zu können, wurde eine entsprechende Integration vorgenommen. Der Integration der Systeme folgte ein Integrationstest, der positiv bewertet werden konnte und erlaubt damit die Aussage, dass die Systeme des Prototyps auf Muradora anwendbar sind.

Die in dieser Arbeit entwickelten Systeme für Rollen, digitale Objekte und Workflows lassen sich auf Muradora anwenden. Allerdings wird damit eine Erweiterung von Muradora für Rollen, welche der Prototyp verwendet, um den Zugriff auf Objekttypen und Workflows einzuschränken, notwendig. Diese Erweiterung kann bei der Anmeldung an Muradora vorgenommen werden. Im Weiteren sind für die Benutzerinteraktion mit Funktionen für Workflows entsprechende Seiten, wie beispielsweise Submit und Publish, in das Frontend von Muradora einzubetten (vgl. Abschnitt 5.3.2).

Insgesamt bleibt festzuhalten, dass Muradora einen interessanten Ansatz für die Flexibilität von Rechten für digitale Informationsobjekte eines Fedora Repository bietet. Das Frontend von Muradora bietet einige Funktionen, wie beispielsweise ein Warenkorb für digitale Objekte und die Möglichkeit Literatureinträge in BibTeX[62] oder Endnote[63] zu exportieren. Leider ist aber auch die grafische Benutzungsschnittstelle nicht ausgereift und stellt sowohl Benutzer als auch (Weiter-) Entwickler von Muradora vor einige nicht trivial nachvollziehbare Probleme.

Schließlich ist für die Ergebnisse dieser Arbeit festzuhalten, dass Workflows in Muradora verwendet werden können, um digitale Informationsobjekte zu pflegen.

[62] http://www.bibtex.org/ (2009-03-18)
[63] http://www.endnote.com/ (2009-03-18)

Allerdings sollte eine Verwendung von Muradora in der betrachteten Version gründlich abgewogen werden. So ist angesichts der genannten Probleme Muradora besser nur als Prototyp zu betrachten, von dem für weitere Entwicklungen gelernt werden kann.

## 6.3 Beantwortung der Fragestellungen

In Abschnitt 1.2, der Einleitung, dieser Arbeit wurden einige Fragestellungen angesprochen, die hier untersucht werden sollten. Die vorhergehenden Kapitel haben diese Fragestellungen mit der Entwicklung von Architektur und Prototyp und der Analyse von Muradora implizit beantwortet.

Im Folgenden werden die Fragestellungen noch einmal aufgegriffen und ihre Ergebnisse zusammengefasst dargestellt.

1. Wie können Workflows für digitale Informationsobjekte definiert werden?

   *Digitale Informationsobjekte können beliebige Daten kapseln und mit Metadaten beschrieben werden. Workflows definieren im Normalfall einen abstrakten Arbeitsablauf. Wie kann nun ein Workflow Eigenschaften eines digitalen Objektes beachten?*

Workflows können für digitale Objekte definiert werden. Dazu wurde hier ein XML Schema für die Definition von Workflows entwickelt. Der Aufbau eines Workflows ermöglicht die Verarbeitung von Aktivitäten und verbindet Aktivitäten durch Transitionen. Aktivitäten enthalten dabei Funktionen zur Bearbeitung von digitalen Informationsobjekten und deren Metadaten. Transitionen können Bedingungen enthalten, welche mit einem Status eines digitalen Objektes verbunden sein können. Bedingungen können damit den Ablauf eines Workflows einschränken beziehungsweise steuern (vgl. Abschnitt 3.4).

2. Welchen Nutzen würden generische beziehungsweise dynamische Workflows für digitale Informationsobjekte erbringen?

   *Bei der Einführung von Workflows müssen bestimmte Anforderungen gegeben sein, welche den Aufwand für die Einführung und Wartung von Workflows rechtfertigen.*

Anforderungen für die Einführung von Workflows können beispielsweise in der Regelmäßigkeit oder Struktur eines Geschäftsprozesses gesehen werden (vgl. Kapitel 3). Die Einführung von Workflows kann dann solche Geschäftsprozesse unter-

stützen. Die Formulierung von dynamischen Workflows also Workflows welche die Daten von digitalen Objekten beachten, ermöglicht damit die Definition von Bedingungen für den Ablauf eines Workflows. So kann eine Verlagerung eines Teils der Logik in die Workflow-Definition erfolgen und einzelne Aktivitäten können auf eine entsprechende Logik verzichten.

Damit kann eine Wiederverwendbarkeit und bessere Wartbarkeit von Aktivitäten erreicht werden, die dann bei entsprechender Umsetzung, als atomare Einheiten betrachtet werden können. Diese Arbeit zeigt ein Szenario, in dem ein Benutzer zunächst ein Dokument auf einen Server lädt und dann Metadaten zu diesem Dokument angibt. Die Umsetzung als atomare Einheiten ermöglicht schon in diesem Szenario das einfache Austauschen der Aktivitäten Hochladen und Eingabe von Metadaten, indem die Workflow-Definition entsprechend umgestellt werden kann.

3. Welche Informationen eines digitalen Informationsobjektes sind für den Ablauf eines Workflows interessant?

   *Ein digitales Informationsobjekt kann über Metadaten verfügen, welche für den Ablauf eines Workflows von Interesse sein können. Zum Beispiel kann ein digitales Informationsobjekt über einen Status verfügen, welcher im Workflow beachtet werden soll. Das Szenario in Abschnitt 3.5 verfügt über einen solchen Fall.*

Prinzipiell können alle Informationen eines digitalen Informationsobjektes für den Ablauf eines Workflows von Interesse sein. Damit entsprechende Informationen in Workflows verwendet werden können, ermöglicht der entwickelte Prototyp, über die Formulierung von Bedingungen, den Zugriff auf beliebige Teile eines digitalen Objektes.

Damit können beliebige Informationen eines digitalen Objekts zu einem Status aggregiert werden, der wiederum von Workflows verwendet werden kann. Die Auswahl, welche Informationen einen Status bilden sollen, ist anwendungsspezifisch und kann nicht allgemein genannt werden.

Das in dieser Arbeit verwendete Szenario zeigt eine Bedingung, die an den recordStatus eines digitalen Objektes gebunden ist. Dabei können Benutzer digitale Objekte für Veröffentlichungen erstellen, welche über verschiedene Zustände (create, request, review und published) verfügen können (vergleiche dazu Abschnitt 3.5).

4. Auf welche Weise können Informationen über Benutzerrechte in Workflows eingebunden werden?

   *Workflows können an bestimmte Rechte gebunden sein. So kann es angebracht sein, dass ein Workflow oder auch einzelne Schritte eines Workflows nur von bestimmten Benutzern ausgeführt werden dürfen.*

Systeme, welche mit Informationsobjekten arbeiten, bieten für Benutzer dieses Systems eine Schnittstelle zur Interaktion an. In dieser Arbeit wurden grafische Benutzungsschnittstellen in Form von Web-Anwendungen betrachtet, zu denen das Muradora Framework und der Prototyp gehören.

So werden in der praktischen Anwendung nicht alle Benutzer eines Systems auf alle Funktionen oder alle Objekte, die mit einem System verwaltet werden, Zugriff haben dürfen. Durch Rollen und Rechte kann hier der Zugriff auf das Frontend und digitale Informationsobjekte eingeschränkt werden.

Eine Einschränkung für die Arbeit mit Workflows ist für diese Arbeit damit auch relevant. So dürfen Teile eines Workflows beispielsweise nur von bestimmten Benutzergruppen bearbeitet werden. Die entwickelte Workflow-Definition erlaubt die Angabe von Rollen für Aktivitäten und Transitionen eines Workflows. Durch eine solche Einschränkung werden unbefugte Benutzer von der Ausführung von Aktivitäten oder Transitionen abgehalten. Damit müssen Aktivitäten selbst sich nicht mehr um die Befugnis eines Benutzers zur Ausführung der Aktivität kümmern.

5. Wie können Beziehungen von digitalen Informationsobjekten untereinander dargestellt, aufgelöst und für Workflows benutzt werden?

   *Besitzen digitale Informationsobjekte Beziehungen zu anderen Objekten, so müssen auch diese Beziehungen gepflegt werden können.*

Mit Workflows werden hier digitale Informationsobjekte bearbeitet oder erstellt. Dabei können solche Objekte Beziehungen zu anderen Objekten besitzen, die wiederum entweder auch als digitales Informationsobjekt oder als externe Ressource vorliegen können. Im Wesentlichen sind zwei Probleme zu unterscheiden: die Darstellung und die Auflösung von Beziehungen.

Für die Darstellung von Beziehungen lässt sich eine Reihe von Möglichkeiten identifizieren. Dazu können beispielsweise Beziehungen mit einfachen Formularen oder mit interaktiven Formularen, wie mit Technologien des Web 2.0, bearbeitet werden.

Beziehung können dazu zum Beispiel auch als Tag Clouds[64] oder geografisch, in digitalen Karten, dargestellt werden. So zeigt das Szenario mit dem Prototypen die Verwaltung von digitalen Informationsobjekten für Veröffentlichungen, die am AWI entstehen. Dabei werden durch die Metadaten von digitalen Objekten hier unter anderem auch geografische Orte (Coverage) erfasst.

Beziehungen werden innerhalb des Fedora Frameworks mit Hilfe eines Datastreams beschrieben. Dieser Datastream ermöglicht die Angabe von beliebigen Beziehungen zwischen digitalen Informationsobjekten in der Sprache RDF. Es existieren eine Reihe von Inferenzmaschinen, welche auf RDF angesetzt werden können um entsprechende Beziehungen auflösen zu können. Damit wird also eine Verbindung zwischen konkreten digitalen Informationsobjekten ermöglicht (vgl. Abschnitt 2.5).

Diese Arbeit zeigt weiterhin die Verwendung von Objekttypen, die als Prototyp für digitale Informationsobjekte betrachtet werden können. Objekttypen können dabei in OWL beschrieben werden und damit durch ausgewählte Eigenschaften auch Beziehungen zu anderen Objekttypen besitzen. So zeigt das Szenario beispielsweise einen Zusammenhang zwischen den Objekttypen „inBook“ und „Book“, wobei in diesem Zusammenhang eine inverse Beziehung zu sehen ist. Auch für OWL stehen Werkzeuge zur Inferenz zur Verfügung, so dass Beziehungen zwischen Objekttypen als abstrakte Beziehung betrachtet werden können (vgl. Abschnitt 3.7).

Innerhalb des vorgestellten Workflow Szenarios besitzt eine Veröffentlichung in der Regel Referenzen zu anderen Veröffentlichungen, welche mit Metadaten beschrieben werden können. Somit können mit Hilfe von RDF und Objekttypen Aussagen über Beziehungen getroffen werden, die innerhalb eines Workflows zur Pflege digitaler Informationsobjekte bearbeitet werden können.

Ein weiterführendes Szenario wäre die Verwendung des Wissens über Beziehungen, um einen Workflow entsprechend dieser Beziehungen zu erweitern. So kann eine Aktivität eines ausgehenden Workflows die Beziehungen eines digitalen Objektes verwenden, um einen neuen Workflow für ein solches Objekt anzustoßen.

[64] Tag Clouds erlauben die grafische Anordnung von Begriffen (Tags) in einer Weise, welche die Zusammenhänge und Beziehungen zwischen solchen Begriffen illustrieren kann.

## 6.4 Abschließende Bewertung

Im Folgenden wird eine abschließende Bewertung gegeben. Dabei bezieht sich dieser Abschnitt auf die Ergebnisse und Probleme dieser Arbeit und der betrachteten Anwendungsumgebung. Abschnitt 7.2 liefert im Gegensatz dazu noch einen breiteren Ausblick.

In dieser Arbeit wurde die Strukturierung von digitalen Objekten in Collections und Member vorgestellt. Diese einfache Struktur liegt in der Aufbauorganisation der Anwendungsumgebung begründet, bei der zunächst alle digitalen Objekte in eine Collection eingeordnet werden sollen. Das Potenzial der Beziehungen, die bereits von Fedora unterstützt werden, wird damit nicht ausgenutzt. Auch der entwickelte Ansatz für Objekttypen findet für die Einordnung von digitalen Objekten in Collections daher an dieser Stelle keine Anwendung. Es kann aber davon ausgegangen werden, dass durch die weitere Bereitstellung von Collections eine Strukturierung von digitalen Informationsobjekten vorgenommen werden kann, welche sich positiv auf die Benutzbarkeit von Frontends auswirken würde.

Abschnitt 6.2 stellte die Anforderungen an den entwickelten Prototypen und Muradora vor und zeigte eine entsprechende Bewertung der Ergebnisse dieser Anforderungen. Es zeigte sich, dass die Anforderungen für den Prototypen weitgehend erfüllt werden konnten und damit eine Pflege von digitalen Informationsobjekten mit Hilfe von Workflows möglich wird. Die Teilsysteme des Prototyps lassen sich auf Muradora anwenden und können generell für andere Frontends weiter verwendet werden. Dabei sind die Teilsysteme auf die Grundprinzipien der service-orientierten Architektur ausgelegt, so dass bei Bedarf eine Trennung der Systeme beispielsweise durch Web Services erfolgen kann.

Durch die Bewertung von Muradora wurde allerdings auch deutlich, dass ein erheblicher Aufwand zur Einarbeitung in Muradora erforderlich ist. Dabei sollte Muradora nur als Prototyp betrachtet werden, beziehungsweise für den Einsatz von Muradora eine nicht zu geringe Integrations- und Testphase eingeplant werden, wenn Muradora produktiv genutzt werden soll.

Für den praktischen Einsatz der entwickelten Systeme sind auch hier in Bezug auf eine Anwendungsumgebung weitere Tests zu empfehlen. Diese Arbeit entwickelte einen Prototypen, dessen Funktionsfähigkeit durch Tests und Szenarien bestätigt wurde. Dennoch gilt es, mögliche Fehlerquellen beispielsweise durch einen Testbetrieb zu identifizieren und daraus weitere Erkenntnisse abzuleiten.

Im Weiteren wurde angesprochen, dass die Sprache zur Definition von Workflows einige Einschränkungen im Vergleich zu anderen Sprachen besitzt. Auch dieser Punkt könnte durch Testbetrieb weitere Anforderungen erbringen. Eine mögliche Vereinfachung liegt etwa in der Bereitstellung eines Werkzeuges, welches aus ereignisgesteuerten Prozessketten oder einem grafischen Modell eine entsprechende Workflow-Definition ableiten kann.

Mit Blick auf das Workflowmanagement zeigt der Prototyp zum einen die Möglichkeit Workflows zu definieren, zu instanzieren und zu beobachten. Das entwickelte Workflow-System stellt dazu Funktionen bereit um Workflows auszuführen und Informationen über den aktuellen Zustand eines Workflows zu erhalten. Mit diesen Funktionen kann das Workflow-System auch im Workflowmanagement genutzt werden um Workflows zu kontrollieren, zu steuern und Entscheidungen zur weiteren Ausführung von Workflows mit einzubeziehen.

Um die Ergebnisse eines Workflows und damit eines Geschäftsprozesses auch für das Workflowmanagement verwenden zu können, sollte eine Erweiterung des Prototyps um eine Persistenz von Workflow-Instanzen ermöglicht werden. Persistenz könnte dabei den aktuellen Zustand von Workflow-Instanzen, beispielsweise in einer Datenbank, speichern und damit auch den Zugriff über verschiedene Systeme auf die Workflow Informationen erlauben. Eine Möglichkeit dazu bietet das Objekt-relationale Mapping (ORM) an, wie es unter anderem vom Hibernate[65] Framework angeboten wird. Als weitere Einsatzmöglichkeit könnte damit auch die Versionierung und Archivierung von Workflows erlaubt werden.

Im Gegensatz zum Prototypen, in dem Workflow-Definitionen als XML Dokumente in Form von Dateien gespeichert werden, könnten diese auch in einem Fedora Repository abgelegt werden. Die Flexibilität des Datenmodells von Fedora wurde in dieser Arbeit angesprochen und deutet an, dass selbstverständlich auch Workflows gespeichert und Persistenz für Workflow-Instanzen damit ermöglicht werden könnte. Dieser Ansatz wurde in der Arbeit nicht weiter verfolgt, da zunächst nur digitale Objekte für Veröffentlichungen im Fedora Repository abgelegt werden sollten, was von der Anwendungsumgebung vorgegeben wurde.

Mit einem weiteren Blick auf das in dieser Arbeit behandelte Szenario und den damit verbundenen Workflow beziehungsweise Geschäftsprozess sollten die Ergebnisse dieser Arbeit auch bei zukünftigen Entwicklungen betrachtet werden. So werden in den nächsten Jahren in der Anwendungsumgebung weitere Anforderungen in Hinblick auf

[65] http://www.hibernate.org/ (2009-03-18)

Workflows hervorgebracht werden, die aus Erkenntnissen aktueller Projekte stammen. Eine wesentliche Aufgabe wird dann sein, vorhandene Geschäftsprozesse zu identifizieren, zu modellieren und in geeignete Workflows umzusetzen, um auch zukünftig mit digitalen Objekten arbeiten zu können. Dabei schließt sich auch die Frage nach neuen Geschäftsprozessen und der Flexibilität von existierenden Geschäftsprozessen in der Anwendungsumgebung an, um für sich ändernde beziehungsweise sich entwickelnde Anforderungen auch weiterhin Sorge tragen zu können.

Für die Indexierung von Metadaten wurde in der Entwicklung der Solr Search Server verwendet. Standardmäßig werden mit Blick auf Muradora hier im Wesentlich nur DC Metadaten und Fedora Metadaten für digitale Objekte indexiert. Um weitere Vorteile für die Performance gewinnen zu können, sollte eine Indexierung von Metadaten je Anwendung in Erwägung gezogen werden. Im vorgestellten Szenario können dabei beispielsweise Felder, wie Objekttyp, Organisationseinheit oder auch assoziierte Dokumente, indexiert werden. Bei der Auswahl von Feldern, die für eine Indexierung verwendet werden sollen, ist dabei aber auch eine Kosten-Nutzen-Rechnung erforderlich. So stellt sich bei einer Indexierung vieler Metadaten beispielsweise die Frage, ob digitale Objekte nicht direkt in einer Datenbank abgelegt und indexiert werden sollten.

Ein Problem bei der Zusammenarbeit des Solr Search Servers mit Fedora beziehungsweise Muradora liegt in der Tatsache, dass beim Löschen von digitalen Objekten die Referenz auf ein digitales Objekt im Index, zumindest in einigen Fällen, erhalten bleibt. In den betrachteten Versionen ermöglicht nur eine Neu-Indexierung die Entfernung von nicht mehr vorhandenen digitalen Objekten, welche eine erhebliche Systemauslastung – schon bei 15.000 digitalen Objekten – mit sich bringt. Es ist klar festzuhalten, dass dieses Problem für Anwendungsumgebungen wie dem AWI mit einem hohen Aufkommen an digitalen Objekten (möglicherweise Millionen digitale Objekte jährlich) nicht akzeptabel ist.

Im Folgenden werden nochmals einige Aspekte aufgegriffen, die über die an diese Arbeit gestellten Anforderungen hinausgehen. Die genannten weiteren Anforderungen sollten dabei unbedingt für Weiterentwicklungen betrachtet und weiter ausformuliert werden. Für die folgenden Aspekte wird je eine Prioritätenliste vorgeschlagen.

1. Analyse und Beschreibung bestehender Prozesse und Erhebung von notwendigen beziehungsweise ergänzenden Prozessen

   Die Analyse von bestehenden beziehungsweise ergänzenden Prozessen stellt eine wichtige Aufgabe dar, die über den Rahmen dieser Arbeit hinausgeht. So sind nicht nur bestehende Prozesse isoliert zu identifizieren, sondern im Zusammenhang mit anderen Prozessen, insbesondere fachspezifisch und auch fachübergreifend, zu sehen. Die Anforderungen an solche Prozesse müssen dazu konkret je Prozess festgehalten werden und mit den Benutzern dieser Prozesse erörtert werden. Wesentlich dabei ist auch die Schaffung von Akzeptanz von Benutzern für modellierte Prozesse als Workflow. Es schließt sich weiterhin die Frage der Adäquatheit einer Workflow-Lösung an.

2. Modellierung von Workflows

   In Organisationen ist die Personengruppe zu betrachten, die Workflows modelliert. Workflows werden in dieser Arbeit in einer eigenen XML-basierten Sprache formuliert. Zwar sind Workflows damit, im Gegensatz zu komplexen Workflow-Lösungen auf dem Markt, recht einfach zu formulieren, doch würde eine Werkzeugunterstützung zur Modellierung von solchen Workflows Administratoren die Arbeit erleichtern. Es muss auch festgehalten werden, dass in Organisationen der Umstand besteht, Modellierungen durch informatikfremde Personen vorzunehmen – im Gegensatz zu Unternehmen, die sich professionelle Modellierer leisten können oder müssen.

   Wenn ein Bedarf aus denen in Punkt 1 analysierten Anforderungen und Prozessen besteht, die deutlich über die in dieser Arbeit entwickelte Workflow-Lösung hinausgehen, sollten existierende Workflow-Lösungen untersucht werden. Es ist dabei abzuwägen ob größere Workflow-Lösungen im Konflikt zu den in Punkt 2 genannten Umständen stehen.

3. Persistenz und Sicherheit von Workflows

   Einige Möglichkeiten von Persistenz wurden bereits zuvor in diesem Abschnitt vorgestellt. Dabei ermöglicht Persistenz auch die Abdeckung eines gewissen Sicherheitsbedürfnisses. So stehen geeignet persistent verwaltete Workflows beispielsweise auch nach einem Absturz des Workflow-Systems weiter zur Verfügung. Weitere Anforderungen sollten in Bezug auf Ausnahmebehandlungen und Abbruchbedingungen von Workflows ermittelt werden. Zum

Beispiel sollte der Abbruch einer Aktivität einen Workflow sinnvoll beenden und damit verbundene digitale Objekte zweckmäßig verwalten.

Die Sicherheit betreffende Anforderungen sollten klassisch in Bezug auf Web-Anwendungen und weiter mit Blick auf die Handhabung von Workflows betrachtet werden. Der Begriff Sicherheit kann dabei neben der Frage von personellen Zuständigkeiten für Workflows auch breiter gefasst werden. Ein breiteres Verständnis kann dabei beispielsweise auch eine Protokollierung von Interaktionen mit Workflows und Auswirkungen von Aktivitäten eines Workflows beinhalten, um Probleme aufzudecken und gegebenenfalls zu beheben.

Als zusammenfassendes Ergebnis der Evaluierung können die folgenden Punkte als Empfehlung festgehalten werden.

- Fedora bietet ein flexibles Datenmodell für heterogene digitale Objekte und deren Beziehungen. Zur Strukturierung von digitalen Objekten sollten Objekttypen identifiziert und definiert werden. Diese sollten gemeinsam, das heißt insbesondere fachübergreifend und organisationsübergreifend, verwendet werden können.

- In einer Organisation implizit oder explizit existierende Prozesse beziehungsweise Workflows müssen identifiziert, analysiert und dokumentiert werden. Gemeinsamkeiten von Workflows sollten dabei besonders betrachtet werden um beispielsweise eine Wiederverwendbarkeit von Workflows oder Teilen davon zu erreichen.

- Für die in dieser Arbeit entwickelte Workflow-Lösung sind durch Testbetrieb und Einbindung in konkrete Projekte weitere Anforderungen zu erheben und diese gegebenenfalls geeignet zu implementieren. Wenn notwendig, sollten auch andere Workflow-Lösungen betrachtet werden.

- Die Akzeptanz der Benutzer für ein System zur Pflege digitaler Informationsobjekte mit Workflows muss hergestellt und gefördert werden. Dazu gehören unter anderem die Einbeziehung von Benutzergruppen bei der Entwicklung, die Verbreitung von Informationen und die Schulung im Umgang mit solchen Systemen.

- Muradora als Web-Anwendung sollte in der betrachteten Version nicht ohne sorgfältige Prüfung der benötigten Funktionen produktiv genutzt werden. Der Umstieg auf die neue Muradora-Version, welche auf Fedora 3 ausgelegt ist, sollte frühzeitig vorgenommen werden, um ausführliche Tests und Anpassungen zu erlauben.

- Die Verwendung und Notwendigkeit feingranularer Rechte für digitale Objekte ist zu prüfen. Hier ist festzuhalten, welche Anforderungen an solche Rechte existieren und inwieweit diese mit der in Muradora vorgestellten XACML-Lösung problemadäquat und performant gelöst werden können.

- Die Anforderungen an eine Indexierung von digitalen Objekten, Metadaten und Volltexten sollten für verwendete heterogene Formate gesammelt werden. Entsprechend sind Felder zu identifizieren, die indexiert werden sollen.

Das nächste Kapitel schließt diese Arbeit mit einer Zusammenfassung und einem Ausblick ab. Dabei werden aktuelle Themen und weitere mögliche Fragen angeschnitten, die über diese Arbeit hinausgehen.

# 7 Zusammenfassung und Ausblick

Das Alfred-Wegener-Institut für Polar- und Meeresforschung (AWI) sammelt durch seine Forschung laufend große Mengen heterogener Daten und Informationen über verschiedene wissenschaftliche Bereiche. Diese Daten und Information müssen geeignet gespeichert und verwaltet werden. Dazu führt das AWI in einem aktuellen Projekt das Fedora Framework als Repository und Muradora als Frontend für digitale Informationsobjekte ein. Fedora ermöglicht die Verwaltung digitaler Informationsobjekte, die mit Metadaten semantisch beschrieben werden können.

Als Großforschungseinrichtung entstehen am AWI Ergebnisse, die in Dokumenten beschrieben und dann veröffentlicht werden sollen. Die Veröffentlichung von Dokumenten unterliegt einem bestimmten Prozess, der mit Hilfe von Workflows abgebildet werden kann.

Dabei stellt sich die Frage, wie Workflows zur Pflege digitaler Informationsobjekte eingesetzt werden können und wie Workflows in Muradora verwendet werden können.

In der vorliegenden Arbeit werden Ansätze und ein Prototyp entwickelt, die demonstrieren, wie Workflows in Anwendungen für digitale Informationsobjekte, genutzt werden können. Dazu werden grundsätzlich die Prinzipien einer service-orientierten Architektur betrachtet. Es werden entsprechende Anforderungen an den Prototypen und Muradora formuliert und im Anschluss an die Entwicklung evaluiert. Im Laufe der Arbeit wird ein Szenario mit Bezug zum AWI vorgestellt und im Prototypen angewandt.

## 7.1 Ergebnisse

Aus dieser Arbeit gehen mehrere Ergebnisse hervor:

- eine Architektur, welche die Bereitstellung von Objekttypen und Workflows in Bezug auf Repositories ermöglicht und Rechte und Rollen diesbezüglich beachtet,
- ein Entwurf und ein Prototyp, welche die Architektur und die Anforderungen an ein System zur Pflege digitaler Informationsobjekte mit Hilfe von Workflows umsetzen,
- eine Integration der Architektur und des Prototyps in Muradora und
- die Evaluation der Ergebnisse mit den gestellten Anforderungen.

Die Evaluierung des Prototypen zeigt im Wesentlichen die Erfüllung der gestellten Anforderungen. Des Weiteren zeigt die Evaluierung von Muradora interessante Ansatzpunkte für Frontends von Repositories für digitale Informationsobjekte und flexible Rechte über XACML Policies auf. Allerdings stellen die undokumentierte Struktur von Muradora und Fehler des sich in der Entwicklung befindlichen Frameworks verschiedene Probleme dar. So kann hier empfohlen werden, Muradora nur als Prototypen zu betrachten und von einem produktiven Einsatz in der betrachteten Version abzusehen.

Als Ergebnis dieser Arbeit kann insgesamt festgehalten werden, dass Workflows für die Pflege digitaler Informationsobjekte verwendet werden können und die hier entwickelten Systeme für Workflows, Objekttypen und Rechte und Rollen in das Muradora Framework integrierbar sind. Eine Vertiefung der Evaluierung und der Ergebnisse dieser Arbeit ist in Kapitel 6 zu finden.

Die Ergebnisse dieser Arbeit können dabei auch über das Muradora Framework hinaus auf andere Systeme und Anwendungen übertragen werden. So können die vorgestellten Entwicklungen prinzipiell auch für andere Systeme und Frontends verwendet werden, um Workflows und Objekttypen in einer Anwendung benutzen zu können.

## 7.2 Ausblick

Die Verwendung von Repositories zur Verwaltung und Archivierung von digitalen Informationen und Inhalten ist ein aktuelles Thema für verschiedene Organisationen. So werden Repositories wie Fedora stetig weiter entwickelt. Auch Alternativen wie beispielsweise DSpace[66] und Erweiterungen von Repositories, insbesondere auch um Workflows, sind in Arbeit. So schreibt beispielsweise die Deutsche Forschungsgemeinschaft (DFG) aktuell den „Aufbau und Vernetzung von Repositorien“ im Jahr 2009 aus[67].

Das neue Fedora 3 bietet im Gegensatz zu der in dieser Arbeit betrachteten Version 2 eine Content Model Architecture (CMA). Diese erlaubt die Definition einer Art von Objekttypen für digitale Objekte. Damit stellt sich hier beispielsweise die Frage, welche Möglichkeiten die CMA bietet und wie die hier entwickelten Ansätze für Objekttypen und Workflows auch auf die CMA anwendbar sind. Aktuelle Diskussionen beziehen sich auf die Möglichkeiten von OWL für Content Models in der CMA (vgl. Blekinge-

[66] http://www.dspace.org/ (2009-03-18)
[67] http://www.dfg.de/forschungsfoerderung/wissenschaftliche_infrastruktur/lis/aktuelles/download/ausschreibung_massnahme_repositorien_081212.pdf (2009-03-18)

Rasmussen 2009). Es schließt sich auch weiterhin die Frage zur Formulierung und Darstellung von Beziehungen zwischen digitalen Objekten beziehungsweise Content Models oder Objekttypen an.

Ein weiterer offener Punkt liegt in der Semantik von Objekttypen über Organisationen hinweg. Aktuelle Themen sind die Interoperabilität und Integration von mehreren Repositories, um Benutzern beispielsweise den Zugriff über ein Frontend auf mehrere Repositories zu ermöglichen. Um den Austausch von digitalen Informationsobjekten zwischen verschiedenen Repositories zu erlauben, müsste eine gemeinsame Semantik beziehungsweise eine ähnliche Bedeutung für Objekttypen und ihren Inhalt gefunden werden. Das Finden solcher Gemeinsamkeiten könnte durch die Einführung von de facto Standards für Objekttypen (Content Models) und damit eventuell auch Ontologien in entsprechenden Communities erfolgen. Es stellt sich die Frage, ob dieses eine adäquate Möglichkeit für unterschiedliche Anwendungsumgebungen darstellt und die Interoperabilität von Repositories damit unterstützt werden kann, oder ob es Alternativen dazu gibt.

Es existieren verschiedene Repository-Lösungen auf dem Markt. Sollen digitale Objekte und Objekttypen über Repositories hinweg verwendet und ausgetauscht werden, so stellen sich Fragen im Hinblick auf die Möglichkeiten zur Integration von Repositories. Insbesondere sind dabei Fragen zu gemeinsamen Schnittstellen und Fragen zur Vernetzung von digitalen Objekten von Interesse. Eine Möglichkeit zur Vernetzung von Repositories könnte mit iRODS[68] gegeben werden.

Neben dem Repository selbst sind auch die Benutzungsschnittstellen dieser Repositories von entscheidender Bedeutung. Schließlich benutzen Menschen diese Frontends um digitale Informationsobjekte zu erstellen oder zu bearbeiten. Damit sind auch weiterhin Workflows zur Abbildung von Prozessen von Interesse. Mit einem Blick auf Workflows könnten weitere Arbeiten in Richtung der Analyse bestehender Workflow-Systeme vorgenommen und ihre Einsatztauglichkeit für die Pflege digitaler Informationsobjekte untersucht werden. Bezogen auf die in dieser Arbeit entwickelte Sprache zur Definition von Workflows kann untersucht werden, ob weitere Anforderungen existieren, die in der vorliegenden Version noch nicht betrachtet worden sind.

Als praktischer Ausblick sollten die Ergebnisse dieser Arbeit auch auf andere Szenarien und Frontends angewendet werden. Für die Handhabung von Workflows und

[68] iRODS soll die Integration von verschiedenen Datenbanken und einen einheitlichen Zugriff auf Datenquellen erlauben. Siehe dazu auch https://www.irods.org (2009-03-18) und http://www.diceresearch.org/DICE_Site/Intro.html (2009-03-18).

Objekttypen könnten Werkzeuge entwickelt werden, welche die Bearbeitung von Workflow-Definition und Objekttypen unterstützen. In der Anwendungsumgebung des AWI werden in den kommenden Jahren Projekte mit Bezug zu Repositories und Workflows zur Pflege digitaler Informationsobjekte, insbesondere bei der Langzeitarchivierung[69] von Daten, von Interesse sein. So können die Ergebnisse im Speziellen dort und generell für ähnliche Anwendungen, in denen die Speicherung, Verwaltung und Arbeit mit Informationsobjekten von Bedeutung ist, wieder verwendet werden.

Weitergehende Fragen ergeben sich mit Blick auf den Umfang von Datenmengen für Repositories und den Einsatz von Workflows für große Datenmengen beziehungsweise Zahlen von digitalen Informationsobjekten. Dabei stellen sich Fragen zur Performance, Organisation und Verwaltung. Zu betrachtende Problemfelder sind dabei digitale Informationsobjekte, Workflows sowie Rechte und Rollen innerhalb einer Organisation oder über mehrere Organisationen hinweg.

Ebenfalls interessant ist, in welcher Weise neue Technologien, beispielsweise aus dem Bereich des Web 2.0, den Zugriff auf Repositories und damit auf digitale Objekte und deren Beziehungen für Benutzer unterstützen können. In diesem Zusammenhang stellen sich Fragen zur Gebrauchstauglichkeit und eine Bewertung von Frontends für Repositories und den Einsatz von Workflows für die Pflege digitaler Informationsobjekte. Für eine Optimierung von Frontends je Anwendungsgebiet und für bestimmte Benutzergruppen sollten entsprechende Untersuchungen in Hinblick auf Anforderungen und Erwartungen vorgenommen werden, die von Benutzern an Frontends beziehungsweise Repositories gestellt werden.

## 7.3 Erfahrungen

Die Arbeit an der Diplomarbeit erforderte zunächst eine gezielte Auseinandersetzung mit den für die Anwendungsumgebung relevanten Technologien. Zu diesen Technologien zählten insbesondere Fedora und natürlich Muradora. Die Einarbeitung in Fedora und vor allen Dingen auch Muradora erforderte einen erheblichen zeitlichen Aufwand.

Bereits die reine Installation der notwendigen Technologien gestaltete sich schwieriger als erwartet und kostete überproportional viel Zeit. Das AWI stellte für die Installation

[69] Bei der Langzeitarchivierung von Daten wird in dem wissenschaftlichen Kontext von Repositories von Zeiträumen von 10 Jahren bis zu 100 Jahren gesprochen (Peterson u. a. 2007).

der Technologien einen entsprechenden Server bereit. Neben den eigentlichen Fedora und Muradora mussten unter anderem auch verschiedene Datenbanksysteme für die Frameworks eingerichtet werden. Leider ergaben sich durch nicht dokumentierte technologische Abhängigkeiten zwischen den einzelnen Systemen beziehungsweise dem Betriebssystem Probleme die identifiziert und dann gelöst werden mussten.

Nachdem die technischen Voraussetzungen geschaffen wurden musste die Struktur und Funktionsweise von Muradora durch eigene Analysen und Tests bestimmt werden, da keine entsprechende Dokumentation für das Muradora Framework zur Verfügung stand und bis heute zur Verfügung steht. Die Komplexität von Muradora und der generelle Aufbau der Web-Anwendung machte auch diesen Schritt zu keinem trivialen Problem.

Im weiteren Verlauf der Diplomarbeit wurde eine Architektur aus den gesammelten Anforderungen extrahiert und in einen Entwurf mit Prototyp umgesetzt. Dabei kommt der Entscheidung, eine eigene Workflow-Engine zu entwickeln, eine wesentliche Bedeutung zu, die durch Expertenmeinungen, aus Kreisen von Fedora und Muradora und der Anwendungsumgebung, durchaus Unterstützung fand.

Schließlich ist es positiv zu sehen, dass die Ergebnisse dieser Arbeit in Bezug auf Fedora, Muradora und die Verwendung von Workflows und auch Objekttypen, sowie Rechten und Rollen, auch weiterhin in der Anwendungsumgebung Beachtung finden werden. Damit haben die Ergebnisse einen praktischen Nutzen, der in kommenden Projekten wieder zu finden sein wird.

Neben den Erfahrungen, die durch die Einarbeitung in die neuen Technologien gesammelt werden konnten, ist auch der enge Kontakt zu den Mitarbeitern des AWI und das Kennenlernen des Facettenreichtums der Anwendungsumgebung am AWI, der Einblick in künftige Anforderungen und Herausforderungen eine durchweg positive Erfahrung gewesen.

# 8 Literatur

van der Aalst, W. M. P. u. a. (2005): Case Handling: A New Paradigm for Business Process Support. *Data & Knowledge Engineering, 53*, S. 129-162.

Agostini A. und De Michelis, G. (2000): Improving Flexibility of workflow Management Systems. In: *Business Process Management*. Springer, Berlin. S. 289-342.

Allemang, D. und Hendler J. A. (2008): *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann, Amsterdam.

Berners-Lee, T. (1994): RFC 1630 – Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web. http://www.rfc-editor.org/rfc/rfc1630.txt [2009-03-20]

Berners-Lee, T. u. a. (2005): RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax. http://www.rfc-editor.org/rfc/rfc3986.txt [2009-03-20]

Blekinge-Rasmussen, A. (2009): Fedora Enhanced Content Models. Letzte Änderung: 2009-02-16. http://fedora-commons.org/confluence/display/DEV/Fedora+Enhanced+Content+Models [2009-03-20]

Blekinge-Rasmussen, A. und Fiedler Christiansen, K. (2008): Digital Repositories and Data Models. In: *2nd European Workshop on the Use of Digital Object Repository Systems in Digital Libraries (DORSDL2).* Aarhus. Verfügbar unter: http://dorsdl2.cvt.dk/dorsdl2-6-blekinge.pdf [2009-03-20]

Booth, D. u. a. (2004): Web Services Architecture. W3C Working Group Note. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ [2009-03-20]

Box, D. u. a. (2000): Simple Object Access Protocol (SOAP) 1.1. W3C Note. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ [2009-03-20]

Boyer, J. M. (2007): XForms 1.1. W3c Candidate Recommendation. http://www.w3.org/TR/2007/CR-xforms11-20071129/ [2009-03-20]

Chinnici, R. u. a. (2007): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation. http://www.w3.org/TR/2007/REC-wsdl20-20070626/ [2009-03-20]

Clement, L. u. a. (2004): UDDI Version 3.0.2. UDDI Spec Technical Commitee Draft. http://uddi.org/pubs/uddi-v3.0.2-20041019.htm [2009-03-20]

Cover, R. (2007): Muradora GUI for Fedora Repository Uses SAML and XACML for Federated Identity. http://xml.coverpages.org/ni2007-10-26-a.html [2009-03-20]

Dean, M. und Schreiber, G. (2004): OWL Web Ontology Language Reference. W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-ref-20040210/ [2009-03-20]

Dumke, R. (2003): *Software Engineering: Eine Einführung für Informatiker und Ingenieure - Systeme, Erfahrungen, Methoden, Tools* (4. überarbeitete und erweiterte Auflage). vieweg, Braunschweig.

Eicker, S. (2008): Repository-System (Repository). In: Kurbel, K. et al. (Hrsg.): *Enzyklopädie der Wirtschaftsinformatik* (Zweite Auflage). Oldenbourg, München. Letzte Änderung: 2008-09-26. http://www.enzyklopaedie-der-wirtschaftsinformatik.de [2009-03-20]

Erl, T. (2005): *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall International, Upper Saddle River.

Faulkner, X. (2000): *Usability Engineering*. Palgrave Macmillan, Houndmills.

Fedora Commons (2008a): About Fedora Commons. http://www.fedora.info/about/ [2009-03-20]

Fedora Commons (2008b): Digital Object Relationships. Letzte Änderung: Davis, D. 2008-06-30. http://www.fedora-commons.org/confluence/display/FR22DOC/Digital+Object+Relationships [2009-03-20]

Fedora Commons (2008c): Fedora Digital Object Model. Letzte Änderung: Davis, D. 2008-06-30. http://www.fedora-commons.org/confluence/display/FR22DOC/Fedora+Digital+Object+Model [2009-03-20]

Fedora Commons (2008d): Introduction to FOXML. Letzte Änderung: Davis, D. 2008-06-30. http://www.fedora-commons.org/confluence/display/FR22DOC/Introduction+to+FOXML [2009-03-20]

Fedora Commons (2008e): Web Service Interfaces. Letzte Änderung: Davis, D. 2008-07-16. http://www.fedora-commons.org/confluence/display/FR22DOC/Web+Service+Interfaces [2009-03-20]

Fedora Development Team (2008): Fedora Tutorial #1 - Introduction to Fedora. https://fedora-commons.org/confluence/download/attachments/4718930/tutorial1.pdf?version=1 [2009-03-20]

Fensel, D. u. a. (2006): *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, Berlin.

Fielding, R. T. (2000): Architectural Styles and the Design of Network-based Software Architectures. Dissertation. University of California, Irvine. Verfügbar unter: http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf [2009-03-20]

Gamma, E. u. a. (1995): *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam.

Gruber, T. R. (1993): A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, Volume 5*, Issue 2, S. 199-220. Verfügbar unter: http://tomgruber.org/writing/ontolingua-kaj-1993.pdf [2009-03-20]

Heinrich, L. J. (2002): *Informationsmanagement: Planung, Überwachung und Steuerung der Informationsinfrastruktur*. Oldenbourg, München.

Higgins, S. u. a. (2007): The Draft DCC Curation Lifecycle Model. In: *3rd International Digital Curation Conference "Curating our Digital Scientific Heritage: a Global Collaborative Challenge"*, Washington. Verfügbar unter: http://www.dcc.ac.uk/events/dcc-2007/posters/DCC_Curation_Lifecycle_Model.pdf [2009-03-19]

Hitzler, P. u. a. (2008): *Semantic Web: Grundlagen*. Springer, Berlin.

Jablonski, S. u. a. (1997): *Workflow-Management - Entwicklung von Anwendungen und Systemen - Facetten einer neuen Technologie*. dpunkt, Heidelberg.

Keller, G. u. a. (1992): Semantische Prozeßmodellierung auf der Grundlage von "Ereignisgesteuerter Prozeßketten (EPK)". In: Scheer, A.-W. (Hrsg.): *Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89*. Saarbrücken. Verfügbar unter: http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf [2009-03-20]

Koivunen, M. und Miller, E. (2002): W3C Semantic Web Activity. In: Hyvönen, E. (Hrsg.): *Sematic Web Kick-Off in Finnland - Vision, Technologies, Research, and Applications.* HIIT Publications, Helsinki. S. 27-43. Verfügbar unter: http://www.cs.helsinki.fi/u/eahyvone/stes/semanticweb/kick-off/proceedings.pdf [2009-03-20]

Krafzig, D. u. a. (2004): *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall International, Upper Saddle River.

Krallmann, H. und Trier, M. (2008): Workflow-Modellierung. In: Kurbel, K. et al. (Hrsg.): *Enzyklopädie der Wirtschaftsinformatik* (Zweite Auflage). Oldenbourg, München. Letzte Änderung: 2008-09-22. http://www.enzyklopaedie-der-wirtschaftsinformatik.de [2009-03-20]

Lackes, R. und Siepermann, M. (2008): Web 2.0. In: Kurbel, K. et al. (Hrsg.): *Enzyklopädie der Wirtschaftsinformatik* (Zweite Auflage). Oldenbourg, München. Letzte Änderung: 2008-09-14. http://www.enzyklopaedie-der-wirtschaftsinformatik.de [2009-03-20]

Lagoze, C. u. a. (2006): Fedora: an architecture for complex objects and their relationships. *International Journal on Digital Libraries, Volume 6,* Issue 2. S. 124-138.

Manola, F. und Miller, E. (2004): RDF Primer. W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ [2009-03-20]

Masak, D. (2007): *SOA? - Serviceorientierung in Business und Software*. Springer, Berlin.

Moats, R. (1997): RFC 2141 - URN Syntax. http://www.rfc-editor.org/rfc/rfc2141.txt [2009-03-20]

Moses, T. (2005): eXtensible Access Control Markup Language (XACML) Version 2.0 - OASIS Standard. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf [2009-03-20]

Müller, J. (2005): *Workflow-based Integration: Grundlagen, Technologien, Management*. Springer-Verlag, Berlin.

Muradora (2008): The Melcoe PDP. Letzte Änderung: Nishen, 2008-02-22. http://www.muradora.org/muradora/wiki/MelcoePDPDoc [2009-03-20]

Nguyen, C. (2008): Access Control with Muradora. Letzte Änderung: Nguyen, C. 2008-12-08. https://fedora-commons.org/confluence/display/MURADORA/Access+Control [2009-03-20]

Nguyen, C. u. a. (2007): Federated Authentication and Authorization for Fedora. http://www.ramp.org.au/drama/documents/extended-abstract.pdf [2009-03-20]

Nguyen, C. und Dalziel, J. (2008): Muradora: A Turnkey Fedora GUI Supporting Heterogeneous Metadata, Federated Identity, And Flexible Access Control. In: *OR2008, Third International Conference on Open Repositories 2008*, Southampton.
Verfügbar unter: http://pubs.or08.ecs.soton.ac.uk/111 [2009-03-20]

Nielsen, J. (1993): *Usability Engineering*. Morgan Kaufmann, San Francisco.

Nurcan, S. und Edme, M. (2005): Intention-driven modeling for flexible workflow applications. *Software Process: Improvement and Practice, Volume 10*, Issue 4. S. 363-377.

Peterson, M. u. a. (2007): 100 Year Archive Requirements Survey. Storage Networking Industry Association. http://www.snia.org/forums/dmf/programs/ltacsi/forums/dmf/programs/ltacsi/100_year/100YrATF_Archive-Requirements-Survey_20070619.pdf [2009-03-20]

Prud'hommeaux, E. und Seaborne, A. (2008): SPARQL Query Language for RDF. W3C Recommendation. http://www.w3.org/TR/rdf-sparql-query/ [2009-03-20]

Rechenberg, P. und Pomberger, G. (2006): *Informatik Handbuch*. Hanser Fachbuchverlag, München.

Reijers, H. u. a. (2003): The Case Handling Case. *International Journal of Cooperative Information Systems, Volume 12*, Issue 3. S. 365-391.

Russell, N. u. a. (2006): Workflow Control-Flow Patterns: A Revised View. In: *BPM Center Report BPM-06-22, BPMCenter.org*.
Verfügbar unter: http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf [2009-03-20]

Russell, N. u. a. (2004): Workflow Data Patterns. http://www.workflowpatterns.com/documentation/documents/data_patterns%20BETA%20TR.pdf [2009-03-20]

Russell, N. u. a. (2005): Workflow Data Patterns: Identification, Representation and Tool Support. In: *Conceptual Modeling – ER 2005*. Springer, Berlin. S. 353-368.

Treloar, A. und Harboe-Ree, C. (2008): Data management and the curation continuum: how the Monash experience is informing repository relationships. In: *Vala 2008 Conference, 14th Biennial Conference & Exhibition*, Melbourne.
Verfügbar unter: http://www.valaconf.org.au/vala2008/papers2008/111_Treloar_Final.pdf [2009-03-20]

W3C (2003): Extensible Markup Language (XML). http://www.w3.org/XML/ [2009-03-20]

Wenz, C. (2007): *JavaScript und AJAX - Das umfassende Handbuch*. Galileo Press, Bonn.

Wimmel, H. (2008): *Entscheidbarkeit bei Petri-Netzen: Überblick und Kompendium*. Springer, Berlin.

Wohed, P. u. a. (2008): Patterns-based Evaluation of Open Source BPM Systems: The Cases of jBPM, OpenWFE, and Enhydra Shark. http://eprints.qut.edu.au/archive/00014320/ [2009-03-20]

Workflow Management Coalition (2008): Workflow Management Coalition Workflow Standard - Process Definition Interface - XML Process Definition Language. Document Number WFMC-TC-1025, Final Approved. Verfügbar nach Anmeldung unter: http://wfmc.org/ [2009-03-20]

Workflow Management Coalition (1999): Workflow Management Coalition - Terminology & Glossary. Document Number WFMC-TC-1011, Issue 3.0. Verfügbar nach Anmeldung unter: http://wfmc.org/ [2009-03-20]

Workflow Management Coalition (1995): Workflow Management Coalition - The Workflow Reference Model. Document Number TC00-1003, Issue 1.1. Verfügbar nach Anmeldung unter: http://wfmc.org/ [2009-03-20]

# Anhang

## A Checkliste der Anforderungen

Zur besseren Übersicht der Anforderungen wird im Folgenden eine Checkliste derjenigen gegeben, die an das System gestellt werden. Die Bewertung dieser Anforderungen ist im Kapitel 6, der Evaluierung, zu finden.

| Nr | Bezeichnung |
|---|---|
| 1 | Anforderungen an Workflows |
| 1.1 | Workflows sollen in XML definiert werden |
| 1.2 | Workflows sollen die Definition von Rollen ermöglichen |
| 1.3 | Die Ausführung eines Workflows soll Bedingungen enthalten können, die in Zusammenhang mit digitalen Informationsobjekten stehen |
| 1.4 | Workflows sollen durch Benutzer des Systems gestartet werden können |
| 1.5 | Workflows sollen durch Administratoren beobachtet werden können |
| 2 | Anforderungen an die Workflow-Engine |
| 2.1 | Die Workflow-Engine soll Workflows laden können |
| 2.2 | Die Workflow-Engine soll Workflows instanzieren können |
| 2.3 | Die Workflow-Engine soll Workflows abarbeiten können |
| 2.4 | Die Workflow-Engine soll die Interaktion von Schritten eines Workflows mit einem Benutzer ermöglichen |
| 2.5 | Technologien wie sie im „Web 2.0“ verwendet werden können, sollen in der Benutzungsschnittstelle von Workflows benutzt werden können |
| 3 | Anforderungen an digitale Objekte und Objekttypen |
| 3.1 | Objekttypen sollen als ein Prototyp für digitale Informationsobjekte verwendbar sein |
| 3.2 | Objekttypen sollen mit anderen Objekttypen in Beziehung stehen können |
| 3.3 | Die Abfrage von Beziehungen zwischen Objekttypen soll ermöglicht werden |
| 3.4 | Objekttypen sollen mit Workflows assoziiert werden können |
| 3.5 | Digitale Objekte sollen mit einem Benutzer in Zusammenhang gebracht werden können |
| 4 | Anforderungen an Rechte und Rollen |
| 4.1 | Benutzer sollen Rollen besitzen können |
| 4.2 | Rollen sollen die Möglichkeiten zur Interaktion mit dem System einschränken |
| 4.3 | Der Zugriff auf bestimmte Objekttypen soll eingeschränkt werden können |
| 4.4 | Benutzerdaten und Rollen sollen in das System geladen werden können |
| 4.5 | Informationen über Rollen sollen aus einem LDAP Server entnommen werden können |
| 4.6 | Digitale Objekte sollen in Zusammenhang mit Gruppen stehen können |

| | |
|---|---|
| 5 | Anforderungen an die Implementierung |
| 5.1 | Das entwickelte System soll auf das Muradora Framework anwendbar sein |
| 5.2 | Die Implementierung des Systems soll in Java erfolgen |
| 5.3 | Die Grundprinzipien der SOA sollten beachtet werden |
| 5.4 | Die Anwendung soll plattformunabhängig sein |
| 5.5 | Die Anwendung soll speziell auf einem Ubuntu Linux Server lauffähig sein |
| 5.6 | Das System soll als Web-Anwendung realisiert werden |
| 5.7 | Als Web-Anwendung soll das System Struts2 unterstützen |
| 5.8 | Es soll eine Möglichkeit zur Internationalisierung gegeben werden |

| | |
|---|---|
| 6 | Anforderungen an das Szenario |
| 6.1 | Das gewählte Szenario zeigt einen Workflow zur Erstellung eines digitalen Objektes |
| 6.2 | Digitale Objekte sollen ein Schema für Metadaten erhalten |

| | |
|---|---|
| 7 | Anforderungen an Muradora |
| 7.1 | Dokumentation: Handbuch für Muradora |
| 7.2 | Dokumentation: Architektur und Aufbau |
| 7.3 | Dokumentation und Bereitstellung von Schnittstellen |
| 7.4 | Anbindung an das Fedora Repository |
| 7.5 | Benutzerfreundlichkeit des Frontends |
| 7.6 | Verwendbarkeit von Rechten und Rollen |
| 7.7 | Anbindung an LDAP Server für Rechte und Rollen |
| 7.8 | Administration von Rechten und Rollen |
| 7.9 | Eingabe und Bearbeitung von Metadaten |

**Tab. 8.1: Checkliste der Anforderungen**

## B XML Schema für Workflows

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- see XPDL Specification at http://www.wfmc.org/ -->
<schema
  xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:wf="http://www.object_workflow.de/2008/Workflow"
  targetNamespace="http://www.object_workflow.de/2008/Workflow"
  elementFormDefault="qualified">

  <!-- Wurzel-Element für einen Workflow -->
  <xsd:element name="Workflow">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Activities" type="wf:Activities"
          minOccurs="1" maxOccurs="1" />
        <xsd:element name="Transitions" type="wf:Transitions"
          minOccurs="1" maxOccurs="1" />
      </xsd:sequence>

      <xsd:attribute name="id" type="wf:Id" use="required" />
      <xsd:attribute name="name" type="xsd:string" use="optional" />
      <xsd:attribute name="description" type="xsd:string"
        use="optional" />
    </xsd:complexType>
  </xsd:element>

  <!-- Ein Workflow enthält eine Anzahl an Aktivitäten -->
  <xsd:complexType name="Activities">
    <xsd:sequence>
      <xsd:element name="Activity" type="wf:Activity"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <!-- Eine Activität kann mehrere Aktionen beinhalten und wird
    durch eine Reihe von Attributen beschrieben. -->
  <xsd:complexType name="Activity">
    <xsd:sequence>
      <xsd:element name="Action" type="wf:Action"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>

    <xsd:attribute name="id" type="wf:Id" use="required" />
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="description" type="xsd:string"
      use="optional" />
    <xsd:attribute name="role" type="xsd:string" use="required" />
    <xsd:attribute name="startActivity" type="xsd:boolean"
      use="required" />
    <xsd:attribute name="endActivity" type="xsd:boolean"
      use="required" />
  </xsd:complexType>
```

```
<!-- Eine Action beschreibt eine atomare Aktion innherhalb
   einer Aktivität. -->
<xsd:complexType name="Action">
   <xsd:attribute name="id" type="wf:Id" use="optional" />
   <xsd:attribute name="name" type="xsd:string" use="optional" />
   <xsd:attribute name="description" type="xsd:string"
     use="optional" />
   <xsd:attribute name="type" type="wf:ActionType" use="required" />
   <xsd:attribute name="resource" type="xsd:string" use="required" />
</xsd:complexType>

<!-- Aktionen können von verschiedenen Typen sein. -->
<xsd:simpleType name="ActionType">
   <xsd:restriction base="xsd:string">
     <xsd:enumeration value="Class" />
     <xsd:enumeration value="Method" />
     <xsd:enumeration value="Jsp" />
     <xsd:enumeration value="Form" />
     <xsd:enumeration value="WebService" />
   </xsd:restriction>
</xsd:simpleType>

<!-- Ein Workflow enthält eine Anzahl von Transitionen. -->
<xsd:complexType name="Transitions">
   <xsd:sequence>
     <xsd:element name="Transition" type="wf:Transition"
       minOccurs="0" maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>

<!-- Transitionen verbinden Aktivitäten miteinander. Eine Transition
   kann durch Bedingungen (Conditions) für bestimmte Zustände
   eingeschränkt werden. -->
<xsd:complexType name="Transition">
   <xsd:sequence>
     <xsd:element name="Condition" type="wf:Condition"
       minOccurs="0" maxOccurs="1" />
   </xsd:sequence>

   <xsd:attribute name="id" type="wf:Id" use="required" />
   <xsd:attribute name="name" type="xsd:string" use="optional" />
   <xsd:attribute name="description" type="xsd:string"
     use="optional" />
   <xsd:attribute name="from" type="wf:IdRef" use="required" />
   <xsd:attribute name="to" type="wf:IdRef" use="required" />
   <xsd:attribute name="role" type="xsd:string" use="required" />
</xsd:complexType>

<!-- Eine Bedingung ist in dieser Version ein String, der durch
   die Workflow-Engine interpretiert wird. -->
<xsd:complexType name="Condition">
   <xsd:simpleContent>
     <xsd:extension base="xsd:string" />
   </xsd:simpleContent>
</xsd:complexType>
```

```
  <!-- Beschreibungen und Ids -->
  <xsd:complexType name="Description">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string" />
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="Id"><xsd:restriction base="xsd:NMTOKEN" />
  </xsd:simpleType>

  <xsd:simpleType name="IdRef"><xsd:restriction base="xsd:NMTOKEN" />
  </xsd:simpleType>
</schema>
```

**Listing 8.1: XML Schema für Workflows**

## C XML Dokument des Workflows im Szenario

```
<?xml version="1.0" encoding="UTF-8"?>
<wf:Workflow id="epicPublication" name="" description=""
  xmlns:wf="http://www.object_workflow.de/2008/Workflow"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.object_workflow.de/2008/XMLSchema
  workflow.xsd ">

  <wf:Activities>
    <!-- Ereignis: Dokument liegt vor -->

    <!-- Aktivität: Dokument hochladen -->
    <wf:Activity id="upload" name="" role="" description=""
      startActivity="true" endActivity="false">
      <wf:Action id="uploadjsp" name="" type="Jsp" description=""
        resource="epic/upload.jsp;epic.UploadBean" />
    </wf:Activity>

    <!-- Ereignis: Dokument hochgeladen -->

    <!-- Aktivität: Metadaten eingeben -->
    <wf:Activity id="metadata" name="" role="" description=""
      startActivity="false" endActivity="false">
      <wf:Action id="metadatajsp" name="" type="Jsp" description=""
        resource="epic/metadata.jsp;epic.MetadataBean" />
    </wf:Activity>

    <!-- Ereignis: Dokument beschrieben, Metadaten sind eingegeben
      und vollständig -->

    <!-- Aktivität: Anfrage zur Publikation, setze Status auf
      "request" und speichere digitales Objekt -->
    <wf:Activity id="request" name="" role="" description=""
      startActivity="false" endActivity="false">
      <wf:Action id="request" name="" type="Method" description=""
        resource="workflows.epic.Workflow:request" />
      <wf:Action id="requestjsp" name="" type="Jsp" description=""
        resource="epic/request.jsp" />
    </wf:Activity>

    <!-- Ereignis: Anfrage gesendet, digitales Objekt gespeichert -->

    <!-- Aktivität: Status bearbeiten, setze Status auf "review"
      oder "published" -->
    <wf:Activity id="status" name="" role="curator" description=""
      startActivity="false" endActivity="false">
      <wf:Action id="statusjsp" name="" type="Jsp" description=""
        resource="epic/status.jsp;epic.StatusBean" />
    </wf:Activity>

    <!-- Ereignis: Status gesetzt -->

    <!-- Aktivität: digitales Objekt speichern -->
    <wf:Activity id="published" name="" role="curator" description=""
      startActivity="false" endActivity="true">
```

```
      <wf:Action id="publishjsp" name="" type="Jsp" description=""
        resource="epic/publish.jsp" />
      <wf:Action id="publish" name="" type="Method" description=""
        resource="workflows.epic.Workflow:publish" />
    </wf:Activity>

    <!-- Ereignis: digitales Objekt veröffentlicht -->
  </wf:Activities>

  <wf:Transitions>
    <wf:Transition id="t1" name="" role="" description=""
      from="upload" to="metadata">
    </wf:Transition>

    <wf:Transition id="t2" name="" role="" description=""
      from="metadata" to="request">
    </wf:Transition>

    <wf:Transition id="t3a" name="" role="curator" description=""
      from="request" to="status">
    </wf:Transition>

    <wf:Transition id="t3b" name="" role="" description=""
      from="request" to="upload">
    </wf:Transition>

    <wf:Transition id="t4a" name="" role="curator" description=""
      from="status" to="published">
      <wf:Condition>
        epic.MetadataBean:epic.recordStatus.name.toLowerCase ==
          "published"
      </wf:Condition>
    </wf:Transition>

    <wf:Transition id="t4b" name="" role="curator" description=""
      from="status" to="upload">
    </wf:Transition>
  </wf:Transitions>
</wf:Workflow>
```

**Listing 8.2: Workflow zur Veröffentlichung eines Dokuments im Szenario**

## D Verwendete Technologien

Die folgende Tabelle gibt eine Übersicht der verwendeten Technologien und die jeweilige Bezugsquelle an.

| Name | Version | Verwendung |
|---|---|---|
| Java Development Kit<br>http://java.sun.com/javase/6/ | 1.6.0_10 | Programmierung, Virtuelle Maschine |
| Fedora<br>http://fedora.info/ | 2.2.3 | Repository für digitale Objekte |
| Muradora<br>http://www.muradora.org/ | 1.3 | Frontend, GUI, Rechte |
| Apache Tomcat<br>http://tomcat.apache.org/ | 6.0.16 | Server für Web-Anwendungen |
| Apache Struts 2<br>http://struts.apache.org/2.x/ | 2.0.12 | MVC Framework |
| OpenSymphony SiteMesh<br>http://www.opensymphony.com/sitemesh/ | 2.3 | Decoration Framework |
| Jena<br>http://jena.sourceforge.net/ | 2.5.7 | Framework für Ontologien |
| Apache Solr<br>http://lucene.apache.org/solr/ | 1.3 | Index Search Server |
| Postgre SQL Server<br>http://www.postgresql.org/ | 8.3.5 | Datenbank Fedora und Muradora |
| Oracle Berkeley DB XML<br>http://www.oracle.com/database/berkeley-db/xml/ | 2.3.10 | Datenbank für XACML Dokumente |

**Tab. 8.2: Übersicht verwendeter Technologien**

Die angegebenen Technologien besitzen zum Teil Abhängigkeiten zu weiteren Technologien, die der Bezugsquelle entnommen werden können.

## CD mit Prototyp und Werkzeugen

## Abschließende Erklärung

Ich versichere hiermit, dass ich meine Diplomarbeit „Evaluierung eines Frameworks für das Workflowmanagement zur Pflege digitaler Informationsobjekte“ selbstständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Oldenburg, 26. März 2009